sgcUI .NET 2026.10

October 2026

.NET용 문서

Copyright © 2012-2026 eSeGeCe Software
info@esegece.com
www.esegece.com

Contents

sgcUI 26

sgcUI 개요 26
설치 27
플랫폼과 패키지 29
에디션 31
.NET 패키지 32

sgcPDF 34

소개 34
설치 36
아키텍처 37
.NET 설치 40
.NET API 차이점 42
.NET 데모 44
빠른 시작 46
TsgcPDFDocument 48
TsgcPDFEditor 52
TsgcPDFViewer 55
TsgcPDFTextExtractor 57
TsgcPDFExporter 59
TsgcPDFOptimizer 61
TsgcPDFSignatureHandler 63
TsgcPDFSignatureVerifier 65
텍스트와 폰트 66
이미지 68
그래픽, 투명도, 셰이딩 69
표 70
바코드 71

주석 73
북마크 79
레이어 80
폼 81
암호화 83
전자 서명 84
PDF/A 85
PDF 파일 최적화 86
페이지 병합, 분할 및 추출 87
텍스트 추출 88
렌더링과 내보내기 90
오류 처리 및 악의적인 입력 91
렌더러 아키텍처 92
TsgcPDFRendererRaster 94
TsgcPDFRendererGDI 97
지원하는 렌더링 기능 99
TsgcPDFViewerControl 101
TsgcPDFThumbnailView 108
TsgcPDFOutlineView 109
TsgcPDFCommentsView 110
TsgcPDFViewerFMX 112
TsgcPDFCommentsViewFMX 116
OpenType 폰트 118
복합 스크립트와 셰이핑 119
중국어, 일본어, 한국어 텍스트 121
텍스트 추출: 문자 위치 122
신뢰할 수 없는 파일에 대한 제한 124
이미지 디코더 125
TIFF 가져오기 127
TsgcPDFImageExtractor 128
TsgcPDFCanvas 131

EMF에서 PDF로 133
TsgcPDFVCLCanvasHelper 134
HTML에서 PDF로 135
2D 바코드와 GS1-128 138
TsgcPDFBarcodeControl 140
TsgcPDFBarcodeFMX 144
TsgcPDFBarcodeSymbol 147
Linux 152
macOS, iOS, Android 154
C++Builder 155
첨부 파일 156
Factur-X 및 ZUGFeRD 인보이스 158
증분 저장과 압축 출력 160
대용량 파일 162
FDF and XFDF 163
폼 값과 모양 164
PAdES 서명 166
타임스탬프와 장기 검증 168
서명 검증 170
TsgcPDFPAdESSigner(sgcSign 브리지) 171
인증서 암호화 173
TsgcPDFRedactor 175
태그된 PDF 178
PDF/UA와 PDF/A 레벨 A 181
접근성 검사기 182
FAQ 184

sgcReports 186

Introduction 186
설치 187
Getting Started 189

Report Model 190
Expressions and Text Templates 192
Data Sources 194
Groups, Headers, Footers and Columns 195
Fonts 196
Thread Safety and Server Use 197
Accessibility (PDF/UA) 198
PDF/A 199
TsgcReport 200
TsgcReportEngine 201
TsgcReportDataSetSource 202
TsgcReportJSONSource 203
TsgcReportEventSource 204
TsgcReportPDFExport 205
TsgcReportXLSXExport 206
TsgcReportCSVExport 207
TsgcReportImageExport 208
TsgcReportPreviewVCL 209
TsgcReportPreviewFMX 211
TsgcReportHTTPEndpoint 212
TsgcReportHTMLViewer 214
TsgcReportHTMLDesigner 215
TsgcReportBuilder 216
JSON Report Definitions 218
How To: Invoice with Master-Detail Data 219
How To: A Grouped Report with Totals 220
How To: Export to XLSX, CSV and PNG 221
How To: An Accessible PDF Validated with veraPDF 222
How To: Serve Reports Over HTTP 223
How To: A Linux64 Report Daemon 224
How To: A Mobile FMX Report 225

How To: The Web Report Designer 226

sgcCharts 227

Introduction 227
Architecture 228
Editions 230
설치 231
플랫폼 233
Quick Start 236
Demos 238
TsgcChart 241
TsgcChartSeries 243
Axes 246
Title, Legend and Margins 249
플롯 영역 251
색상 스케일 253
Themes and Palette 255
TsgcChartSeries_Line, TsgcChartSeries_FastLine, TsgcChartSeries_StepLine 256
TsgcChartSeries_Area 258
TsgcChartSeries_Bar 259
TsgcChartSeries_Scatter 260
TsgcChartSeries_Candlestick, TsgcChartSeries_OHLC 261
TsgcChartSeries_Pie 263
TsgcChartSeries_Radar 266
TsgcChartSeries_Heatmap 268
TsgcChartSeries_Treemap 270
TsgcChartSeries_Funnel 273
TsgcVCLChart 275
TsgcFMXChart 277
TsgcChartRenderer 279
TsgcChartInteraction 281

TsgcChartDataSource 285
TsgcChartDataSource_Manual 287
TsgcChartDataSource_DataSet 288
TsgcChartDataSource_JSON 289
TsgcChartDataSource_CSV 290
TsgcChartDataSource_Table 291
Bridge Overview 293
TsgcChartDataSource_MQTT 294
TsgcChartDataSource_WebSocket 295
TsgcChartDataSource_SSE 296
TsgcChartDataSource_Binance 297
TsgcChartSeries_Indicator 299
TsgcChartSeries_Oscillator 302
TsgcChartSeries_Volume 304
Panes 305
TsgcChartAggregator 307
TsgcChartDrawing 309
TsgcChartBuffer 312
Decimation 314
Canvas 315
Fonts 318
Indicators 321
.NET Overview 323
Quick Start: WinForms 325
Quick Start: WPF 327
Quick Start: Avalonia 329
Headless Rendering 332
Series Gallery 336
Bridge 342
.NET Differences from Delphi 346
Troubleshooting 348

sgcMaps 349

소개 349
개요 350
설치 352
빠른 시작 VCL 354
빠른 시작 FMX 355
데모 356
01 Offline Map 357
02 Map Features 358
03 Services 360
04 Offline Services 361
05 Static Render 362
06 FireMonkey 364
07 Vector Tiles 365
TsgcVCLMap 366
TsgcFMXMap 370
타일, 레이어, 공급자 373
벡터 타일 376
마커와 클러스터링 378
도형과 레이블 380
히트맵 382
GeoJSON, GPX, KML 383
지오메트리 함수 385
정적 지도 렌더링 386
오프라인 타일과 타일 다운로더 388
OpenStreetMap 리더 390
오프라인 지오코딩 인덱스 391
경로 그래프와 오프라인 라우팅 392
TsgcMapService 393
TsgcMapGeocoder 395
TsgcMapDirections 397

TsgcMapPlaces 399
TsgcMapElevation 401
TsgcMapTimeZone 403
TsgcMapRouteCalculator 405
TsgcMapOfflineRouter 408
TsgcMapOfflineGeocoder 409
TsgcMapGeoIndexBuilder 410
TsgcMapLocation 412
도구 414
GeoIndexBuilder 415
RouteGraphBuilder 417
Lazarus 418
.NET 개요 420
.NET 설치 423
빠른 시작: WinForms 425
헤드리스 렌더링 428
.NET 오프라인 데이터 431
클래스 매핑 435
.NET과 Delphi의 차이점 438
.NET 데모 441
문제 해결 443

sgcDashboard 446

소개 446
아키텍처 447
에디션 449
설치 450
플랫폼 452
빠른 시작 VCL 455
빠른 시작 FMX 456
빠른 시작 C++Builder 457

데모 458
TsgcVCLDashboard 459
TsgcVCLDashboardTile 461
TsgcFMXDashboard 463
TsgcFMXDashboardTile 465
TsgcDashboardKPI 467
TsgcVCLKPICard 469
TsgcFMXKPICard 471
TsgcDashboardSource 473
TsgcDashboardSource_WebSocket 474
TsgcDashboardSource_MQTT 476
TsgcDashboardSource_SSE 478
레이아웃 480
테마 481
영속성 482
실시간 데이터 483
헤드리스 KPI 이미지 484
.NET 개요 485
NuGet에서 설치 487
WinForms 빠른 시작 489
WPF 빠른 시작 493
Avalonia 빠른 시작 495
.NET 클래스 498
Delphi와 .NET의 차이점 501
.NET 데이터 바인딩 503
.NET 헤드리스 렌더링 506
.NET 라이브 피드 508
.NET 데모 513
문제 해결 515

sgcGrid 516

소개 516
아키텍처 517
에디션 519
설치 520
플랫폼 522
성능 525
빠른 시작 VCL 526
빠른 시작 FMX 527
빠른 시작 C++Builder 528
데모 529
TsgcVCLGrid 530
TsgcFMXGrid 535
TsgcGridColumn 540
TsgcGridColumns 542
TsgcGridData 543
TsgcGridLineView 546
TsgcGridDataStore 549
TsgcGridRenderer 550
TsgcGridFormatRule 552
TsgcGridFormatRules 553
IsgcGridDataHost와 IsgcUITable 554
TsgcGridSource_DataSet 555
TsgcGridFeedSource 556
TsgcGridFeedSource_WebSocket 558
TsgcGridFeedSource_MQTT 560
TsgcGridFeedSource_SSE 562
TsgcGridAIQuery 564
실시간 갱신과 스레딩 565
정렬 566
필터링 567
그룹화 568

Summaries 570
트리 행 572
마스터 디테일 574
선택과 키보드 576
열 크기 조정 577
고정 열 578
변경 시 깜박임 579
조건부 서식 580
제자리 편집 581
가상 모드 582
JSON과 CSV 583
Excel 내보내기 584
DataSet 바인딩 586
실시간 피드 587
그리드에 질문하기 588
그리드에서 차트 만들기 589
테마 590
헤드리스 렌더링 591
.NET 개요 592
NuGet에서 설치 595
WinForms 빠른 시작 597
WPF 빠른 시작 599
Avalonia 빠른 시작 601
.NET 클래스 604
Delphi와 .NET의 차이점 608
.NET 데이터 바인딩 610
.NET 헤드리스 렌더링 615
.NET DataTable 바인딩, JSON 및 CSV 617
.NET 제자리 편집 620
.NET 실시간 피드 622
.NET 그리드에 질문하기 625

.NET 성능 627
.NET 그룹화와 요약 629
.NET 트리 행과 마스터 디테일 634
.NET Excel 내보내기 638
.NET 데모 641
문제 해결 643

sgcNavigation 645

소개 645
아키텍처 647
에디션 648
설치 649
플랫폼 651
VCL 빠른 시작 654
FMX 빠른 시작 655
C++Builder 빠른 시작 656
데모 657
TsgcVCLNavigationBar 658
TsgcFMXNavigationBar 660
TsgcNavigationItem 662
TsgcNavigationItems 664
TsgcNavigationModel 665
사이드바 모드 667
아코디언 모드 668
접기와 애니메이션 669
이미지와 글리프 670
배지 671
키보드와 포커스 672
Action 링크 673
테마 674
헤드리스 렌더링 675

.NET 개요 677
NuGet으로 설치 679
WinForms 빠른 시작 680
WPF 빠른 시작 683
Avalonia 빠른 시작 685
.NET 클래스 688
Delphi와 .NET의 차이점 690
.NET 데이터 바인딩 693
.NET 헤드리스 렌더링 697
.NET 데모 699
문제 해결 701

sgcToolbar 702

소개 702
아키텍처 703
에디션 704
설치 705
플랫폼 707
빠른 시작 VCL 710
빠른 시작 FMX 711
빠른 시작 C++Builder 712
데모 713
TsgcVCLToolbar 714
TsgcFMXToolbar 716
TsgcVCLStatusBar 718
TsgcFMXStatusBar 720
TsgcToolbarItem 722
TsgcToolbarItems 724
TsgcStatusBarPanel 725
TsgcStatusBarPanels 726
TsgcToolbarModel 727

TsgcStatusBarModel 729
항목 종류 731
토글과 라디오 그룹 732
드롭다운 메뉴 733
오버플로 734
Action 링크 735
이미지와 글리프 736
상태 패널 737
진행률과 표시기 738
스레드 안전 갱신 739
크기 조절 핸들 740
키보드 741
테마 742
헤드리스 렌더링 743
.NET 개요 744
NuGet 설치 746
빠른 시작 WinForms 747
빠른 시작 WPF 750
빠른 시작 Avalonia 753
.NET 클래스 756
Delphi와 .NET의 차이점 759
.NET 데이터 바인딩 764
.NET 헤드리스 렌더링 767
.NET 데모 769
문제 해결 771

sgcWizard 773

소개 773
아키텍처 774
에디션 775
설치 776

플랫폼 778
빠른 시작 VCL 781
빠른 시작 FMX 782
빠른 시작 C++Builder 783
데모 784
TsgcVCLWizard 785
TsgcVCLWizardPage 787
TsgcFMXWizard 788
TsgcFMXWizardPage 790
TsgcVCLStepper 791
TsgcFMXStepper 793
TsgcWizardStep 795
TsgcWizardSteps 796
TsgcStepperModel 797
탐색과 건너뛴 페이지 799
유효성 검사 800
Finish와 Cancel 801
키보드 802
단계 상태 803
스테퍼 레이아웃 804
버튼 캡션과 현지화 805
테마 806
디자인 타임 807
헤드리스 렌더링 808
.NET 개요 809
NuGet에서 설치 811
빠른 시작 WinForms 812
빠른 시작 WPF 814
빠른 시작 Avalonia 816
.NET 클래스 819
Delphi와 .NET의 차이점 822

.NET 데이터 바인딩 824
.NET 헤드리스 렌더링 828
.NET 데모 830
문제 해결 831

sgcPivotGrid 832

소개 832
아키텍처 833
에디션 835
설치 836
플랫폼 838
빠른 시작 VCL 841
빠른 시작 FMX 842
빠른 시작 C++Builder 843
빠른 시작 DataSet 844
빠른 시작 헤드리스 845
데모 846
TsgcVCLPivotGrid 847
TsgcFMXPivotGrid 850
TsgcVCLPivotFieldList 853
TsgcFMXPivotFieldList 855
TsgcPivotEngine 857
TsgcPivotField 860
TsgcPivotCalculatedField 862
TsgcPivotSource_DataSet 863
TsgcPivotSource_Table 864
TsgcPivotGridRenderer 865
TsgcPivotFormatRule 867
영역 868
집계 869
날짜 그룹화 870

합계 871
필터 872
Top N 874
값 표시 형식 875
계산 필드 876
레이아웃 저장 877
내보내기 878
확장과 축소 879
정렬 881
드릴다운 882
조건부 서식 883
차트 연동 884
키보드와 접근성 886
성능 887
TsgcPivotGridAIQuery 888
피벗에 질문하기 891
.NET 개요 894
NuGet에서 설치 896
WinForms 빠른 시작 897
WPF 빠른 시작 900
Avalonia 빠른 시작 903
.NET 클래스 906
.NET과 Delphi의 차이점 909
.NET 데이터 바인딩 912
.NET 헤드리스 렌더링 915
.NET 데모 918
문제 해결 919

sgcPlanner 921

소개 921
아키텍처 922

에디션 925
설치 927
플랫폼 929
VCL 빠른 시작 933
FMX 빠른 시작 934
C++Builder 빠른 시작 935
데모 936
TsgcPlannerData 938
TsgcVCLScheduler 942
TsgcFMXScheduler 945
TsgcVCLGantt 948
TsgcFMXGantt 951
TsgcVCLKanban 954
TsgcFMXKanban 957
TsgcPlannerSource_DataSet 960
TsgcPlannerICalendar 962
TsgcPlannerMSProject 964
TsgcPlannerRenderer 966
공유 데이터 968
동기화된 뷰 970
근무 달력 971
반복 973
시간 눈금과 확대/축소 975
끌어서 놓기 977
Dependencies 979
자동 일정 계산 980
크리티컬 패스 981
기준선 982
스윔레인과 WIP 제한 983
DataSet 바인딩 985
JSON 저장 986

iCalendar 987
MS Project XML 989
헤드리스 렌더링 991
테마 993
에디션별 기능 994
Google 캘린더 동기화 996
TsgcPlannerGoogleSync 999
TsgcPlannerICalSubscription 1003
.NET 개요 1006
NuGet으로 설치 1008
WinForms 빠른 시작 1009
WPF 빠른 시작 1011
Avalonia 빠른 시작 1013
.NET 클래스 1015
Delphi와 .NET의 차이 1018
.NET 데이터와 일정 계산 1020
.NET 데이터 바인딩 1023
.NET JSON, iCalendar, MS Project 1025
.NET 헤드리스 렌더링 1027
.NET의 Google 캘린더 동기화와 iCalendar 1029
.NET 데모 1031
문제 해결 1032

sgcEditors 1034

소개 1034
아키텍처 1036
컨트롤 1039
설치 1041
플랫폼 1043
빠른 시작 VCL 1046
빠른 시작 FMX 1047

빠른 시작 C++Builder 1048
빠른 시작 데이터 인식 1049
월 달력 1050
날짜 선택기 1053
시간 선택기 1056
날짜 및 시간 선택기 1059
범위 슬라이더 1062
색상 선택기 1064
토큰 편집기 1066
평점 1069
스핀 편집기 1071
DB 날짜 선택기 1073
DB 시간 선택기 1075
DB 스핀 편집기 1077
DB 평점 1079
DB 범위 슬라이더 1081
TsgcEditorModel 1083
TsgcEditorLocale 1086
TsgcEditorFieldLink 1088
텍스트 입력과 IME 1090
키보드와 접근성 1092
테마 1094
현지화 1095
DPI 1096
헤드리스 출력 1097
에디션 1098
.NET 개요 1099
NuGet 설치 1101
빠른 시작 WinForms 1102
빠른 시작 WPF 1104
빠른 시작 Avalonia 1106

.NET 클래스 1108
Delphi와 .NET의 차이점 1111
.NET 데이터 바인딩 1113
.NET 헤드리스 출력 1116
.NET 데모 1118
문제 해결 1119

sgcGauges 1121

소개 1121
아키텍처 1122
게이지 유형 1124
설치 1126
플랫폼 1128
빠른 시작 VCL 1131
빠른 시작 FMX 1132
빠른 시작 헤드리스 1133
빠른 시작 C++Builder 1134
방사형 게이지 1135
선형 게이지 1137
디지털 게이지 1139
LED 1141
LED 막대 1142
진행 링 1143
VU 미터 1145
시계 1147
나침반 1149
애넌시에이터 1151
TsgcCustomGauge 1153
TsgcGaugeScale와 TsgcGaugeRange 1155
TsgcGaugePointer, TsgcGaugeThreshold 및 TsgcGaugeAnimation 1157
스케일 1159

범위 1160
포인터 1161
임계값 1163
애니메이션 1164
테마 1165
헤드리스 출력 1166
에디션 1167
브리지 개요 1168
MQTT 소스 1170
WebSocket 소스 1171
서버 전송 이벤트 소스 1172
.NET 개요 1173
NuGet 설치 1175
빠른 시작 WinForms 1176
빠른 시작 WPF 1178
빠른 시작 Avalonia 1180
.NET 클래스 1182
Delphi와 .NET의 차이점 1185
.NET 데이터 바인딩 1187
.NET 헤드리스 출력 1190
.NET sgcWebSockets 브리지 1192
.NET 데모 1194
문제 해결 1195

sgcDiagram 1197

소개 1197
sgcDiagram이란 1199
아키텍처 1200
에디션 1202
플랫폼 1203
빠른 시작 VCL 1207

빠른 시작 FMX 1208
빠른 시작 C++Builder 1209
빠른 시작 헤드리스 1210
TsgcVCLDiagramView 1211
TsgcFMXDiagramView 1213
TsgcVCLDiagramPalette 1214
TsgcFMXDiagramPalette 1216
TsgcDiagramDataBinding 1217
이벤트 1219
TsgcDiagramDocument 1221
TsgcDiagramNode 1223
TsgcDiagramConnector 1225
TsgcDiagramPort 1227
TsgcDiagramStyle 1228
TsgcDiagramLayer 1229
TsgcDiagramController 1230
TsgcDiagramRenderer 1232
JSON 형식 1233
계층형 레이아웃 1236
트리 레이아웃 1237
데이터 바인딩 1238
도형 1239
커넥터와 라우팅 1241
포트와 접착 1242
편집과 키보드 1243
스냅과 가이드 1245
실행 취소와 클립보드 1246
레이어 1247
직렬화 1248
내보내기 1250
테마 1251

뷰어 모드 1252
.NET 개요 1253
NuGet 설치 1255
빠른 시작 WinForms 1256
빠른 시작 WPF 1258
빠른 시작 Avalonia 1260
.NET 클래스 1262
Delphi와 .NET의 차이점 1264
.NET 데이터 바인딩 1266
.NET 헤드리스 출력 1268
.NET 데모 1270
문제 해결 1271

Index 1272

sgcUI 개요

sgcUI는 Delphi와 C++Builder용 eSeGeCe의 비주얼 및 문서 컴포넌트 제품군입니다. 제품들은 하나의 소스 폴더, 하나의 버전 include 파일, 하나의 설치 프로그램, 그리고 이 도움말을 공유합니다. 각 제품은 개별적으로 라이선스가 부여되며, 설치 프로그램은 사용자의 라이선스에 포함된 제품만 설치합니다.

제품 목록

제품	용도
sgcPDF	네이티브 PDF 라이브러리입니다. 텍스트와 글꼴, 그리기, 이미지, 보안, 접근성, 뷰어 컨트롤을 갖추고 PDF 문서를 만들고 편집하고 렌더링합니다. sgcPDF를 참조하십시오.
sgcReports	sgcPDF를 기반으로 한 헤드리스 보고서 엔진입니다. VCL 및 FMX 애플리케이션은 물론 Windows 서비스, Linux64 데몬, HTTP 엔드포인트에서도 사용할 수 있습니다. sgcReports를 참조하십시오.
sgcCharts	실시간 데이터 차트입니다. 데이터 소스가 점 또는 OHLC 캔들을 링 버퍼에 추가하고, 차트는 디시메이션된 스냅숏을 그리므로 빠른 피드에서도 부드럽게 렌더링됩니다. sgcCharts를 참조하십시오.
sgcMaps	대화형 지도입니다. 지도 컨트롤은 브라우저 컨트롤 없이 래스터 및 벡터 타일을 네이티브로 그리고, 그 위에 마커, 경로, 도형, 레이블, 히트맵을 표시합니다. sgcMaps를 참조하십시오.
sgcDashboard	타일 그리드로 구성된 대시보드입니다. 각 타일에는 차트, 지도, PDF 뷰어, 보고서 미리 보기 등 모든 컨트롤을 넣을 수 있습니다. sgcDashboard를 참조하십시오.
sgcGrid	실시간 데이터 그리드입니다. 행은 어떤 스레드에서든 키로 갱신되고, 변경된 셀은 깜박이며 사라지고, 한꺼번에 몰리는 갱신은 하나로 합쳐져 타이머에 맞춰 다시 그려집니다. sgcGrid를 참조하십시오.
sgcNavigation	애플리케이션의 페이지를 아이콘, 캡션, 배지, 그룹 머리글이 있는 항목 목록으로 보여 주는 내비게이션 바입니다. sgcNavigation을 참조하십시오.
sgcToolbar	버튼, 토글, 라디오 그룹, 드롭다운 버튼, 레이블, 구분선, 스페이서가 있는 툴바와 상태 표시줄입니다. sgcToolbar를 참조하십시오.
sgcWizard	단계별 마법사와 단계 표시기입니다. 스테퍼 밴드와 버튼 바가 있는 페이지 컨테이너입니다. sgcWizard를 참조하십시오.
sgcUICore	sgcCharts, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard의 공통 기반입니다. 단독으로 판매되거나 선택되지 않으며, 이 제품들 중 하나를 설치하면 설치 프로그램이 자동으로 추가합니다.

다음에 볼 내용

- 설치에서는 sgcUI 설치 프로그램과 제품을 선택하는 방법을 설명합니다.
- 에디션에서는 각 제품의 Registered, Trial, Community 에디션을 나열합니다.
- .NET 패키지에서는 각 제품의 NuGet 패키지를 나열합니다.

설치

모든 sgcUI 제품은 하나의 설치 프로그램으로 설치됩니다. 제품마다 별도의 설치 프로그램을 다운로드하거나 실행할 필요가 없습니다.

로그인

Registered 설치 프로그램은 eSeGeCe 계정의 사용자 이름과 암호로 로그인하도록 요청합니다. 이전 설치에서 저장된 라이선스를 다시 사용하거나 오프라인 라이선스를 활성화할 수도 있습니다. 라이선스 서버는 계정이 보유한 제품을 반환하며, All-Access 라이선스는 모든 제품을 보유합니다.

제품 선택

컴포넌트 페이지에는 제품마다 확인란이 하나씩 표시됩니다. 라이선스에 포함되지 않은 제품은 비활성화되고 선택되지 않은 상태로 남습니다. 설치할 제품을 데모 및 도움말 파일과 함께 선택하십시오.
일부 제품은 다른 제품을 필요로 하며, 설치 프로그램이 자동으로 추가합니다:

- sgcReports를 선택하면 그 기반 엔진인 sgcPDF도 함께 설치됩니다.
- sgcCharts, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard는 sgcUICore를 기반으로 하므로, 이 중 하나를 선택하면 sgcUICore가 설치됩니다.

sgcWebSockets 통합

sgcWebSockets bridges 그룹에서는 sgcCharts for sgcWebSockets, sgcDashboard for sgcWebSockets, sgcGrid for sgcWebSockets, sgcReports HTML for sgcWebSockets를 제공합니다. 브리지는 sgcWebSockets가 이미 컴퓨터에 설치되어 있고 브리지가 확장하는 제품을 보유한 경우에만 사용할 수 있습니다. 그렇지 않으면 비활성화됩니다.

패키지 컴파일

파일을 복사한 후 설치 프로그램은 선택한 모든 Delphi 및 C++Builder 버전에 대해 사용자의 컴퓨터에서 패키지를 컴파일하고, 디자인 타임 패키지를 IDE에 등록합니다. 패키지는 의존 순서대로 컴파일됩니다. 먼저 sgcUICore, 그다음 그 위에 구축된 제품, 그다음 sgcPDF, sgcReports, sgcMaps, 마지막으로 sgcWebSockets 브리지입니다.

독립 실행형 설치에서 이전하기

이전 버전의 sgcPDF, sgcReports, sgcMaps, sgcCharts에는 각각 전용 설치 프로그램이 있었습니다. sgcUI 설치 프로그램이 그러한 설치를 발견하면, 이전 설치가 각 IDE에 추가한 라이브러리 경로와 디자인 패키지를 나열하고, 사용자가 해당 페이지를 확인한 후에만 제거합니다. 독립 실행형 제품 폴더를 여전히 가리키는 검색 경로 및 탐색 경로 항목도 제거되므로, IDE가 한 유닛의 사본을 둘 찾는 일은 없습니다.

Trial 및 Community 에디션

- Trial 설치 프로그램은 라이선스 없이 모든 제품을 트라이얼 에디션으로 설치합니다. Delphi용 Trial 설치 프로그램 하나와 C++Builder용 하나가 있습니다.
- Community 설치 프로그램은 sgcCharts, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard를 sgcUICore와 함께 설치합니다. sgcPDF에는 Community 에디션이 없으므로, 라이선스가 있는 sgcPDF가 이미 설치되어 있는 경우에만 sgcReports도 제공합니다. Community 설치 프로그램은 sgcWebSockets 브리지를 제공하지 않습니다. 에디션을 참조하십시오.

제품 세부 정보

각 제품은 자체 설치 항목에서 패키지, 요구 사항, 설치되는 파일을 설명합니다:

- sgcPDF

- sgcReports
- sgcCharts
- sgcMaps
- sgcDashboard
- sgcGrid
- sgcNavigation
- sgcToolbar
- sgcWizard

플랫폼과 패키지

sgcUICore와 모든 sgcUI 컴포넌트 제품에는 프레임워크에 독립적인 런타임 패키지(sgc<Product>), VCL 런타임 패키지(sgc<Product>VCL), XE6부터의 FireMonkey 런타임 패키지(sgc<Product>FMX), 그리고 VCL 및 FireMonkey 디자인 타임 패키지(dclsgc<Product>와 dclsgc<Product>FMX)가 들어 있습니다. sgcUICore에는 디자인 타임 패키지가 없습니다. 프레임워크 독립 패키지는 Win32와 Win64용으로 빌드되며, Delphi 10.3부터는 Linux64와 macOS용으로도 빌드됩니다. FireMonkey 패키지는 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터 빌드됩니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 각 제품의 패키지, IDE 버전, 대상 플랫폼은 아래 제품별 항목에 나와 있습니다.

제품	항목
sgcCharts	플랫폼
sgcDashboard	플랫폼
sgcGrid	플랫폼
sgcNavigation	플랫폼
sgcToolbar	플랫폼
sgcWizard	플랫폼
sgcPivotGrid	플랫폼
sgcPlanner	플랫폼
sgcEditors	플랫폼
sgcGauges	플랫폼
sgcDiagram	플랫폼

C++Builder Win64: 런타임 패키지

C++Builder Win64(클래식 툴체인, C++Builder 12와 13): sgcUI Win64 런타임 패키지는 Delphi 컴파일러가 링크하므로, 애플리케이션이 런타임 패키지를 사용할 때도 sgcUI 코드 안에서 처리되는 예외는 정상적으로 동작합니다. C++ Win64 링커(ilink64)에는 여전히 자체 결함이 있습니다. 런타임 패키지로 빌드한 애플리케이션에 정적으로 링크된 Delphi 코드는, 클래스가 런타임 패키지에 있는 예외를 `except on E: <class>` 블록이 처음 잡을 때 `_DelphiExceptionHandler`에서 충돌합니다.

- 런타임 패키지와 링크가 활성화된 경우(Project Options, Packages, Runtime Packages) 프로젝트가 사용하는 sgcUI 런타임 패키지를 모두 나열하여 sgcUI 유닛이 실행 파일에 링크되지 않도록 합니다.
- .cbproj에서는 이 패키지들이 각각 PackageImports와 LinkPackageImports 양쪽에 있어야 합니다. PackageImports에만 나열된 패키지는 링크되지 않습니다.
- 런타임 패키지와 링크가 비활성화된 경우에는 모든 것이 정적으로 링크되어 정상적으로 동작합니다.
- Win32와 Win64x(Modern)는 영향을 받지 않습니다.

C++Builder 10.3 Rio, Win64: 보조 스레드의 예외

Delphi 예외가 보조 스레드에서 발생하고(예: sgcWebSockets 스레드나 임의의 TThread에서 실행되는 이벤트 핸들러 안), 그 스레드의 C++ 코드가 이를 잡거나, 임시 UnicodeString 같은 로컬 객체를 소멸시키는 C++ 함수를 통과하며 언와인드하면 애플리케이션이 액세스 위반으로 충돌합니다. 이것은 C++Builder 10.3 Win64 런타임(cc64260mt.dll)의 결함입니다. sgc 코드 없이도 발생하며, C++Builder 10.1, 10.2, 10.4, 11, 12, 13은 영향을 받지 않습니다. 해결 방법은 C++Builder 10.4 이상을 사용하거나, 보조 스레드에서 실행되는 Win64 코드에서 예를 들어 StrToInt 대신 TryStrToInt를 사용해 예외를 발생시키지 않는 것입니다.

C++Builder 13 Win64x: 패키지 파일 이름

Win64x(Modern) 런타임 패키지는 클래식 Win64 패키지와 파일 이름이 같고(예: sgcUICoreC13.bpl) 둘 다 64비트이므로 Windows가 둘을 구별하지 못합니다. 런타임 패키지를 사용하는 Win64x 애플리케이션이 Win64 bpl을 먼저 찾으면 시작에 실패합니다(0xC0000139, 진입점을 찾을 수 없음). Win64x bpl 폴더(Bpl\Win64x)를 Win64 폴더보다 앞서 PATH에 두거나, Win64x bpl을 실행 파일과 같은 폴더에 배포하십시오. 런타임 패키지 없이 링크한 애플리케이션은 영향을 받지 않습니다.

Delphi XE7 FireMonkey: FMX.Forms를 먼저

Delphi XE7에만 해당합니다. FMX.Types 또는 sgcUI FMX 유닛이 프로그램의 uses 순서에서 FMX.Forms보다 앞에 있고 스타일이 적용된 컨트롤을 사용했다면, FireMonkey 프로그램이 종료할 때 "Runtime error 216"으로 끝날 수 있습니다(FMX가 기본 스타일보다 먼저 피커 서비스를 해제하기 때문입니다). XE6, XE8 이후 버전은 영향을 받지 않습니다. 해결 방법은 IDE가 새 FMX 애플리케이션에서 하는 것처럼 .dpr의 uses 절 첫 번째 유닛으로 FMX.Forms를 나열하는 것입니다.

PaintTo로 VCL 폼 캡처하기

sgcUI 컨트롤이 있는 VCL 폼을 PaintTo로 비트맵에 캡처하려면 호출 전후로 비트맵 캔버스를 잠그십시오(`Bitmap.Canvas.Lock`, 이어서 `Unlock`). 잠그지 않은 캔버스에 PaintTo를 하면 캡처 중에 폼의 네이티브 컨트롤이 메시지를 처리할 때 디바이스 컨텍스트를 잃을 수 있으며, 그러면 비트맵이 비어 있거나 일부만 나옵니다. FireMonkey 폼에서는 대신 MakeScreenshot을 사용합니다.

참고 항목

- 설치
- 에디션

에디션

모든 sgcUI 제품은 각 에디션에서 동일한 소스로 빌드됩니다. 에디션은 빌드에 포함되는 유닛과 기능만 결정합니다.

- **Registered**: 라이선스가 보유한 제품의 모든 유닛이며 sgcWebSockets 브리지를 포함합니다.
- **Trial**: 평가용 모든 제품입니다. 트라이얼 빌드는 `SGC_TRIAL` 정의로 컴파일됩니다.
- **Community**: 아래에 나열된 제품을 위한 무료 에디션으로, `SGC_COMMUNITY` 정의로 컴파일됩니다. sgcWebSockets 브리지는 포함되지 않습니다.

제품별 Community 에디션

제품	Community	Community에 포함되지 않는 항목
sgcPDF	아니요	Community 설치 프로그램은 sgcPDF를 제공하지 않습니다.
sgcReports	예, 라이선스가 있는 sgcPDF가 필요합니다	sgcWebSockets용 sgcReports HTML 브리지입니다. sgcReports Community는 라이선스가 있는 sgcPDF가 이미 설치되어 있는 경우에만 제공됩니다.
sgcCharts	예	재무 기능: 지표, 재무 계열, 집계기, 그리기 도구. sgcWebSockets 브리지.
sgcMaps	아니요	Community 설치 프로그램은 sgcMaps를 제공하지 않습니다.
sgcDashboard	예	런타임에서 타일 끌기 및 크기 조정, SaveLayout과 LoadLayout. sgcWebSockets 브리지.
sgcGrid	예	셀 제자리 편집. sgcWebSockets 브리지.
sgcNavigation	예	탐색 항목의 Action 링크.
sgcToolbar	예	오버플로 메뉴와 Action 링크.
sgcWizard	예	없음. Community 빌드에는 전체 기능이 포함됩니다.
sgcUICore	예	필요로 하는 Community 제품과 함께 설치됩니다.

각 제품은 자체 도움말에서 에디션을 더 자세히 설명합니다. 예를 들어 sgcCharts 에디션을 참조하십시오.

.NET 패키지

sgcUI 제품의 .NET 버전은 NuGet 패키지로 배포됩니다. 각 제품에는 제품 이름을 딴 코어 패키지와 UI 프레임워크나 통합을 위한 애드온 패키지가 있습니다. 코어 패키지와 애플리케이션에 필요한 애드온을 추가하십시오.

제품	코어 패키지	애드온 패키지
sgcPDF	`esegece.sgcPDF`	`esegece.sgcPDF.WinForms`, `esegece.sgcPDF.sgcSign`
sgcReports	`esegece.sgcReports`	`esegece.sgcReports.WinForms`, `esegece.sgcReports.AspNetCore`, `esegece.sgcReports.HTML`, `esegece.sgcReports.sgcSign`
sgcCharts	`esegece.sgcCharts`	`esegece.sgcCharts.Rendering`, `esegece.sgcCharts.WinForms`, `esegece.sgcCharts.Wpf`, `esegece.sgcCharts.Avalonia`, `esegece.sgcCharts.sgcWebSockets`
sgcMaps	`esegece.sgcMaps`	`esegece.sgcMaps.Rendering`, `esegece.sgcMaps.WinForms`, `esegece.sgcMaps.Offline`
sgcDashboard	`esegece.sgcDashboard`	`esegece.sgcDashboard.Rendering`, `esegece.sgcDashboard.WinForms`, `esegece.sgcDashboard.Wpf`, `esegece.sgcDashboard.Avalonia`, `esegece.sgcDashboard.sgcWebSockets`
sgcGrid	`esegece.sgcGrid`	`esegece.sgcGrid.Rendering`, `esegece.sgcGrid.WinForms`, `esegece.sgcGrid.Wpf`, `esegece.sgcGrid.Avalonia`, `esegece.sgcGrid.sgcWebSockets`, `esegece.sgcGrid.sgcHTML`
sgcNavigation	`esegece.sgcNavigation`	`esegece.sgcNavigation.Rendering`, `esegece.sgcNavigation.WinForms`, `esegece.sgcNavigation.Wpf`, `esegece.sgcNavigation.Avalonia`
sgcToolbar	`esegece.sgcToolbar`	`esegece.sgcToolbar.Rendering`, `esegece.sgcToolbar.WinForms`, `esegece.sgcToolbar.Wpf`, `esegece.sgcToolbar.Avalonia`
sgcWizard	`esegece.sgcWizard`	`esegece.sgcWizard.Rendering`, `esegece.sgcWizard.WinForms`, `esegece.sgcWizard.Wpf`, `esegece.sgcWizard.Avalonia`
sgcUICore	`esegece.sgcUICore`	`esegece.sgcUICore.Rendering`, `esegece.sgcUICore.WinForms`, `esegece.sgcUICore.Wpf`, `esegece.sgcUICore.Avalonia`. 이 패키지들은 sgcUICore를 기반으로 하는 제품의 종속성이므로 NuGet이 자동으로 복원합니다.

Community 패키지

패키지의 Community 빌드는 동일한 ID에 `.Community` 접미사가 붙습니다. 예를 들어 `esegece.sgcGrid.Community`입니다. 각 제품의 Community 에디션에서 빠지는 항목은 에디션을 참조하십시오.

소개

sgcPDF .NET은 PDF 문서를 생성, 열기, 편집, 서명, 렌더링, 표시하기 위한 .NET 라이브러리입니다. sgcPDF Delphi 라이브러리를 동일한 클래스 이름으로 완전히 매니지드로 이식한 버전이며 NuGet 패키지로 배포되므로, 배포해야 할 PDF 엔진 DLL도 없고 라이선스가 필요한 서드파티 라이브러리도 없습니다.

주요 기능

- 코드로 PDF 문서를 생성합니다. 14가지 표준 폰트 또는 서브셋을 지원하는 임베디드 TrueType과 OpenType(CFF) 폰트로 텍스트를 작성하고, CJK를 포함한 유니코드 텍스트, 복합 스크립트 셰이핑(아랍어 연결, 리가처, 오른쪽에서 왼쪽으로 쓰는 텍스트), 선, 곡선, 이미지, 투명도, 그라디언트, 패턴을 사용할 수 있습니다.
- TCanvas와 유사한 API(TsgcPDFCanvas)로 그리기, EMF 및 VCL 캔버스 드로잉을 벡터 PDF로 변환, HTML과 CSS를 PDF 페이지로 변환하는 기능을 제공합니다.
- 이미지: JPEG, PNG, BMP 및 다중 페이지 TIFF 가져오기, JPEG(베이스라인 및 프로그레시브), JPEG 2000, CCITT G3/G4, JBIG2용 순수 Pascal 디코더, PNG, JPEG, BMP로의 이미지 추출을 지원합니다.
- 머리글, 테두리, 교차 행 색상, 자동 페이지 나누기를 지원하는 표를 사용할 수 있습니다.
- 바코드: Code 128, GS1-128, Code 39, EAN-13, EAN-8, UPC-A, ITF, Codabar, QR Code(버전 1~40), DataMatrix, PDF417을 지원합니다.
- 주석, 북마크(개요), 선택적 콘텐츠 레이어, 대화형 폼: 생성, 채우기, 평면화, 다중 선택 목록 상자, 유니코드 필드 모양, FDF 및 XFDF 가져오기/내보내기를 지원합니다.
- 기존 문서를 열어 편집하고, 워터마크, 머리글, 바닥글, 페이지 번호를 추가하며, 페이지를 병합, 분할, 추출, 회전, 삭제할 수 있습니다.
- 서명을 그대로 유지하는 증분 저장, 객체 스트림과 상호 참조 스트림을 사용한 압축 출력, 2GB보다 큰 파일의 온디맨드 읽기를 지원합니다.
- PDF/A-3 관계를 갖는 파일 임베드, 그리고 Factur-X, ZUGFeRD, XRechnung 전자 인보이스를 지원합니다.
- 모든 문자의 위치, 폰트, 크기를 포함한 텍스트 추출을 지원합니다.
- 암호(RC4 40비트 및 128비트, AES-128, AES-256 리비전 6) 또는 수신자 인증서(공개 키 암호화)를 이용한 암호화를 지원합니다.
- PFX 파일, Windows 인증서 저장소 또는 임의의 외부 서명자를 이용한 전자 서명, PAdES(ETSI.CAdES.detached), 문서 타임스탬프, 장기 검증을 위한 DSS, 서명 검증, 그리고 선택적인 sgcSign 브리지를 통한 PAdES B-B부터 B-LTA까지를 지원합니다.
- 표시된 영역이나 검색 결과에서 텍스트, 이미지 픽셀, 그래픽, 주석을 실제로 제거하는 진정한 의미의 마스킹(redaction)을 지원합니다.
- 구조 트리 API를 갖춘 태그된 PDF, PDF/UA-1 및 PDF/A-1a, 2a, 3a 출력, PDF/A-1b, 2b, 3b 출력, PDF/A 및 PDF/UA 검사기를 지원합니다.
- 스트림을 압축하고, 중복되거나 사용하지 않는 객체를 제거하며, 이미지를 다운샘플링하는 옵티마이저를 제공합니다.
- 모든 플랫폼에서 순수 Pascal로 페이지를 PNG와 BMP로 렌더링하는 기능, GDI 렌더러, 확대/축소, 선택, 검색, 폼 채우기, 인쇄를 지원하는 VCL 및 FireMonkey용 뷰어 컨트롤을 제공합니다.
- .NET Standard 2.0, .NET Framework 4.7.2, .NET 6, .NET 8, .NET 9를 Windows, Linux, macOS에서 지원하며 Windows Forms 뷰어 컨트롤을 제공합니다.

요구 사항

- esegece.sgcPDF 패키지의 경우 .NET Standard 2.0(.NET Framework 4.6.2 이상, Mono, Xamarin), .NET Framework 4.7.2, .NET 6, .NET 8 또는 .NET 9.
- esegece.sgcPDF.WinForms 패키지(GDI 렌더러, 이미지로 내보내기, 뷰어 컨트롤)의 경우 Windows에서 .NET Framework 4.7.2 또는 .NET 8.
- FireMonkey 유닛은 .NET에서 사용할 수 없습니다.

컴포넌트

sgcPDF .NET은 네임스페이스 esegece.sgcPDF에 동일한 클래스를 제공합니다. Core 클래스는 esegece.sgcPDF 패키지에, VCL 클래스는 esegece.sgcPDF.WinForms에 있으며, TsgcPDFViewerFMX와 TsgcPDFBarcodeFMX는 사용할 수 없습니다:

컴포넌트	패키지	설명
TsgcPDFDocument	Core	PDF 문서: 페이지 생성, 콘텐츠 그리기, 로드, 저장, 암호화.
TsgcPDFEditor	Core	기존 PDF 파일 편집: 워터마크, 페이지 번호, 병합, 폼.
TsgcPDFTextExtractor	Core	폰트와 문자 위치를 포함한 페이지의 텍스트.
TsgcPDFOptimizer	Core	PDF 파일의 크기를 줄입니다.
TsgcPDFSignatureHandler	Core	PDF 문서에 서명합니다.
TsgcPDFSignatureVerifier	Core	PDF 문서의 서명을 검증합니다.
TsgcPDFRedactor	Core	페이지 영역에서 텍스트, 이미지, 그래픽을 제거합니다.
TsgcPDFImageExtractor	Core	PDF의 이미지 목록을 확인하고 저장합니다.
TsgcPDFHTMLRenderer	Core	HTML과 CSS를 PDF 페이지로 변환합니다.
TsgcPDFViewer	VCL	PDF를 표시, 검색, 인쇄하는 네이티브 창입니다.
TsgcPDFExporter	VCL	페이지를 BMP, PNG, JPEG 파일과 썸네일로 렌더링합니다.
TsgcPDFViewerControl	VCL	VCL 뷰어 컨트롤: 확대/축소, 선택, 검색, 폼 채우기, 인쇄.
TsgcPDFThumbnailView	VCL	TsgcPDFViewerControl용 페이지 썸네일입니다.
TsgcPDFOutlineView	VCL	TsgcPDFViewerControl용 북마크 트리입니다.
TsgcPDFBarcodeControl	VCL	바코드 및 QR Code 컨트롤: 11가지 심볼로지, 회전, PNG, SVG, PDF 출력.
TsgcPDFViewerFMX	FMX	Windows, macOS, iOS, Android, Linux용 FireMonkey 뷰어 컨트롤입니다.
TsgcPDFBarcodeFMX	FMX	FireMonkey 바코드 및 QR Code 컨트롤.

.NET 설치, .NET API 차이점, 그리고 빠른 시작부터 시작하십시오.

설치

sgcPDF .NET

sgcPDF .NET은 NuGet을 통해 설치합니다. 프로젝트에 esegece.sgcPDF 패키지를 추가하고, 뷰어 컨트롤을 사용하려면 esegece.sgcPDF.WinForms도 추가하십시오. .NET 설치를 참조하십시오.

아키텍처

sgcPDF는 기능별로 유닛이 나뉘어 있습니다. 실제로 사용하는 형식이 속한 유닛만 *uses*절에 추가하면 됩니다.
sgcPDF .NET에서는 모든 타입이 네임스페이스 esegece.sgcPDF에 있으며, 각 유닛의 전역 함수는 유닛 이름을 딴 클래스의 정적 멤버가 됩니다. 예를 들어 sgcPDF_Editor.sgcPDFMergeFiles 와 같습니다. WinForms 유닛은 esegece.sgcPDF.WinForms 패키지에 있으며 FMX 유닛은 사용할 수 없습니다. .NET API 차이점을 참조하십시오.

코어 유닛

유닛	내용
sgcPDF_Document	TsgcPDFDocument, TsgcPDFPage, TsgcPDFDocumentInfo
sgcPDF_Editor	TsgcPDFEditor, TsgcPDFWatermarkOptions, sgcPDFMergeFiles, sgcPDFExtractPages, sgcPDFGetPageCount
sgcPDF_Fonts	TsgcPDFStandardFont(14가지 표준 폰트), sgcPDFMeasureTextWidth
sgcPDF_Fonts_TrueType	TsgcTrueTypeFont: TrueType과 OpenType(CFF) 파싱, 서브셋, 임베드, 셰이핑
sgcPDF_Shaping	TsgcPDFShaper: OpenType GSUB/GPOS 셰이핑, 아랍어 연결, 유니코드 양방향 알고리즘
sgcPDF_Fonts_CFF, sgcPDF_Fonts_Type1	렌더러가 사용하는 CFF와 Type 1 폰트 프로그램
sgcPDF_Fonts_CJK	TsgcPDFCIDFont, TsgcPDFCJKFontManager(저수준, Windows 전용)
sgcPDF_ContentStream	TsgcPDFContentBuilder(페이지 그리기 연산자), TsgcPDFContentParser
sgcPDF_Graphics	TsgcPDFExtGState, TsgcPDFShading, TsgcPDFPattern, TsgcPDFGraphicsResources
sgcPDF_Canvas	TsgcPDFCanvas: PDF 페이지에서의 TCanvas 스타일 그리기
sgcPDF_HTML	TsgcPDFHTMLRenderer: HTML과 CSS를 PDF로
sgcPDF_Images	TsgcPDFImage: JPEG, PNG, BMP, TIFF 로드
sgcPDF_JPEG, sgcPDF_JPX, sgcPDF_CCITT, sgcPDF_JBIG2	이미지 디코더: JPEG, JPEG 2000, CCITT 팩스, JBIG2
sgcPDF_ColorSpaces	렌더러와 이미지 추출기가 공유하는 색 공간과 함수 평가
sgcPDF_ImageExtract	TsgcPDFImageExtractor, sgcPDFEncodePNG, sgcPDFEncodeBMP
sgcPDF_Tables	TsgcPDFTable, TsgcPDFTableRow, TsgcPDFTableCell
sgcPDF_Barcodes	TsgcPDFBarcodeGenerator, TsgcPDFBarcodeOptions: 선형 바코드, QR Code, DataMatrix, PDF417, GS1-128

유닛	내용
sgcPDF_Barcode_Symbol	TsgcPDFBarcodeSymbol: SVG, PNG, PDF 출력을 갖춘, 프레임워크에 의존하지 않는 바코드 레이아웃
sgcPDF_Annotations	TsgcPDFAnnotationManager와 주석 클래스
sgcPDF_Bookmarks	TsgcPDFBookmarks, TsgcPDFBookmarkItem
sgcPDF_Layers	TsgcPDFLayerManager, TsgcPDFLayer, TsgcPDFLayerGroup
sgcPDF_Forms	TsgcPDFAcroForm, TsgcPDFFormField
sgcPDF_Attachments	TsgcPDFAttachments: 임베드된 파일과 PDF/A-3 첨부 파일
sgcPDF_Tagged	태그된 PDF 구조 트리
sgcPDF_Redaction	TsgcPDFRedactor
sgcPDF_Encryption	TsgcPDFEncryptionMethod, TsgcPDFEncryptionHandler, 해시 및 암호 헬퍼
sgcPDF_Signatures	TsgcPDFSignatureHandler, TsgcPDFSignatureVerifier
sgcPDF_PDFA	TsgcPDFALevel, TsgcPDFAValidator, TsgcPDFAConverter, TsgcPDFXMPMetadata
sgcPDF_Optimize	TsgcPDFOptimizer, TsgcPDFOptimizeOptions, sgcPDFOptimizeFile
sgcPDF_TextExtract	TsgcPDFTextExtractor, sgcPDFExtractPageText, sgcPDFExtractAllText, sgcPDFFindText, sgcPDFCreatePageTextExtractor
sgcPDF_Renderer	TsgcPDFRenderer: 플랫폼 독립적인 콘텐츠 해석기
sgcPDF_Renderer_Canvas	TsgcPDFRenderCanvas: 추상 그리기 백엔드
sgcPDF_Renderer_Fonts	렌더러용 폰트 프로그램과 글리프 윤곽선
sgcPDF_Renderer_Raster	TsgcPDFRendererRaster, TsgcPDFRasterCanvas, sgcPDFRasterAddFontDirectory
sgcPDF_Parser	TsgcPDFParser: 저수준 PDF 리더
sgcPDF_Writer	TsgcPDFWriter: 저수준 PDF 라이터
sgcPDF_Filters	스트림 필터(Flate, LZW, ASCII85, ASCIIHex, RunLength, CCITT, JBIG2, 예측기)와 sgcPDFMaxDecodedSize
sgcPDF_Types	PDF 객체 모델(TsgcPDFDictionary, TsgcPDFArray, TsgcPDFStream...), TsgcPDFRect, TBytes

VCL 유닛(Windows)

유닛	내용
sgcPDF_Renderer_GDI	TsgcPDFRendererGDI, TsgcPDFGDICanvas: Windows 디바이스 컨텍스트에서의 렌더링

유닛	내용
sgcPDF_Export	TsgcPDFExporter, TsgcPDFExportOptions, sgcPDFPageToImage, sgcPDFToImages
sgcPDF_Viewer	TsgcPDFViewer
sgcPDF_ViewerVCL	TsgcPDFViewerControl, TsgcPDFThumbnailView, TsgcPDFOutlineView
sgcPDF_EMF	TsgcPDFEMFPlayer, sgcPDFEMFAddPage
sgcPDF_CanvasVCL	TsgcPDFVCLCanvasHelper
sgcPDF_Barcode_VCL	TsgcPDFBarcodeControl

FireMonkey 유닛(XE5 이상)

유닛	내용
sgcPDF_Renderer_FMX	TsgcPDFRendererFMX, TsgcPDFFMXCanvas, sgcPDFBGRAToFMXBitmap
sgcPDF_ViewerFMX	TsgcPDFViewerFMX
sgcPDF_Barcode_FMX	TsgcPDFBarcodeFMX, TsgcPDFBarcodeTextSettingsFMX

선택적 유닛인 sgcPDF_Sign_sgcSign(sgcSign을 통한 PAdES 서명)은 패키지에 포함되어 있지 않습니다. 사용할 때는 프로젝트에 직접 추가하십시오.

좌표와 단위

모든 위치와 크기는 PDF 포인트(1/72인치) 단위입니다. 원점은 페이지의 **왼쪽 아래** 모서리이며 Y 값은 위로 갈수록 증가합니다. A4 페이지는 595 x 842포인트이고, US Letter 페이지는 612 x 792포인트입니다. *Left, Bottom, Right, Top*으로 전달되는 사각형도 동일한 좌표 공간을 사용합니다. 색상은 0부터 1 사이의 R, G, B 구성 요소로 지정합니다. TsgcPDFCanvas는 예외로, VCL TCanvas처럼 왼쪽 위 원점과 선택한 단위를 사용합니다.

페이지 인덱스

페이지 인덱스는 0부터 시작합니다. 첫 페이지는 0이고 마지막 페이지는 `PageCount - 1`입니다.

문자열

표준 폰트로 그리는 메서드는 `AnsiString`(WinAnsi 인코딩)을 받습니다. TrueType 폰트로 그리는 메서드, 추출된 텍스트, 주석 내용, 북마크 제목, 폼 값은 `WideString`을 사용하므로, 폰트에 포함된 어떤 유니코드 문자든 지원합니다.

.NET 설치

sgcPDF .NET은 sgcPDF Delphi 라이브러리를 완전히 매니지드로 이식한 버전입니다. 동일한 클래스 이름(TsgcPDFDocument, TsgcPDFPage, TsgcPDFEditor 등)을 네임스페이스 `esegece.sgcPDF`에서 사용합니다. NuGet 패키지로 배포되며 네이티브 DLL이 필요하지 않습니다.

NuGet 패키지

패키지	내용	대상 프레임워크
esegece.sgcPDF	라이브러리 본체: 문서 모델, 파서, 라이터, 편집기, 렌더러 코어, 폰트, 이미지, 폼, 서명, 암호화, 텍스트 및 이미지 추출, PDF/A, 태그된 PDF, HTML을 PDF로 변환하는 기능, 래스터 렌더러(TsgcPDFRendererRaster).	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcPDF.WinForms	Windows Forms 애드온: GDI 렌더러와 캔버스, EMF 재생, 이미지로 페이지 내보내기, TsgcPDFViewerControl, TsgcPDFThumbnailView, TsgcPDFOutlineView 컨트롤.	net472, net8.0-windows
esegece.sgcPDF.sgcSign	sgcSign .NET에 대한 선택적 브리지: TsgcPDFPAdESSigner는 모든 sgcSign 키 제공자로 PAdES B-B, B-T, B-LT, B-LTA에 서명합니다. esegece.sgcSign에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0

필수 패키지는 **esegece.sgcPDF**뿐입니다. 화면 렌더링과 뷰어 컨트롤을 사용하려면 WinForms 패키지를, sgcSign을 통한 PAdES 서명을 사용하려면 sgcSign 패키지를 추가하십시오.
netstandard2.0 대상은 .NET Framework 4.6.2 이상, Mono, Xamarin에서 사용할 수 있습니다. 유일한 서드파티 종속성은 System.Security.Cryptography.Pkcs(서명 및 인증서 암호화용 CMS)이며, netstandard2.0과 net472에서는 System.Memory도 필요합니다.

패키지 설치

프로젝트 폴더에서 명령줄로 다음을 실행합니다:

```
dotnet add package esegece.sgcPDF
dotnet add package esegece.sgcPDF.WinForms
dotnet add package esegece.sgcPDF.sgcSign
```

Visual Studio에서는 프로젝트를 마우스 오른쪽 버튼으로 클릭하고 **NuGet 패키지 관리**를 선택한 다음 **esegece.sgcPDF**를 검색하여 필요한 패키지를 설치하십시오.

에디션

모든 에디션은 동일한 기능을 제공합니다. 차이는 기능이 아니라 라이선스에 있습니다.

에디션	패키지	라이선스
Community	esegece.sgcPDF.Community, esegece.sgcPDF.WinForms.Community, esegece.sgcPDF.sgcSign.Community	소규모 팀의 상업적, 비상업적 사용에 무료입니다. 연간 총매출이 100만 유로 미만이고 개발자가 5명 이하인 개인 및 조직이 해당됩니다. 짧고 차단되지 않는 시작 알림이 한 번 표시됩니다. 두 기준 중 하나라도 초과하는 조직은 유료 라이선스가 필요합니다. Community sgcSign 브리지는 esegece.sgcSign.Community에 종속됩니다.

에디션	패키지	라이선스
Registered	esegece.sgcPDF, esegece.sgcPDF.WinForms, esegece.sgcPDF.sgcSign	시작 알림이 제거되며 상업용 라이선스가 적용됩니다.

Community 에디션의 경우 동일한 패키지 ID에 **.Community** 접미사를 붙여 사용합니다. 예를 들어 `dotnet add package esegece.sgcPDF.Community`와 같습니다. 두 에디션 중 하나만 참조해야 합니다.
Registered 패키지는 LICENSE.txt의 라이선스로 배포되며, 설치하려면 이에 동의해야 합니다. Community 패키지는 LICENSE-Community.txt로 배포됩니다.

첫 번째 프로그램

PDF를 생성하여 저장한 다음 다시 로드하여 텍스트를 추출합니다:

```
using System;
using esegece.sgcPDF;

class Program
{
  static void Main()
  {
    // Create a one page A4 document and save it
    using (TsgcPDFDocument oDoc = new TsgcPDFDocument())
    {
      TsgcPDFPage oPage = oDoc.AddPage(595, 842);
      oPage.DrawText("Hello from sgcPDF", 50, 780, TsgcPDFStandardFont.sfHelveticaBold, 24);
      oPage.DrawLine(50, 770, 545, 770, 1.0);
      oPage.DrawText("Created with sgcPDF .NET.", 50, 740, TsgcPDFStandardFont.sfHelvetica, 12);
      oDoc.SaveToFile("hello.pdf");
    }

    // Load it again and extract the text of every page
    using (TsgcPDFDocument oDoc = new TsgcPDFDocument())
    {
      oDoc.LoadFromFile("hello.pdf");
      for (int i = 0; i < oDoc.PageCount(); i++)
        Console.WriteLine(oDoc.ExtractTextW(i));
    }
  }
}
```

참고

- .NET API 차이점
- .NET 데모
- 빠른 시작

.NET API 차이점

sgcPDF .NET은 Delphi API를 그대로 유지합니다. 타입, 열거형 멤버, 메서드, 속성, 매개변수, 상수는 Delphi와 동일한 이름을 사용하므로 이 도움말의 항목이 그대로 .NET에도 적용됩니다. 항목에 Delphi 코드만 표시된 경우 아래 규칙에 따라 C#으로 변환합니다.

네임스페이스와 유닛

- 모든 타입은 네임스페이스 `esegece.sgcPDF`에 있습니다. Delphi에서 *uses* 절에 유닛을 추가하는 대신 `using esegece.sgcPDF;`를 추가합니다.
- Delphi 유닛의 전역 함수, 상수, 변수는 유닛 이름을 딴 정적 클래스의 정적 멤버가 됩니다. 예를 들어 sgcPDF_Editor 유닛의 sgcPDFMergeFiles는 `sgcPDF_Editor.sgcPDFMergeFiles(...)`가 됩니다. `using static esegece.sgcPDF.sgcPDF_Editor;`를 사용하면 Delphi에서처럼 클래스 이름 없이 호출할 수 있습니다.
- 열거형 멤버는 열거형 타입으로 한정됩니다: `sfHelvetica` 대신 `TsgcPDFStandardFont.sfHelvetica`를 사용합니다.

메서드, 속성, 인덱스

- 매개변수가 없는 Delphi 함수도 여전히 메서드이므로 C#에서는 괄호를 붙여 호출합니다: `oPDF.PageCount()`.
- 인덱스 속성은 메서드로 노출됩니다. TsgcPDFDocument의 `Pages[i]`는 `GetPage(i)`가 되며, C# 인덱서 `oPDF[i]`로도 사용할 수 있습니다.
- 레코드는 구조체가 됩니다. 예를 들어 TsgcPDFRect가 그렇습니다.
- 집합(set)은 [Flags] 특성이 붙은 열거형이 됩니다. Delphi 집합 `[ppPrint, ppCopy]`는 `TsgcPDFPermissions.ppPrint | TsgcPDFPermissions.ppCopy`가 되고, `ppPrint in APerms`는 `(APerms & TsgcPDFPermissions.ppPrint) != 0`이 됩니다.
- TBytes는 `byte[]`이며, `var` 매개변수는 `ref` 매개변수가 됩니다.
- 기본값이 있는 매개변수는 기본값을 그대로 유지하며, 오버로드도 Delphi 시그니처를 그대로 유지합니다.

이벤트

이벤트는 Delphi 이벤트 타입 이름을 가진 델리게이트 타입의 속성입니다. 동일한 매개변수를 가진 메서드를 할당합니다. 첫 번째 매개변수는 sender이며, `var` 매개변수는 `ref`로 전달됩니다:

```
oHandler.OnExternalSign = DoExternalSign;

void DoExternalSign(object Sender, byte[] AData, byte[] ADigest, ref byte[] ASignature)
{
  ASignature = MySigner.Sign(ADigest);
}
```

객체 생성 및 해제

- `TsgcPDFDocument.Create`는 `new TsgcPDFDocument()`가 됩니다.
- 컴포넌트는 System.ComponentModel.Component에서 상속하며 IDisposable을 구현합니다. Delphi에서 `try..finally` 블록으로 `Free`를 호출하는 부분은 `using` 블록을 사용하거나 `Dispose`를 호출하십시오. Delphi 코드가 메모리 해제만을 위해 해제하는 객체는 가비지 컬렉터에 맡깁니다.
- 예외는 이름을 그대로 유지합니다: `on E: EsgcPDFError do`는 `catch (EsgcPDFError E)`가 됩니다.

플랫폼

- **esegece.sgcPDF** 패키지는 Windows, Linux, macOS를 포함하여 .NET이 실행되는 모든 곳에서 동작합니다. TsgcPDFRendererRaster로 PNG나 BMP로 렌더링하는 기능은 모든 플랫폼에서 동작합니다.
- **esegece.sgcPDF.WinForms** 패키지는 Windows 전용입니다. Delphi에서 Windows GDI를 사용하는 유닛인 TsgcPDFRendererGDI, TsgcPDFViewer, TsgcPDFExporter, TsgcPDFEMFPlayer, TsgcPDFVCLCanvasHelper와 뷰어 컨트롤인 TsgcPDFViewerControl, TsgcPDFThumbnailView, TsgcPDFOutlineView, 바코드 컨트롤인 TsgcPDFBarcodeControl을 포함합니다. 이 컨트롤들은 Windows Forms 컨트롤이므로 다른 컨트롤과 마찬가지로 `Parent`와 `Dock`을 설정합니다.

- 서명과 인증서 암호화는 Windows CryptoAPI 대신 System.Security.Cryptography를 사용합니다. Windows 인증서 저장소로 서명하려면 Windows가 필요합니다.
- 이름으로 설치된 TrueType 및 CJK 폰트를 찾는 기능은 Windows 폰트 폴더를 사용합니다. 다른 플랫폼에서는 AddTrueTypeFontFromFile로 파일에서 폰트를 로드하십시오.

.NET에서 사용할 수 없는 기능

- FireMonkey 유닛인 TsgcPDFViewerFMX, TsgcPDFBarcodeFMX와 TsgcPDFRendererFMX. FireMonkey, iOS, Android 관련 항목은 Delphi에만 적용됩니다.
- Delphi 패키지의 디자인 타임 등록. 컴포넌트는 코드로 생성하십시오.

빠른 참조

Delphi	C#
uses sgcPDF_Document, sgcPDF_Fonts;	using esegece.sgcPDF;
oPDF := TsgcPDFDocument.Create;	TsgcPDFDocument oPDF = new TsgcPDFDocument();
oPDF.Free;	oPDF.Dispose(); or a using block
oPDF.PageCount	oPDF.PageCount()
oPDF.Pages[i]	oPDF.GetPage(i) or oPDF[i]
sfHelvetica	TsgcPDFStandardFont.sfHelvetica
[ppPrint, ppCopy]	TsgcPDFPermissions.ppPrint | TsgcPDFPermissions.ppCopy
sgcPDFMergeFiles(...)	sgcPDF_Editor.sgcPDFMergeFiles(...)
TBytes	byte[]
oHandler.OnExternalSign := DoExternalSign;	oHandler.OnExternalSign = DoExternalSign;

참고

- .NET 설치
- .NET 데모
- 아키텍처

.NET 데모

sgcPDF .NET의 **demos** 폴더에는 net8.0-windows를 대상으로 하는 26개의 Windows Forms 데모가 들어 있습니다. 각 데모는 동일한 이름의 Delphi 데모를 그대로 반영합니다. Visual Studio에서 데모의 .csproj 파일을 열거나 `dotnet build`로 빌드하십시오. 소스 트리에서는 **sgcUI.sln**에 라이브러리와 모든 데모가 들어 있습니다.

데모	내용
PDFCreator	제목, 선, 본문 텍스트, 테두리가 있는 A4 한 페이지 문서.
PDFAnnotations	스티커 노트, 링크, 자유 텍스트, 하이라이트, 스탬프 등의 주석.
PDFAttachments	첨부 파일 추가, 목록, 추출, 삭제, Factur-X 인보이스.
PDFBarcodeControl	모든 심볼로지를 위한 TsgcPDFBarcodeControl. 회전, 맞춤, 색을 지정하여 PNG, SVG, PDF로 저장합니다. Delphi의 PDFBarcodeControls 데모에 대응합니다.
PDFBarcodes2D	QR Code, DataMatrix 및 기타 2D 바코드.
PDFBookmarks	중첩된 북마크(개요) 트리.
PDFCanvas	TsgcPDFCanvas와 같은 GDI를 이용한 그리기와 EMF를 벡터 PDF로 변환.
PDFCertEncryption	하나 이상의 수신자 인증서를 이용한 암호화.
PDFEditor	워터마크, 페이지 번호, 기존 PDF의 페이지 추출.
PDFEncryption	RC4-128, AES-128, AES-256을 이용한 암호 보호.
PDFFormData	AcroForm을 채운 다음 데이터를 내보내고 가져오기.
PDFForms	텍스트 필드, 콤보 박스, 체크 박스, 버튼이 있는 AcroForm 생성.
PDFImageExtract	PDF의 이미지 목록을 가져와 저장.
PDFImages	페이지에 맞게 크기가 조정된 JPEG, PNG, BMP 이미지.
PDFIncremental	증분 업데이트로 저장된 변경 사항.
PDFLayers	선택적 콘텐츠 그룹(레이어).
PDFPDFA	임베디드 TrueType 폰트를 사용한 PDF/A-1b, 2b, 3b.
PDFRedaction	텍스트와 영역의 실제 편집(레댁션).
PDFSignature	Windows 저장소 인증서로 서명하고 서명 검증.
PDFSignaturePAdES	PFX 파일 또는 외부 서명자를 이용한 CAdES 및 PKCS#7 서명.
PDFTagged	태그된 PDF(PDF/UA-1, PDF/A-2a).
PDFTextExtract	텍스트 추출 및 검색.
PDFTransparency	그라디언트와 투명도.
PDFUnicode	임베디드 TrueType 폰트를 사용한 유니코드 텍스트.
PDFViewer	GDI 렌더러로 현재 페이지 렌더링.

데모	내용
PDFViewerControl	썸네일과 북마크가 있는 TsgcPDFViewerControl.

Delphi 에디션의 FireMonkey 데모는 이식되지 않았습니다. .NET 에디션에는 FireMonkey 유닛이 없기 때문입니다.

프로젝트 참조 또는 NuGet 패키지

소스 트리 내부에서 데모는 라이브러리 프로젝트를 참조합니다. **source** 폴더가 없는, 데모만 포함된 배포판에서는 **demos\Directory.Build.props**와 **demos\Directory.Build.targets**가 sgcPDF NuGet 패키지로 전환합니다. 패키지 모드를 강제하려면 `-p:SgcPDFUseProject=false`를 지정하십시오.
패키지의 에디션은 한 곳, 즉 demos\Directory.Build.props의 **SgcPDFCommunity** 속성에서 선택됩니다. 데모는 기본적으로 Community 패키지를 사용합니다. Registered 패키지로 빌드하려면 `-p:SgcPDFCommunity=false`를 지정하십시오.

참고

- .NET 설치
- .NET API 차이점

빠른 시작

다음 예제는 콘솔 프로그램의 완전한 메서드입니다. `using esegece.sgcPDF;`와 esegece.sgcPDF NuGet 패키지를 추가하고, 렌더링 및 뷰어 예제에는 esegece.sgcPDF.WinForms도 추가하십시오. 각 섹션에 나열된 유닛은 Delphi 유닛입니다. .NET API 차이점을 참조하십시오.

PDF 생성하기

유닛: sgcPDF_Fonts, sgcPDF_Document.

```
static void CreateHelloPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.Info.Title = "Hello sgcPDF";
    oPDF.Info.Author = "eSeGeCe";
    // A4 portrait: 595 x 842 points, origin at the bottom left corner
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    oPage.DrawTextColor("Hello sgcPDF", 50, 780, TsgcPDFStandardFont.sfHelveticaBold, 24, 0, 0.2, 0.6);
    oPage.DrawLine(50, 770, 545, 770, 1.5);
    oPage.DrawText("A PDF written in pure Pascal.", 50, 740, TsgcPDFStandardFont.sfHelvetica, 12);
    oPage.DrawRectangle(40, 40, 515, 762, 1.0);
    oPDF.SaveToFile(AFileName);
  }
}
```

PDF 열고 편집하기

유닛: sgcPDF_Fonts, sgcPDF_Editor. TsgcPDFEditor는 문서를 로드하고 모든 페이지에 변경 사항을 적용한 다음 저장합니다.

```
static void StampExistingPDF(string AInput, string AOutput)
{
  using (TsgcPDFEditor oEditor = new TsgcPDFEditor())
  {
    oEditor.LoadFromFile(AInput);
    TsgcPDFWatermarkOptions oWatermark = new TsgcPDFWatermarkOptions();
    oWatermark.Text = "CONFIDENTIAL";
    oWatermark.Font = TsgcPDFStandardFont.sfHelveticaBold;
    oWatermark.FontSize = 48;
    oWatermark.Rotation = 45;
    oWatermark.Opacity = 0.3;
    oWatermark.R = 0.5;
    oWatermark.G = 0.5;
    oWatermark.B = 0.5;
    oEditor.AddWatermarkAllPages(oWatermark);
    oEditor.AddHeaderText("Internal report", TsgcPDFStandardFont.sfHelvetica, 9);
    oEditor.AddPageNumbers(TsgcPDFStandardFont.sfHelvetica, 10, "Page %d of %d");
    oEditor.AddText(0, "Reviewed", 450, 60, TsgcPDFStandardFont.sfHelveticaOblique, 11);
    oEditor.SaveToFile(AOutput);
  }
}
```

텍스트 추출하기

유닛: sgcPDF_Document.

```
static void PrintAllText(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    for (int i = 0; i < oPDF.PageCount(); i++)
    {
      Console.WriteLine("--- page " + (i + 1) + " ---");
      Console.WriteLine(oPDF.ExtractTextW(i));
    }
    if (oPDF.FindText(0, "Hello"))
      Console.WriteLine("Page 1 contains \"Hello\"");
  }
}
```

페이지를 이미지로 렌더링하기

유닛: sgcPDF_Document, sgcPDF_Export.

```
static void ExportPages(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  using (TsgcPDFExporter oExporter = new TsgcPDFExporter())
  {
    oPDF.LoadFromFile(AFileName);
    TsgcPDFExportOptions oOptions = sgcPDF_Export.sgcPDFDefaultExportOptions();
    oOptions.Format = TsgcPDFExportFormat.efPNG;
    oOptions.DPI = 150;
    oExporter.ExportPage(oPDF, 0, @"out\page1.png", oOptions);
    oExporter.ExportThumbnail(oPDF, 0, @"out\thumb1.bmp", 200, 200);
  }
  // one call helpers that open the file themselves
  sgcPDF_Export.sgcPDFPageToImage(AFileName, @"out\page1_96.png", 0, 96);
}
```

PDF 보기

유닛: sgcPDF_Viewer. TsgcPDFViewer는 네이티브 창을 엽니다. 콘솔 프로그램에서는 `RunMessageLoop`이 창이 닫힐 때까지 유지합니다.

```
static void ShowViewer(string AFileName)
{
  using (TsgcPDFViewer oViewer = new TsgcPDFViewer())
  {
    oViewer.ViewMode = TsgcPDFViewerMode.vmContinuous;
    // top level window; use CreateWindow(AParent, X, Y, W, H) to embed it
    oViewer.CreateWindow(100, 100, 800, 900, "My PDF Viewer");
    oViewer.LoadFromFile(AFileName);
    oViewer.ZoomFit();
    oViewer.Show();
    oViewer.RunMessageLoop();
  }
}
```

참고 항목

- 텍스트와 폰트
- 렌더링과 내보내기

TsgcPDFDocument

유닛: sgcPDF_Document
sgcPDF의 핵심 클래스입니다. 문서의 페이지를 보관하고, 새 콘텐츠를 그리며, 기존 파일을 로드하고, 필요하면 암호화하거나 PDF/A로 결과를 저장합니다. 문서의 북마크, 주석, 폼 필드, 레이어에도 접근할 수 있습니다.

생성 및 로드

메서드	설명
Create / Create(AOwner)	빈 문서를 생성합니다.
LoadFromFile(const AFileName: string)	PDF 파일을 로드합니다. 암호화된 파일은 먼저 SetPassword를 호출하십시오.
LoadFromStream(AStream: TStream)	스트림에서 PDF를 로드합니다.
LoadFromBytes(const AData: TBytes)	메모리에서 PDF를 로드합니다.
SaveToFile(const AFileName: string)	문서를 파일에 씁니다.
SaveToStream(AStream: TStream)	문서를 스트림에 씁니다.

페이지

멤버	설명
AddPage: TsgcPDFPage	A4 페이지(595 x 842 포인트)를 추가합니다.
AddPage(AWidth, AHeight: Double): TsgcPDFPage	지정한 크기의 페이지를 추가합니다.
InsertPage(AIndex: Integer; AWidth, AHeight: Double): TsgcPDFPage	지정한 위치에 페이지를 삽입합니다.
DeletePage(AIndex: Integer)	페이지를 제거합니다.
GetPage(AIndex: Integer) / Pages[AIndex]	페이지를 반환합니다(기본 속성).
PageCount: Integer	페이지 수입니다.
AppendDocument(ADoc: TsgcPDFDocument)	다른 문서의 페이지를 추가합니다.

리소스

메서드	설명
AddTrueTypeFont(const AFontName: WideString; ABold, AItalic: Boolean): Integer	설치된 TrueType 폰트를 로드하고 TsgcPDFPage.DrawTextTT에 사용할 폰트 인덱스를 반환합니다.
AddImage(const AFileName: string): Integer	JPEG, PNG 또는 BMP 파일을 로드하고 이미지 인덱스를 반환합니다.
AddImageFromStream(AStream: TStream): Integer	AddImage와 동일하나 스트림에서 로드합니다.

텍스트

메서드	설명
ExtractText(APageIndex: Integer): AnsiString	페이지의 텍스트를 AnsiString으로 반환합니다.
ExtractTextW(APageIndex: Integer): WideString	페이지의 텍스트를 유니코드로 반환합니다.
FindText(APageIndex: Integer; const AText: WideString): Boolean	페이지에 해당 텍스트가 있으면 True를 반환합니다.
GetPageContentStream(APageIndex: Integer): TBytes	페이지의 디코딩된 콘텐츠 스트림입니다.

보안 및 규격 준수

메서드	설명
SetEncryption(AMethod: TsgcPDFEncryptionMethod; const AUserPassword, AOwnerPassword: AnsiString)	문서를 저장할 때 암호화합니다.
SetPassword(const APassword: AnsiString)	암호화된 파일을 열 때 사용하는 암호입니다. LoadFromFile 전에 호출하십시오.
SetPDFALevel(ALevel: TsgcPDFALevel)	문서를 PDF/A-1b, PDF/A-2b 또는 PDF/A-3b로 저장합니다.

속성

속성	형식	설명
Info	TsgcPDFDocumentInfo	Title, Author, Subject, Keywords, Creator 및 Producer.
Bookmarks	TsgcPDFBookmarks	문서 개요입니다. 북마크를 참조하십시오.
Annotations	TsgcPDFAnnotationManager	모든 페이지의 주석입니다. 주석을 참조하십시오.
AcroForm	TsgcPDFAcroForm	대화형 폼 필드입니다. 폼을 참조하십시오.
Layers	TsgcPDFLayerManager	선택적 콘텐츠 그룹입니다. 레이어를 참조하십시오.

속성	형식	설명
GraphicsResources	TsgcPDFGraphicsResources	투명도 상태, 그라디언트, 패턴입니다.
IsLoaded	Boolean	문서를 로드한 후에는 True입니다.
Parser	TsgcPDFParser	로드된 문서의 파서로, 텍스트 추출기, 검사기, 렌더러가 사용합니다.
PDFALevel	TsgcPDFALevel	SetPDFALevel로 설정한 PDF/A 레벨입니다.
Version (published)	AnsiString	파일 헤더에 기록되는 PDF 버전입니다.

TsgcPDFPage

AddPage와 GetPage가 반환합니다. 이 클래스의 그리기 메서드는 페이지 콘텐츠에 연산자를 추가합니다.

메서드	설명
DrawText(const AText: AnsiString; X, Y: Double; AFont: TsgcPDFStandardFont; AFontSize: Double)	베이스라인 지점 X, Y에서 시작하는 표준 폰트 텍스트입니다.
DrawTextColor(..., R, G, B: Double)	위와 동일하나 텍스트 색상을 지정합니다.
DrawTextTT(const AText: WideString; X, Y: Double; AFontIndex: Integer; AFontSize: Double)	AddTrueTypeFont로 추가한 TrueType 폰트를 사용하는 유니코드 텍스트입니다.
DrawTextTTColor(..., R, G, B: Double)	위와 동일하나 텍스트 색상을 지정합니다.
DrawLine(X1, Y1, X2, Y2: Double; ALineWidth: Double = 1.0)	선입니다. DrawLineColor는 색상을 추가합니다.
DrawRectangle(X, Y, W, H: Double; ALineWidth: Double = 1.0)	사각형 윤곽선입니다. DrawRectangleFill은 색상으로 채웁니다.
DrawCircle(CX, CY, Radius: Double; ALineWidth: Double = 1.0)	원 윤곽선입니다. DrawCircleFill은 색상으로 채웁니다.
DrawImageFromIndex(AImageIndex: Integer; X, Y, W, H: Double)	AddImage로 추가한 이미지를 그립니다.
DrawImageFile(const AFileName: string; X, Y, W, H: Double)	한 번의 호출로 이미지를 로드하고 그립니다.
SetOpacity(AOpacity: Double)	이후 연산에 적용할 불투명도(0에서 1까지)입니다.
DrawRectangleTransparent(X, Y, W, H: Double; R, G, B, AOpacity: Double)	반투명하게 채워진 사각형입니다.
DrawLinearGradient(X, Y, W, H: Double; R0, G0, B0, R1, G1, B1: Double)	선형 그라디언트로 채워진 사각형입니다.

페이지 속성: **Width**, **Height**, **MediaBox**, **PageIndex**, **PageDict**, 그리고 모든 그리기 연산자에 접근할 수 있는 TsgcPDFContentBuilder인 **Content**입니다(그래픽 참조).

이벤트

TsgcPDFDocument에는 이벤트가 없습니다.

예제

```
static void CreateHelloPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.Info.Title = "Hello sgcPDF";
    oPDF.Info.Author = "eSeGeCe";
    // A4 portrait: 595 x 842 points, origin at the bottom left corner
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    oPage.DrawTextColor("Hello sgcPDF", 50, 780, TsgcPDFStandardFont.sfHelveticaBold, 24, 0, 0.2, 0.6);
    oPage.DrawLine(50, 770, 545, 770, 1.5);
    oPage.DrawText("A PDF written in pure Pascal.", 50, 740, TsgcPDFStandardFont.sfHelvetica, 12);
    oPage.DrawRectangle(40, 40, 515, 762, 1.0);
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- 텍스트와 폰트
- 페이지 병합, 분할 및 추출

TsgcPDFEditor

유닛: sgcPDF_Editor
기존 PDF에 대한 고수준 편집 기능입니다. 에디터는 TsgcPDFDocument(속성 **Document**)를 감싸고, 워터마크, 머리글, 바닥글, 페이지 번호, 병합, 페이지 추출, 폼 채우기, 평면화 등을 한 번의 호출로 수행하는 기능을 추가합니다.

생성자

생성자	설명
Create / Create(AOwner: TComponent)	자체 문서를 가진 에디터를 생성합니다.
Create(ADocument: TsgcPDFDocument)	기존 TsgcPDFDocument 인스턴스를 편집합니다.

메서드

메서드	설명
LoadFromFile / SaveToFile	문서를 로드하고 저장합니다.
AddBlankPage / AddBlankPage(AWidth, AHeight: Double)	빈 페이지를 추가합니다.
DeletePage(AIndex: Integer)	페이지를 제거합니다.
RotatePage(AIndex: Integer; ADegrees: Integer)	페이지를 회전합니다. ADegrees는 90, 180 또는 270이어야 합니다.
AddText / AddTextColor(APageIndex, AText, X, Y, AFont, AFontSize [, R, G, B])	페이지에 텍스트를 그립니다.
AddTransparentText(APageIndex, AText, X, Y, AFont, AFontSize, AOpacity)	반투명 텍스트를 그립니다.
AddLine / AddRectangle / AddRectangleFill	페이지에 도형을 그립니다.
AddImageToPage(APageIndex: Integer; const AImageFile: string; X, Y, W, H: Double)	페이지에 이미지 파일을 그립니다.
AddWatermark(APageIndex; const AOptions: TsgcPDFWatermarkOptions)	한 페이지에 회전되고 반투명한 텍스트를 표시합니다.
AddWatermarkAllPages(const AOptions: TsgcPDFWatermarkOptions)	모든 페이지에 동일하게 적용합니다.
AddHeaderText / AddFooterText(const AText: AnsiString; AFont; AFontSize)	모든 페이지의 위쪽이나 아래쪽에 표시되는 텍스트입니다.
AddPageNumbers(AFont; AFontSize: Double; const AFormat: AnsiString)	모든 페이지에 페이지 번호를 표시합니다. AFormat에는 현재 페이지와 전체 페이지 수를 나타내는 두 개의 %d가 필요합니다.
AddTextAnnotation / AddLinkAnnotation / AddHighlightAnnotation	주석을 추가합니다.

메서드	설명
AddBookmark(const ATitle: WideString; APageIndex: Integer)	최상위 북마크를 추가합니다. AddBookmarkChild는 하위 북마크를 추가합니다.
MergeFrom(const AFileName: string)	다른 PDF의 모든 페이지를 추가합니다.
ExtractPages(AFromIndex, AToIndex: Integer; const AOutputFileName: string)	페이지 범위를 새 파일로 저장합니다.
ExtractText(APageIndex): AnsiString / ExtractTextUnicode(APageIndex): WideString / ExtractAllText	텍스트 추출입니다.
SearchText(APageIndex: Integer; const AText: WideString): Boolean	페이지에 해당 텍스트가 있으면 True입니다.
FillFormField(const AFieldName: AnsiString; const AValue: WideString)	폼 필드 값을 설정합니다.
FlattenForm	폼 필드를 정적 페이지 콘텐츠로 변환합니다.
SetEncryption(AMethod; const AUserPassword, AOwnerPassword: AnsiString)	출력을 암호화합니다.
ConvertToPDFA(ALevel: TsgcPDFALevel)	출력을 PDF/A로 저장합니다.
SignDocument(const AOutputFile: string; const ACertSubjectName, AReason, ALocation: WideString)	주체 이름으로 찾은 Windows 저장소의 인증서로 서명하여 저장합니다.

TsgcPDFWatermarkOptions

Text, Font, FontSize, Rotation(도), Opacity(0에서 1까지), R, G, B 필드를 갖는 레코드입니다.

헬퍼 함수

함수	설명
sgcPDFMergeFiles(const AInputFiles: array of string; const AOutputFile: string)	여러 파일을 병합합니다.
sgcPDFExtractPages(const AInputFile, AOutputFile: string; AFromPage, AToPage: Integer)	0부터 시작하는 페이지 범위를 추출합니다.
sgcPDFGetPageCount(const AFileName: string): Integer	파일의 페이지 수입니다.

이벤트

TsgcPDFEditor에는 이벤트가 없습니다.

예제

```
static void StampExistingPDF(string AInput, string AOutput)
{
  using (TsgcPDFEditor oEditor = new TsgcPDFEditor())
  {
```

```
    oEditor.LoadFromFile(AInput);
    TsgcPDFWatermarkOptions oWatermark = new TsgcPDFWatermarkOptions();
    oWatermark.Text = "CONFIDENTIAL";
    oWatermark.Font = TsgcPDFStandardFont.sfHelveticaBold;
    oWatermark.FontSize = 48;
    oWatermark.Rotation = 45;
    oWatermark.Opacity = 0.3;
    oWatermark.R = 0.5;
    oWatermark.G = 0.5;
    oWatermark.B = 0.5;
    oEditor.AddWatermarkAllPages(oWatermark);
    oEditor.AddHeaderText("Internal report", TsgcPDFStandardFont.sfHelvetica, 9);
    oEditor.AddPageNumbers(TsgcPDFStandardFont.sfHelvetica, 10, "Page %d of %d");
    oEditor.AddText(0, "Reviewed", 450, 60, TsgcPDFStandardFont.sfHelveticaOblique, 11);
    oEditor.SaveToFile(AOutput);
  }
}
```

참고 항목

- 페이지 병합, 분할 및 추출
- 폼

TsgcPDFViewer

유닛: sgcPDF_Viewer
네이티브 Windows 창으로 구현된 PDF 뷰어입니다. VCL에 의존하지 않으므로 VCL 애플리케이션에서(Handle을 통해 임의의 창 컨트롤에 임베드하여) 사용할 수 있고, 콘솔이나 비시각적 프로그램에서도(최상위 창으로) 사용할 수 있습니다. 페이지는 GDI 렌더러로 그려집니다.

창

메서드	설명
CreateWindow(AParent: HWND; X, Y, W, H: Integer)	예를 들어 TPanel.Handle 등의 자식 창으로 뷰어를 생성합니다.
CreateWindow(X, Y, W, H: Integer; const ATitle: AnsiString)	최상위 창을 생성합니다.
Show / Hide / Invalidate / DestroyWindow	창 관리입니다.
RunMessageLoop	창이 닫힐 때까지 메시지 루프를 실행합니다. 콘솔 프로그램처럼 자체 메시지 루프가 없는 프로그램에서만 사용하십시오.

문서와 탐색

메서드	설명
LoadFromFile(const AFileName: string)	PDF 파일을 엽니다.
LoadDocument(ADocument: TsgcPDFDocument; AOwns: Boolean = False)	이미 메모리에 있는 문서를 표시합니다.
CloseDocument	문서를 닫습니다.
FirstPage / LastPage / NextPage / PrevPage / GotoPage(AIndex)	탐색입니다.
ZoomIn / ZoomOut / ZoomFit / ZoomFitPage / SetZoom(APercent: Double)	확대/축소입니다.
FindText(const AText: WideString): Boolean / FindNext / ClearSearch	텍스트 검색입니다.
Print / PrintPage(APageIndex: Integer)	인쇄입니다.
GetPageText(APageIndex: Integer): WideString	페이지의 텍스트입니다.
ExportPageToBitmap(APageIndex: Integer; const AFileName: string; AWidth, AHeight: Integer)	페이지를 BMP 파일로 렌더링합니다.

창은 키보드에도 반응합니다. Left와 Page Up은 이전 페이지로, Right와 Page Down은 다음 페이지로 이동하고, Home과 End는 첫 페이지와 마지막 페이지로 이동하며, + 및 - 키는 확대/축소하고, Up과 Down은 스크롤합니다.

속성

속성	형식	설명
ViewMode (published)	TsgcPDFViewerMode	vmSinglePage(기본값) 또는 vmContinuous.
BackColor (published)	COLORREF	페이지 주위의 배경색입니다. 기본값은 $00C8C8C8입니다.
ShowAnnotations (published)	Boolean	주석을 그립니다. 기본값은 True입니다.
Handle	HWND	창 핸들입니다.
CurrentPage	Integer	현재 페이지의 0 기반 인덱스입니다.
PageCount	Integer	페이지 수입니다.
Zoom	Double	현재 확대/축소 배율입니다.
Document	TsgcPDFDocument	표시 중인 문서입니다.

이벤트

TsgcPDFViewer에는 이벤트가 없습니다.

예제

```
static void ShowViewer(string AFileName)
{
  using (TsgcPDFViewer oViewer = new TsgcPDFViewer())
  {
    oViewer.ViewMode = TsgcPDFViewerMode.vmContinuous;
    // top level window; use CreateWindow(AParent, X, Y, W, H) to embed it
    oViewer.CreateWindow(100, 100, 800, 900, "My PDF Viewer");
    oViewer.LoadFromFile(AFileName);
    oViewer.ZoomFit();
    oViewer.Show();
    oViewer.RunMessageLoop();
  }
}
```

참고 항목

- 렌더링과 내보내기

TsgcPDFTextExtractor

유닛: sgcPDF_TextExtract
한 페이지의 텍스트를 각 텍스트 조각의 위치, 폰트 이름, 폰트 크기, 너비를 유지하며 추출합니다. 표준 인코딩, 폰트 Differences, ToUnicode 맵, CID 폰트를 디코딩하므로 TrueType과 CJK 폰트로 작성된 텍스트도 유니코드로 반환됩니다.

생성자

생성자	설명
Create(AParser: TsgcPDFParser; APageDict: TsgcPDFDictionary)	한 페이지의 추출을 준비합니다. 로드된 문서의 TsgcPDFDocument.Parser와 TsgcPDFPage.PageDict를 전달하십시오.
Create(AOwner: TComponent)	IDE에서 사용하는 컴포넌트 생성자입니다. 코드에서 텍스트를 추출하려면 위의 생성자를 사용하십시오.

메서드

메서드	설명
Extract	페이지 콘텐츠를 파싱합니다. 다른 메서드보다 먼저 호출하십시오.
GetText: WideString	읽기 순서대로 정렬된 페이지 텍스트이며, 텍스트 줄마다 한 줄입니다.
GetTextWithPositions: WideString	모든 조각의 좌표가 포함된 텍스트입니다.
GetFragmentCount: Integer	조각 수입니다.
GetFragment(AIndex: Integer): TsgcPDFTextFragment	조각입니다: Text, X, Y, FontSize, FontName, Width.
FindText(const ASearchText: WideString; ACaseSensitive: Boolean = False): Boolean	페이지 텍스트를 검색합니다.

헬퍼 함수

함수	설명
sgcPDFExtractPageText(AParser: TsgcPDFParser; APageIndex: Integer): WideString	한 페이지의 텍스트입니다.
sgcPDFExtractAllText(AParser: TsgcPDFParser): WideString	모든 페이지의 텍스트입니다.
sgcPDFFindText(AParser: TsgcPDFParser; APageIndex: Integer; const AText: WideString): Boolean	한 페이지를 검색합니다.

TsgcPDFParser를 직접 사용하는 경우 로드한 다음 파싱해야 합니다. `LoadFromFile`을 호출한 다음 `Parse`를 호출하십시오.

이벤트

TsgcPDFTextExtractor에는 이벤트가 없습니다.

예제

```
static void PrintTextPositions(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    using (TsgcPDFTextExtractor oExtractor = new TsgcPDFTextExtractor(oPDF.Parser,
      oPDF.GetPage(0).PageDict))
    {
      oExtractor.Extract();
      for (int i = 0; i < oExtractor.GetFragmentCount(); i++)
      {
        TsgcPDFTextFragment oFragment = oExtractor.GetFragment(i);
        Console.WriteLine(string.Format("{0:0.0}, {1:0.0}  {2}  ({3} {4:0.0} pt)", oFragment.X,
          oFragment.Y, oFragment.Text, oFragment.FontName, oFragment.FontSize));
      }
    }
  }
}
```

참고 항목

- 텍스트 추출

TsgcPDFExporter

유닛: sgcPDF_Export
TsgcPDFDocument의 페이지를 GDI 렌더러(TsgcPDFRendererGDI)를 사용하여 BMP, PNG 또는 JPEG 이미지로 렌더링합니다.

메서드

메서드	설명
ExportPage(ADocument: TsgcPDFDocument; APageIndex: Integer; const AFileName: string)	Options 속성을 사용하여 페이지를 렌더링합니다.
ExportPage(ADocument; APageIndex; const AFileName: string; const AOptions: TsgcPDFExportOptions)	명시적인 옵션으로 페이지를 렌더링합니다.
ExportPageToStream(ADocument; APageIndex; AStream: TStream; const AOptions)	페이지를 스트림으로 렌더링합니다.
ExportAllPages(ADocument; const AOutputDir, AFilePrefix: string [; const AOptions])	모든 페이지를 폴더에 렌더링합니다.
ExportPages(ADocument; AFromPage, AToPage: Integer; const AOutputDir, AFilePrefix: string; const AOptions)	페이지 범위를 렌더링합니다.
ExportThumbnail(ADocument; APageIndex; const AFileName: string; AMaxWidth, AMaxHeight: Integer)	지정한 크기에 맞춰 축소한 페이지를 렌더링합니다.
ExportAllThumbnails(ADocument; const AOutputDir, AFilePrefix: string; AMaxWidth, AMaxHeight: Integer)	모든 페이지의 썸네일입니다.

TsgcPDFExportOptions

필드	기본값	설명
Format	efPNG	efBMP, efPNG 또는 efJPEG.
DPI	150	해상도입니다. 72DPI는 1포인트당 1픽셀로 렌더링합니다.
JPEGQuality	85	JPEG 품질이며, 1부터 100까지입니다.
BackgroundColor	$00FFFFFF	페이지 배경입니다(COLORREF).
AntiAlias	False	출력을 부드럽게 처리합니다.

`sgcPDFDefaultExportOptions`로 기본값을 가져올 수 있습니다.

헬퍼 함수

함수	설명
sgcPDFPageToImage(const APDFFile, AImageFile: string; APageIndex: Integer; ADPI: Integer = 150)	파일을 열어 한 페이지를 렌더링합니다.
sgcPDFToImages(const APDFFile, AOutputDir: string; AFormat: TsgcPDFExportFormat = efPNG; ADPI: Integer = 150)	파일을 열어 모든 페이지를 렌더링합니다.

이벤트

TsgcPDFExporter에는 이벤트가 없습니다.

예제

```
static void ExportPages(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  using (TsgcPDFExporter oExporter = new TsgcPDFExporter())
  {
    oPDF.LoadFromFile(AFileName);
    TsgcPDFExportOptions oOptions = sgcPDF_Export.sgcPDFDefaultExportOptions();
    oOptions.Format = TsgcPDFExportFormat.efPNG;
    oOptions.DPI = 150;
    oExporter.ExportPage(oPDF, 0, @"out\page1.png", oOptions);
    oExporter.ExportThumbnail(oPDF, 0, @"out\thumb1.bmp", 200, 200);
  }
  // one call helpers that open the file themselves
  sgcPDF_Export.sgcPDFPageToImage(AFileName, @"out\page1_96.png", 0, 96);
}
```

참고 항목

- 렌더링과 내보내기

TsgcPDFOptimizer

유닛: sgcPDF_Optimize
PDF 파일을 다시 작성하여 크기를 줄입니다. 압축되지 않은 스트림을 압축하고, 중복되거나 참조되지 않는 객체를 제거하며, 이미지를 다운샘플링하고 메타데이터, 북마크, 주석, JavaScript, 썸네일 같은 선택적 콘텐츠를 제거할 수 있습니다.

메서드

메서드	설명
Optimize(const AInputFile, AOutputFile: string)	파일을 최적화합니다.
OptimizeStream(AInputStream, AOutputStream: TStream)	스트림을 최적화합니다.
GetSizeReduction: Int64	마지막 최적화로 절감된 바이트 수입니다.
GetLog: TStringList	마지막 최적화 보고서입니다. 옵티마이저가 소유하므로 직접 해제하지 마십시오.

TsgcPDFOptimizeOptions

필드	기본값	설명
CompressStreams	True	Flate로 압축되지 않은 스트림을 압축합니다.
RemoveDuplicateObjects	True	동일한 객체를 병합합니다.
RemoveUnusedObjects	True	문서에서 도달할 수 없는 객체를 제거합니다.
DownsampleImages	False	MaxImageDPI를 초과하는 이미지를 축소합니다.
MaxImageDPI	150	다운샘플링 목표 해상도입니다.
JPEGQuality	75	다시 압축된 이미지의 품질입니다.
RemoveMetadata	False	XMP 메타데이터를 제거합니다.
RemoveBookmarks	False	개요(북마크)를 제거합니다.
RemoveAnnotations	False	주석을 제거합니다.
RemoveJavaScript	False	JavaScript 동작을 제거합니다.
FlattenForms	False	폼 필드를 평면화합니다.
RemoveThumbnails	False	임베드된 페이지 썸네일을 제거합니다.

`sgcPDFDefaultOptimizeOptions`로 기본값을 가져오거나, `sgcPDFOptimizeFile(const AInputFile, AOutputFile: string)`을 한 번 호출하여 기본값으로 파일을 최적화할 수 있습니다.

이벤트

TsgcPDFOptimizer에는 이벤트가 없습니다.

예제

```
static void OptimizePDF(string AInput, string AOutput)
{
  using (TsgcPDFOptimizer oOptimizer = new TsgcPDFOptimizer())
  {
    TsgcPDFOptimizeOptions oOptions = sgcPDF_Optimize.sgcPDFDefaultOptimizeOptions();
    oOptions.DownsampleImages = true;
    oOptions.MaxImageDPI = 150;
    oOptions.RemoveMetadata = true;
    oOptimizer.Options = oOptions;
    oOptimizer.Optimize(AInput, AOutput);
    Console.WriteLine("Saved " + oOptimizer.GetSizeReduction() + " bytes");
    Console.WriteLine(oOptimizer.GetLog().Text);
  }
}
```

참고 항목

- PDF 파일 최적화

TsgcPDFSignatureHandler

유닛: sgcPDF_Signatures
X.509 인증서와 해당 개인 키로 PDF 문서에 서명합니다. 서명은 Windows CryptoAPI를 통해 생성되는 분리형 PKCS#7(CMS) 서명(*/SubFilter /adbe.pkcs7.detached*)이며, 증분 업데이트로 추가되므로 원본 바이트는 그대로 유지됩니다.

인증서

메서드	설명
LoadCertificateFromPFX(const AFileName: string; const APassword: AnsiString)	PFX(PKCS#12) 파일에서 인증서와 개인 키를 로드합니다.
LoadCertificateFromStore(const ASubjectName: WideString)	현재 사용자의 MY 저장소에서 주체 이름으로 인증서를 찾습니다.
LoadCertificateFromThumbprint(const AThumbprint: AnsiString)	SHA-1 지문으로 인증서를 찾습니다.
ListCertificates: TStringList	저장소의 인증서 목록입니다. 호출자가 목록을 해제해야 합니다.
GetCertificateInfo: TsgcPDFCertificateInfo	로드된 인증서의 SubjectName, IssuerName, SerialNumber, NotBefore, NotAfter, Thumbprint입니다.

서명

메서드	설명
SetField(const AName: AnsiString; APageIndex: Integer; ALeft, ABottom, ARight, ATop: Double)	페이지에 표시되는 서명 필드입니다.
SetInvisible	보이지 않는 서명입니다.
SignDocument(const AInputFile, AOutputFile: string)	파일에 서명합니다.
SignDocumentStream(AInputStream, AOutputStream: TStream)	스트림에 서명합니다.

속성

속성	형식	설명
SignerName (published)	WideString	서명에 기록되는 서명자 이름입니다. Name과 동일합니다.
Reason (published)	WideString	서명 이유입니다.
Location (published)	WideString	서명 장소입니다.

속성	형식	설명
ContactInfo (published)	AnsiString	연락처 정보입니다.
HashAlgorithm (published)	TsgcPDFSignatureHashAlgorithm	shaSHA1, shaSHA256(기본값), shaSHA384 또는 shaSHA512.
Appearance	TsgcPDFSignatureAppearance	표시되는 서명 모양입니다: ShowName, ShowDate, ShowReason, ShowLocation, Text, FontName, FontSize, ImageData.
CertInfo	TsgcPDFCertificateInfo	로드된 인증서의 정보입니다.

이벤트

TsgcPDFSignatureHandler에는 이벤트가 없습니다.

예제

```
static void SignPDF(string AInput, string AOutput)
{
  using (TsgcPDFSignatureHandler oSigner = new TsgcPDFSignatureHandler())
  {
    oSigner.LoadCertificateFromPFX("signer.pfx", "test123");
    Console.WriteLine("Signing as " + oSigner.GetCertificateInfo().SubjectName);
    oSigner.Reason = "Document approval";
    oSigner.Location = "Barcelona";
    oSigner.HashAlgorithm = TsgcPDFSignatureHashAlgorithm.shaSHA256;
    // visible signature field on page 1: Left, Bottom, Right, Top
    oSigner.SetField("Signature1", 0, 50, 50, 250, 100);
    oSigner.SignDocument(AInput, AOutput);
  }
}
```

참고 항목

- TsgcPDFSignatureVerifier
- 전자 서명

TsgcPDFSignatureVerifier

유닛: sgcPDF_Signatures
PDF 문서의 전자 서명을 검증합니다. 각 서명이 암호학적으로 유효한지, 서명된 바이트 범위가 여전히 파일 전체를 포함하는지, 서명자 인증서가 Windows 인증서 저장소의 신뢰할 수 있는 루트로 체인되는지를 확인합니다.

메서드

메서드	설명
LoadDocument(const AFileName: string)	서명된 PDF를 로드합니다.
GetSignatureCount: Integer	서명 수입니다.
VerifySignature(AIndex: Integer): Boolean	서명이 서명된 바이트와 일치하면 True입니다.
IsDocumentModified(AIndex: Integer): Boolean	이 서명 이후 파일이 변경되었으면 True입니다.
IsSignatureTrusted(AIndex: Integer): Boolean	인증서 체인을 Windows가 신뢰하면 True입니다.
GetSignatureInfo(AIndex: Integer): TsgcPDFCertificateInfo	서명자의 인증서 정보입니다.

자체 서명된 테스트 인증서는 유효하지만 신뢰할 수 없는 것으로 검증됩니다.

이벤트

TsgcPDFSignatureVerifier에는 이벤트가 없습니다.

예제

```
static void VerifyPDF(string AFileName)
{
  using (TsgcPDFSignatureVerifier oVerifier = new TsgcPDFSignatureVerifier())
  {
    oVerifier.LoadDocument(AFileName);
    for (int i = 0; i < oVerifier.GetSignatureCount(); i++)
    {
      TsgcPDFCertificateInfo oCert = oVerifier.GetSignatureInfo(i);
      Console.WriteLine("Signer:   " + oCert.SubjectName);
      Console.WriteLine("Valid:    " + oVerifier.VerifySignature(i));
      Console.WriteLine("Modified: " + oVerifier.IsDocumentModified(i));
      Console.WriteLine("Trusted:  " + oVerifier.IsSignatureTrusted(i));
    }
  }
}
```

참고 항목

- TsgcPDFSignatureHandler
- 전자 서명

텍스트와 폰트

sgcPDF는 두 종류의 폰트로 텍스트를 그립니다. 모든 PDF 리더가 제공하는 14가지 표준 PDF 폰트와, 파일에 임베드되는 TrueType 폰트입니다.

표준 폰트

표준 폰트는 **TsgcPDFStandardFont**(유닛 sgcPDF_Fonts)로 선택합니다: sfCourier, sfCourierBold, sfCourierOblique, sfCourierBoldOblique, sfHelvetica, sfHelveticaBold, sfHelveticaOblique, sfHelveticaBoldOblique, sfTimesRoman, sfTimesBold, sfTimesItalic, sfTimesBoldItalic, sfSymbol, sfZapfDingbats입니다. 임베드되지 않으므로 파일 크기를 작게 유지하지만, WinAnsi 문자 집합만 지원합니다. `TsgcPDFPage.DrawText`와 `DrawTextColor`로 사용하십시오. `sgcPDFMeasureTextWidth(AFont, AText, AFontSize)`는 문자열의 너비를 포인트 단위로 반환하며, 이를 이용해 텍스트를 가운데 정렬하거나 오른쪽 정렬할 수 있습니다.

TrueType과 유니코드

`TsgcPDFDocument.AddTrueTypeFont(AFontName, ABold, AItalic)`는 패밀리 이름으로 설치된 Windows 폰트를 로드하고 폰트 인덱스를 반환합니다. `DrawTextTT`나 `DrawTextTTColor`로 그리는 텍스트는 WideString이므로, 폰트에 포함된 어떤 문자든 사용할 수 있습니다. 문서를 저장할 때는 실제로 사용된 글리프만(폰트 서브셋) ToUnicode 맵과 함께 임베드되므로, 텍스트를 검색하고 복사하고 추출할 수 있습니다.

```
static void CreateFontsPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // TrueType fonts are loaded from the installed system fonts, subset and embedded
    int vArial = oPDF.AddTrueTypeFont("Arial", false, false);
    int vTimes = oPDF.AddTrueTypeFont("Times New Roman", true, false);
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // the 14 standard fonts need no embedding
    oPage.DrawText("Standard Helvetica", 50, 780, TsgcPDFStandardFont.sfHelvetica, 14);
    oPage.DrawText("Standard Courier", 50, 760, TsgcPDFStandardFont.sfCourier, 14);
    // TrueType text is Unicode
    oPage.DrawTextTT("Arial TrueType", 50, 720, vArial, 14);
    oPage.DrawTextTTColor("Times New Roman Bold", 50, 700, vTimes, 14, 0.6, 0, 0);
    oPage.DrawTextTT("Cyrillic: \u041F\u0440\u0438\u0432\u0435\u0442", 50, 670, vArial, 14);
    oPage.DrawTextTT("Greek: \u0391\u03BB\u03C6\u03B1", 50, 650, vArial, 14);
    // measure a standard font string to right align it at x = 545
    double vWidth = sgcPDF_Fonts.sgcPDFMeasureTextWidth(TsgcPDFStandardFont.sfHelvetica, "Right aligned", 12);
    oPage.DrawText("Right aligned", 545 - vWidth, 620, TsgcPDFStandardFont.sfHelvetica, 12);
    oPDF.SaveToFile(AFileName);
  }
}
```

CJK 텍스트

중국어, 일본어, 한국어 텍스트도 같은 방식으로 동작합니다. 해당 문자를 포함한 TrueType 폰트를 추가하고 DrawTextTT로 그리십시오. 폰트는 Identity-H 인코딩을 사용하는 CID 폰트로 임베드됩니다. **TsgcPDFCJKFont** 열거형(유닛 sgcPDF_Fonts_CJK)은 일반적인 Windows CJK 폰트를 나열하며, `sgcPDFCJKFontNameFromEnum`은 해당 패밀리 이름을 반환합니다.

값	폰트	스크립트
cjkMSGothic, cjkMSMincho, cjkMeiryo	MS Gothic, MS Mincho, Meiryo	일본어
cjkSimSun, cjkSimHei, cjkMicrosoftYaHei	SimSun, SimHei, Microsoft YaHei	중국어 간체

값	폰트	스크립트
cjkMingLiU, cjkMicrosoftJhengHei	MingLiU, Microsoft JhengHei	중국어 번체
cjkBatang, cjkGulim, cjkMalgunGothic	Batang, Gulim, Malgun Gothic	한국어

```
static void CreateCJKPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // any installed TrueType font that covers the script works the same way
    int vKorean = oPDF.AddTrueTypeFont(sgcPDF_Fonts_CJK.sgcPDFCJKFontNameFromEnum(TsgcPDFCJKFont.cjkMalgunGothic),
      false, false);
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // "Annyeonghaseyo" in Hangul
    oPage.DrawTextTT("\uC548\uB155\uD558\uC138\uC694", 50, 780, vKorean, 20);
    oPDF.SaveToFile(AFileName);
  }
}
```

폰트는 PDF를 생성하는 컴퓨터에 설치되어 있어야 합니다. 글리프가 임베드되므로 PDF를 읽는 쪽에는 폰트가 필요하지 않습니다.

참고 항목

- TsgcPDFDocument
- 텍스트 추출
- PDF/A

이미지

sgcPDF는 JPEG, PNG, BMP 이미지를 임베드합니다. JPEG 데이터는 다시 압축하지 않고 있는 그대로(DCTDecode) 저장됩니다. PNG와 BMP 픽셀은 Flate로 압축됩니다.
`TsgcPDFDocument.AddImage`(또는 `AddImageFromStream`)로 이미지를 한 번 추가한 다음, `TsgcPDFPage.DrawImageFromIndex`로 필요한 만큼 여러 번 그리십시오. 파일에는 이미지가 한 번만 저장됩니다. `DrawImageFile`은 한 번의 호출로 파일을 로드하고 그립니다. X와 Y는 이미지의 왼쪽 아래 모서리이고, W와 H는 페이지에서의 크기(포인트 단위)입니다.

```
static void CreateImagesPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // JPEG is embedded as is (DCTDecode), PNG and BMP are recompressed
    int vLogo = oPDF.AddImage("img.png");
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // the image is added once and can be drawn many times
    oPage.DrawImageFromIndex(vLogo, 50, 600, 200, 100);
    oPage.DrawImageFromIndex(vLogo, 300, 600, 100, 50);
    // or load and draw a file in one call
    oPage.DrawImageFile("img.jpg", 50, 400, 300, 150);
    oPDF.SaveToFile(AFileName);
  }
}
```

TsgcPDFImage 클래스(유닛 sgcPDF_Images)는 문서 없이도 이미지를 로드하고 검사할 수 있습니다. Width, Height, BitsPerComponent, ColorSpace, HasAlpha입니다. TIFF 같은 다른 형식은 예외를 발생시킵니다.

참고 항목

- TsgcPDFDocument
- PDF 파일 최적화

그래픽, 투명도, 셰이딩

TsgcPDFPage에는 일반적인 도형을 위한 헬퍼가 있습니다. DrawLine, DrawRectangle, DrawRectangleFill, DrawCircle, DrawCircleFill과 그 색상 버전입니다. 투명도는 `SetOpacity`(이후의 작업에 영향을 줌)나 `DrawRectangleTransparent`를 사용하십시오. `DrawLinearGradient`는 첫 번째 색상에서 두 번째 색상까지 얇은 색상 띠로 그리는 색상 그라디언트로 사각형을 채웁니다.

콘텐츠 빌더

`TsgcPDFPage.Content`는 페이지의 **TsgcPDFContentBuilder**(유닛 sgcPDF_ContentStream)를 반환합니다. PDF 그리기 연산자를 하나씩 제공하며, 여기서 기록하는 모든 내용은 헬퍼와 동일한 페이지 콘텐츠로 들어갑니다.

- 그래픽 상태: SaveState, RestoreState, SetMatrix, SetLineWidth, SetLineCap, SetLineJoin, SetMiterLimit, SetDashPattern.
- 색상: SetStrokeColorRGB, SetFillColorRGB, SetStrokeColorCMYK, SetFillColorCMYK, SetStrokeGray, SetFillGray.
- 경로: MoveTo, LineTo, CurveTo, CurveToV, CurveToY, Rectangle, ClosePath.
- 칠하기: Stroke, CloseAndStroke, Fill, FillEvenOdd, FillAndStroke, EndPath, ClipNonZero, ClipEvenOdd.
- 텍스트: BeginText, EndText, SetFont, SetCharSpacing, SetWordSpacing, SetTextRise, SetTextRenderMode, SetTextScale, MoveTextPos, SetTextMatrix, ShowText.
- 기타: SetGraphicsState, DrawXObject, BeginMarkedContent, EndMarkedContent, 그리고 임의의 연산자를 쓰는 AddOperator입니다.

변경 사항이 이후의 그리기로 새어나가지 않도록 SaveState는 항상 RestoreState와 짝을 이루어 사용하십시오.

```
static void CreateGraphicsPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // shapes, gradients and transparency through the page helpers
    oPage.DrawLinearGradient(50, 680, 495, 100, 0.2, 0.4, 0.8, 0.9, 0.9, 1.0);
    oPage.DrawRectangleFill(70, 520, 200, 120, 1, 1, 0);
    oPage.DrawRectangleTransparent(150, 550, 200, 120, 1, 0, 0, 0.5);
    oPage.SetOpacity(0.6);
    oPage.DrawCircleFill(450, 580, 60, 0, 0.5, 0);
    oPage.SetOpacity(1.0);
    // low level path operators through the content builder
    TsgcPDFContentBuilder oContent = oPage.Content;
    oContent.SaveState();
    oContent.SetStrokeColorCMYK(0, 1, 1, 0);
    oContent.SetLineWidth(3);
    oContent.SetDashPattern(new double[] { 6, 3 }, 0);
    oContent.MoveTo(50, 400);
    oContent.CurveTo(150, 500, 250, 300, 350, 400);
    oContent.Stroke();
    oContent.RestoreState();
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- TsgcPDFDocument
- 레이어

표

TsgcPDFTable(유닛 sgcPDF_Tables)은 행과 열을 배치하고, 셀 안의 텍스트를 줄바꿈하며, 테두리와 배경을 그립니다.

- 열 수로 생성한 다음, SetColumnWidths, SetColumnWidth 또는 SetColumnWidthPercent로 너비를 설정하십시오.
- SetHeaders는 머리글 행을 추가합니다. AddRowData는 문자열 배열로 데이터 행을 추가합니다. AddRow와 AddHeaderRow는 셀을 완전히 제어할 수 있는 TsgcPDFTableRow를 반환합니다(Style, ColSpan, RowSpan을 가진 TsgcPDFTableCell).
- 스타일 지정: SetDefaultFont, SetHeaderFont, SetHeaderBackground, SetAlternateRowColors, SetBorder, SetPadding, 또는 DefaultStyle과 HeaderStyle 레코드입니다.

그리기

메서드	설명
DrawOnPage(APage: TObject; AX, AY: Double)	TsgcPDFPage에 전체 표를 그립니다. AX, AY는 왼쪽 위 모서리입니다.
DrawOnPageWithOverflow(APage: TObject; AX, AY, AMaxHeight: Double): Integer	머리글 행과 AMaxHeight에 맞는 데이터 행을 그리고, 맞지 않은 첫 번째 행의 인덱스를 반환합니다.
DrawMultiPage(ADocument: TObject; AX, AY, AMaxHeight: Double; APageWidth, APageHeight: Double)	TsgcPDFDocument에 지정한 크기의 페이지를 추가하고, AX, AY부터 표를 그리며, 필요에 따라 페이지를 더 추가하면서 각 페이지에 머리글 행을 반복합니다.

```
static void CreateTablePDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFTable oTable = new TsgcPDFTable(3);
    oTable.SetColumnWidths(200, 150, 145);
    oTable.SetHeaders("Product", "Quantity", "Price");
    oTable.SetHeaderBackground(0.2, 0.4, 0.7);
    oTable.SetAlternateRowColors(0.93, 0.95, 1.0);
    oTable.SetBorder(0.5, 0.6, 0.6, 0.6);
    for (int i = 1; i <= 80; i++)
      oTable.AddRowData("Item " + i, (i * 3).ToString(), (i * 10) + ".00");
    // rows that do not fit continue on new A4 pages
    oTable.DrawMultiPage(oPDF, 50, 790, 740, 595, 842);
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- TsgcPDFDocument

바코드

TsgcPDFBarcodeGenerator(유닛 sgcPDF_Barcodes)는 바코드를 벡터 그래픽으로 그리므로 어떤 크기에서도 선명하게 인쇄됩니다. 지원하는 심볼로지(TsgcPDFBarcodeType)는 bcCode128, bcCode39, bcEAN13, bcEAN8, bcUPCA, bcITF, bcCodabar, bcQRCode, bcDataMatrix, bcPDF417, bcGS1_128입니다. QR Code 버전과 오류 정정 수준, DataMatrix, PDF417, GS1-128은 2D 바코드와 GS1-128에서 설명합니다.
Draw 메서드의 첫 번째 매개변수는 페이지(TsgcPDFPage)나 그 **콘텐츠 빌더**인 `TsgcPDFPage.Content`입니다. 다른 객체를 전달하면 예외가 발생합니다. AX와 AY는 왼쪽 아래 모서리입니다.

메서드	설명
DrawBarcode(APage: TObject; ABarcodeType; const AData: AnsiString; AX, AY: Double)	기본 옵션으로 그립니다.
DrawBarcode(APage: TObject; ABarcodeType; const AData: AnsiString; AX, AY: Double; const AOptions: TsgcPDFBarcodeOptions)	사용자 지정 옵션으로 그립니다.
DrawQRCode(APage: TObject; const AData: AnsiString; AX, AY, AModuleSize: Double)	모듈 너비가 AModuleSize 포인트인 QR Code를 그립니다.
EncodeLinear(ABarcodeType: TsgcPDFBarcodeType; const AData: AnsiString): TBytes	선형 심볼로지(Code 128, Code 39, EAN-13, EAN-8, UPC-A, ITF, Codabar, GS1-128)의 막대와 공백 너비를 모듈 단위로 막대부터 순서대로 반환합니다. DrawBarcode가 그리는 것과 같은 너비입니다. 2D 형식은 예외를 발생시키며, 자체 인코더가 있습니다. 참조: 2D 바코드와 GS1-128.

TsgcPDFBarcodeOptions

`sgcPDFDefaultBarcodeOptions`로 기본값을 가져올 수 있습니다. BarWidth 1.0, Height 50, ShowText True, TextFont sfHelvetica, TextFontSize 8, QuietZone 10, 검은색 전경과 흰색 배경(ForegroundR/G/B, BackgroundR/G/B)입니다.

```
static void CreateBarcodesPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // barcodes are drawn into the page content builder
    TsgcPDFBarcodeGenerator oBarcodes = new TsgcPDFBarcodeGenerator();
    oBarcodes.DrawBarcode(oPage.Content, TsgcPDFBarcodeType.bcCode128, "SGC-2026-0001", 50, 700);
    TsgcPDFBarcodeOptions oOptions = sgcPDF_Barcodes.sgcPDFDefaultBarcodeOptions();
    oOptions.Height = 60;
    oOptions.BarWidth = 1.2;
    oBarcodes.DrawBarcode(oPage.Content, TsgcPDFBarcodeType.bcEAN13, "5901234123457", 50, 580, oOptions);
    // QR code, 3 points per module
    oBarcodes.DrawQRCode(oPage.Content, "https://www.esegece.com", 350, 580, 3);
    oPDF.SaveToFile(AFileName);
  }
}
```

바코드 컨트롤

폼에 바코드를 표시하려면 TsgcPDFBarcodeControl(VCL) 또는 TsgcPDFBarcodeFMX(FireMonkey)를 올려 놓고 Symbology와 Text를 설정합니다. 두 컨트롤 모두 2D 바코드와 GS1-128에서 설명하는 2D 심볼과 GS1-128을 포함한 11가지 심볼로지를 지원하며 모든 모듈을 정수 픽셀로 그립니다. 둘 다 TsgcPDFBarcodeSymbol을 호스팅하며, 이것은 폼 없이도, 예를 들어 서비스나 Linux에서도 동작합니다.

- SaveToPNG와 SaveToSVG는 심볼을 이미지로 저장합니다.
- DrawToPDF는 컨트롤과 같은 모듈, 회전, 여백 영역, 텍스트 줄로 심볼을 벡터 사각형으로 PDF 페이지에 그립니다.
- Runs, ModuleAt, LayoutAxes는 사용자 지정 그리기를 위해 모듈과 픽셀 레이아웃을 반환합니다.

참고 항목

- TsgcPDFDocument
- 2D 바코드와 GS1-128
- TsgcPDFBarcodeControl
- TsgcPDFBarcodeSymbol

주석

주석은 `TsgcPDFDocument.Annotations`(TsgcPDFAnnotationManager, 유닛 sgcPDF_Annotations)가 관리합니다. 각 Add 메서드는 0 기반 페이지 인덱스를 받고 새 주석을 반환하므로, 색상(SetColor), Author, Contents, Opacity, Flags를 설정할 수 있습니다.

메서드	주석
AddText(APageIndex; ALeft, ABottom, ARight, ATop: Double; const AText: WideString)	스티커 메모입니다(TsgcPDFTextAnnotation, Icon과 Open 포함).
AddLink(APageIndex; ALeft, ABottom, ARight, ATop: Double; ADestPage: Integer)	다른 페이지로의 링크입니다.
AddLinkURI(APageIndex; ALeft, ABottom, ARight, ATop: Double; const AURI: AnsiString)	웹 주소로의 링크입니다.
AddFreeText(APageIndex; ALeft, ABottom, ARight, ATop: Double; const AText: WideString)	텍스트 상자입니다(SetFont, SetTextColor, Alignment).
AddHighlight / AddUnderline / AddStrikeOut(APageIndex; const AQuadPoints: array of Double)	텍스트 마크업입니다. 8개 숫자가 하나의 사각형을 나타냅니다.
AddStamp(APageIndex; ALeft, ABottom, ARight, ATop: Double; const AStampName: AnsiString)	Approved나 Draft 같은 고무도장입니다.
AddLine(APageIndex; AX1, AY1, AX2, AY2: Double)	StartStyle과 EndStyle이 있는 선입니다(예: OpenArrow).
AddInk(APageIndex; const APoints: array of Double)	X, Y 쌍으로 이루어진 자유형 그리기입니다.
AddSquare / AddCircle(APageIndex; ALeft, ABottom, ARight, ATop: Double)	사각형 또는 타원(TsgcPDFShapeAnnotation, SetInteriorColor 사용).
AddReply(AParent: TsgcPDFAnnotation; const AText: WideString)	AParent에 답글을 다는 텍스트 주석.

문서를 로드하면 기존 주석이 형식이 있는 객체로 동일한 관리자에 읽혀서, `Count`, `GetAnnotationsForPage`, `Annotations`로 확인할 수 있습니다. 이 주석은 변경한 뒤 다시 저장할 수 있습니다. 아래의 "주석 읽기"와 "편집과 저장"을 참고하십시오.

```
static void CreateAnnotationsPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.AddPage(595, 842).DrawText("Annotated page", 70, 790, TsgcPDFStandardFont.sfHelveticaBold, 20);
    // sticky note: page index, then Left, Bottom, Right, Top
    TsgcPDFTextAnnotation oNote = oPDF.Annotations.AddText(0, 70, 680, 94, 704, "Please review");
    oNote.SetColor(1, 1, 0);
    oNote.Icon = "Comment";
    oNote.Author = "Reviewer";
    // web link
    oPDF.Annotations.AddLinkURI(0, 70, 625, 280, 645, "https://www.esegece.com");
    // highlight: one quadrilateral, 8 numbers (x1 y1 x2 y2 x3 y3 x4 y4)
    TsgcPDFHighlightAnnotation oHighlight = oPDF.Annotations.AddHighlight(0,
      70, 786, 240, 786, 70, 806, 240, 806);
    oHighlight.SetColor(1, 1, 0);
    oHighlight.Opacity = 0.5;
    oPDF.Annotations.AddStamp(0, 350, 420, 530, 470, "Approved");
    TsgcPDFLineAnnotation oLine = oPDF.Annotations.AddLine(0, 70, 350, 300, 350);
    oLine.SetColor(1, 0, 0);
```

```
    oLine.EndStyle = "OpenArrow";
    oPDF.SaveToFile(AFileName);
  }
}
```

도형

`AddSquare`와 `AddCircle`은 지정한 사각형, 또는 그 사각형에 내접하는 타원을 추가하고 **TsgcPDFShapeAnnotation**을 반환합니다. SetColor는 테두리 색상을, SetInteriorColor는 채우기 색상을 설정하며(호출하지 않으면 채우기가 없습니다), BorderWidth는 테두리 두께를 포인트 단위로 지정합니다. 도형은 모양 스트림과 함께 저장되므로 모든 뷰어가 동일하게 그립니다.

```
static void CreateShapesPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.AddPage(595, 842);
    // rectangle: page index, then Left, Bottom, Right, Top
    TsgcPDFShapeAnnotation oBox = oPDF.Annotations.AddSquare(0, 70, 600, 270, 700);
    oBox.SetColor(0, 0, 1);
    oBox.SetInteriorColor(0.85, 0.9, 1);
    oBox.BorderWidth = 2;
    oBox.Contents = "Check this block";
    // ellipse inscribed in the rectangle
    TsgcPDFShapeAnnotation oCircle = oPDF.Annotations.AddCircle(0, 320, 600, 420, 700);
    oCircle.SetColor(1, 0, 0);
    oPDF.SaveToFile(AFileName);
  }
}
```

주석 읽기

LoadFromFile과 LoadFromStream은 모든 페이지의 주석을 `Annotations`로 읽습니다. 각 주석은 해당 유형의 클래스로 생성되고 속성은 파일의 내용으로 채워지므로, 일반적인 `is` 검사나 캐스트로 접근할 수 있습니다.

파일의 서브타입	읽는 클래스와 속성
Text	TsgcPDFTextAnnotation: Icon, Open.
Link	TsgcPDFLinkAnnotation: DestPage 또는 웹 주소.
FreeText	TsgcPDFFreeTextAnnotation: 글꼴, 글자 색상, Alignment.
Highlight, Underline, StrikeOut	TsgcPDFHighlightAnnotation(어느 유형인지는 AnnotType으로 구분): QuadCount, QuadPoint.
Stamp	TsgcPDFStampAnnotation: StampName.
Line	TsgcPDFLineAnnotation: X1, Y1, X2, Y2, StartStyle, EndStyle.
Ink	TsgcPDFInkAnnotation: PathCount, PathPointCount, GetPathPoint.
Square, Circle	TsgcPDFShapeAnnotation: HasInteriorColor, InteriorR, InteriorG, InteriorB.
Popup, FileAttachment, Widget 등	기본 클래스인 TsgcPDFAnnotation.

모든 주석 클래스에는 파일에서 로드한 주석과 새 주석 모두에 대해 다음 멤버가 있습니다.

멤버	설명
Author, Contents: WideString	작성자(/T)와 텍스트(/Contents). Acrobat 등의 뷰어가 악센트 문자나 비라틴 문자 코멘트를 기록할 때처럼 UTF-16으로 저장된 텍스트는 유니코드로 디코딩됩니다.
ModDate: TDateTime	수정 날짜(/M). 로컬 시간이며 없으면 0입니다.
Name: AnsiString	주석 이름(/NM).
InReplyTo: TsgcPDFAnnotation	이 주석이 답글을 다는 대상, 또는 함께 그룹화된 주석(/IRT). 최상위 주석이면 nil입니다.
ReplyType: AnsiString	답글이면 비어 있거나 R, 그룹 멤버이면 Group(/RT).
IsReply: Boolean	InReplyTo가 설정되어 있고 ReplyType이 Group이 아니면 True입니다.
BorderWidth: Double	테두리 또는 선의 두께(포인트 단위).
HasColor, ColorR, ColorG, ColorB	색상(/C). 파일에서 읽었거나 SetColor로 설정한 값입니다. 구성 요소는 0에서 1까지입니다.
Modified: Boolean	로드된 주석이 문서를 로드한 이후 또는 마지막 ResetModified 이후에 변경되었으면 True입니다. 사용자 인터페이스용 플래그입니다. 코드에서 만든 주석은 항상 False입니다.
ChangedSinceLoad: Boolean	로드된 주석이 문서를 로드했을 때의 상태와 다르면 True이며, 이 경우 다음 저장에서 해당 주석을 다시 기록합니다. ResetModified는 이 값을 바꾸지 않습니다. 코드에서 만든 주석은 항상 False입니다.
MoveBy(ADX, ADY: Double)	주석을 ADX, ADY 포인트만큼 이동합니다. Rect와 선의 점, 사각형 점, 잉크 경로가 이동 대상입니다.
SourceObjectNumber: Integer	로드된 주석의 파일 내 객체 번호이며, 코드에서 만든 주석은 0입니다.

선, 텍스트 마크업, 잉크의 기하 정보는 X1, Y1, X2, Y2(TsgcPDFLineAnnotation), QuadCount와 QuadPoint(AIndex)(AIndex 는 0 부 터 QuadCount * 8 - 1 까 지, TsgcPDFHighlightAnnotation), PathCount, PathPointCount(APath), GetPathPoint(APath, AIndex, AX, AY)(TsgcPDFInkAnnotation)로 읽습니다.

```
static void ListAnnotations(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    foreach (TsgcPDFAnnotation oAnnot in oPDF.Annotations.Annotations)
    {
      // replies are listed under their parent, popups have no content
      if (oAnnot.IsReply() || (oAnnot.AnnotType == TsgcPDFAnnotationType.atPopup))
        continue;
      Console.Write("Page {0}: {1}", oAnnot.PageIndex + 1, oAnnot.GetType().Name);
      if (oAnnot.Author != "")
        Console.Write(", by " + oAnnot.Author);
      // no date: sgcPDF_Attachments.ZeroDate, the Delphi TDateTime 0
      if (oAnnot.ModDate != sgcPDF_Attachments.ZeroDate)
        Console.Write(", " + oAnnot.ModDate.ToString());
      Console.WriteLine();
      if (oAnnot.Contents != "")
        Console.WriteLine("  " + oAnnot.Contents);
      if (oAnnot is TsgcPDFStampAnnotation oStamp)
        Console.WriteLine("  stamp: " + oStamp.StampName);
      foreach (TsgcPDFAnnotation oReply in oPDF.Annotations.GetReplies(oAnnot))
        Console.WriteLine("  reply from " + oReply.Author + ": " + oReply.Contents);
    }
  }
```

```
}

static void WriteGeometry(TsgcPDFAnnotation AAnnot)
{
  if (AAnnot is TsgcPDFLineAnnotation oLine)
    Console.WriteLine("Line from {0:F1}, {1:F1} to {2:F1}, {3:F1}",
      oLine.X1, oLine.Y1, oLine.X2, oLine.Y2);
  else if (AAnnot is TsgcPDFHighlightAnnotation oMarkup)
  {
    // highlight, underline or strike-out: 8 numbers per quadrilateral
    for (int i = 0; i < oMarkup.QuadCount; i++)
      Console.WriteLine("Quad {0}: first corner {1:F1}, {2:F1}",
        i, oMarkup.QuadPoint(i * 8), oMarkup.QuadPoint(i * 8 + 1));
  }
  else if (AAnnot is TsgcPDFInkAnnotation oInk)
  {
    for (int i = 0; i < oInk.PathCount; i++)
    {
      double vX, vY;
      oInk.GetPathPoint(i, 0, out vX, out vY);
      Console.WriteLine("Path {0}: {1} points, starts at {2:F1}, {3:F1}",
        i, oInk.PathPointCount(i), vX, vY);
    }
  }
  if (AAnnot.HasColor)
    Console.WriteLine("Color {0:F2} {1:F2} {2:F2}, border {3:F1}",
      AAnnot.ColorR, AAnnot.ColorG, AAnnot.ColorB, AAnnot.BorderWidth);
}
```

답글

답글은 `InReplyTo`가 다른 주석(파일의 /IRT 항목)을 가리키는 주석으로, Acrobat에서 코멘트에 다는 답변이 그 예입니다. LoadFromFile은 부모가 다른 페이지에 있는 경우에도 모든 답글을 부모에 연결합니다. 관리자에는 답글과 목록 자체를 위한 다음 멤버가 있습니다.

멤버	설명
GetAnnotationsForPage(APageIndex: Integer): TList	페이지의 주석을 목록 순서대로 반환합니다. 목록은 해제하십시오.
GetReplies(AAnnotation: TsgcPDFAnnotation): TList	주석의 직접 답글을 목록 순서대로 반환합니다. 목록은 해제하십시오.
AddReply(AParent: TsgcPDFAnnotation; const AText: WideString): TsgcPDFTextAnnotation	AParent에 답글을 다는 텍스트 주석을 해당 페이지에 해당 Rect로 추가합니다.
Remove(AAnnotation: TsgcPDFAnnotation)	주석을 제거하고 해제합니다. 그 주석에 답글을 다는 모든 주석, 그 주석과 그룹화된 모든 주석(재귀적), 그리고 팝업도 함께 제거됩니다.
LinkReplies	로드된 주석의 /IRT를 해당 객체 번호를 가진 로드된 주석으로 연결합니다. LoadFromFile이 호출하므로, LoadFromPage 이후에만 직접 호출하십시오.
Modified: Boolean	주석이 추가 또는 제거되었거나 로드된 주석이 변경된 경우, 문서를 로드한 이후 또는 마지막 ResetModified 이후에 True입니다. 사용자 인터페이스용 "저장하지 않은 변경 사항" 플래그입니다.
ChangedSinceLoad: Boolean	주석이 추가 또는 제거되었거나 로드된 주석이 파일과 다른 경우, 문서를 로드한 이후에 True입니다. ResetModified는 이 값을 바꾸지 않습니다.

멤버	설명
ResetModified	Modified를 지웁니다. 예를 들어 저장한 뒤에 사용합니다. 편집 내용은 유지되며 다음 저장에서도 기록됩니다. "변경 추적"을 참고하십시오.
MarkLoaded	현재 상태를 파일의 상태로 간주하고 Modified와 ChangedSinceLoad를 지웁니다. 문서가 로드한 뒤에 이를 호출합니다. "변경 추적"을 참고하십시오.

답글은 SaveToFile과 SaveIncremental에 의해 부모를 가리키는 /IRT와 함께 기록되므로, sgcPDF와 다른 뷰어에서도 답글로 읽힙니다. AddReply는 Author와 ModDate를 설정하지 않으므로 직접 설정하십시오.

편집과 저장

로드된 주석의 속성은 새 주석과 마찬가지로 변경할 수 있습니다. Contents, Author, Rect, 색상, Flags, ModDate 등을 바꾸거나 MoveBy를 사용할 수 있습니다. SaveToFile과 SaveIncremental이 변경 내용을 기록합니다.

- SaveIncremental은 변경된 주석을 파일의 원래 바이트 뒤에 자신의 객체 번호로 다시 기록합니다. 새 주석과 답글은 같은 리비전에서 해당 페이지에 추가됩니다.
- 코드에서 ModDate를 변경하지 않았다면 현재 시간으로 설정됩니다.
- Contents, Author, Name, ModDate, Rect, Flags 또는 답글 항목만 변경되면 파일의 모양 스트림이 유지됩니다. 그 밖의 변경에서는 스탬프와 도형에 새 모양이 만들어지고, 다른 유형은 모양이 제거되어 뷰어가 속성으로부터 다시 그립니다.
- Remove는 주석을 그 답글과 팝업과 함께 삭제하며, 다음 저장에서는 이들이 없는 페이지가 기록됩니다.

SaveIncremental은 원래 바이트를 변경하지 않으므로 기존 모든 서명의 바이트 범위는 그대로이고 서명 값은 유효하게 유지됩니다. 검증 도구가 새 리비전을 허용하는지는 서명의 권한에 따라 달라집니다. DocMDP 권한 P=3인 인증 서명 또는 승인 서명만 있는 경우에는 코멘트를 추가하고 변경할 수 있습니다. P=1 또는 P=2이면 검증 도구는 허용되지 않는 변경으로 보고합니다. SaveToFile로 전체 저장을 하면 파일 전체를 다시 쓰므로 기존 서명이 깨집니다. 서명이 없는 문서에만 사용하십시오. 증분 저장과 압축 출력과 전자 서명을 참고하십시오.

```
static void ReviewNotes(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    foreach (TsgcPDFAnnotation oAnnot in oPDF.Annotations.GetAnnotationsForPage(0))
    {
      if ((oAnnot is TsgcPDFTextAnnotation) && !oAnnot.IsReply())
      {
        // change a loaded note
        oAnnot.Contents = oAnnot.Contents + " (checked)";
        // and answer it
        TsgcPDFTextAnnotation oReply = oPDF.Annotations.AddReply(oAnnot, "Fixed in version 2");
        oReply.Author = "Reviewer";
        oReply.ModDate = DateTime.Now;
        break;
      }
    }
    // appends the changes after the original bytes of the file
    if (oPDF.Annotations.Modified)
      oPDF.SaveIncremental(AFileName);
  }
}
```

변경 추적

두 개의 플래그가 무엇이 변경되었는지 알려주며, 각각 고유한 역할이 있습니다.

- `Modified`는 사용자 인터페이스용 "저장하지 않은 변경 사항" 플래그입니다. `Annotations.Modified`는 주석이 추가 또는 제거되었거나 로드된 주석이 변경된 경우, 문서를 로드한 이후 또는 마지막 ResetModified 이후에 True입니다. 각 주석의 `Modified` 속성은 로드된 주석 중 어느 것이 변경되었는지 알려줍니다. 저장 버튼을 활성화하거나 닫기 전에 사용자에게 확인할 때 사용합니다.

- `ChangedSinceLoad`는 파일과 비교합니다. `Annotations.ChangedSinceLoad`는 주석이 추가 또는 제거되었거나 로드된 주석이 문서를 로드했을 때의 상태와 다른 경우에 True입니다. SaveToFile과 SaveIncremental은 각 주석의 `ChangedSinceLoad` 함수를 사용하여 로드된 주석 중 어느 것을 변경 내용과 함께 기록할지 결정합니다.

`ResetModified`는 Modified만 지웁니다. 저장한 뒤에 호출하면 저장할 내용이 남아 있지 않음을 사용자 인터페이스에 표시할 수 있습니다. ChangedSinceLoad는 바꾸지 않으므로 편집 내용을 잃는 일이 없습니다. 다음 SaveToFile 또는 SaveIncremental은 문서를 로드한 이후에 이루어진 모든 변경을 이후의 변경과 함께 기록합니다. 저장한 파일을 다시 로드할 필요가 없으며, 예제는 같은 문서에서 계속 작업합니다.
`MarkLoaded`는 주석의 현재 상태를 파일의 상태로 간주하고 두 플래그를 모두 지웁니다. LoadFromFile과 LoadFromStream이 주석을 읽은 뒤에 이를 호출하므로 애플리케이션에서 필요한 경우는 드뭅니다. 저장한 뒤에 "저장하지 않은 변경 사항" 플래그를 지우려면 대신 ResetModified를 호출하십시오.

```
static void MoveNoteAndSave(TsgcPDFDocument ADocument, string AFileName)
{
  foreach (TsgcPDFAnnotation oAnnot in ADocument.Annotations.Annotations)
  {
    if (oAnnot is TsgcPDFTextAnnotation)
    {
      // 20 points right and 20 points down
      oAnnot.MoveBy(20, -20);
      Console.WriteLine("Note changed: " + oAnnot.Modified);
      break;
    }
  }
  if (ADocument.Annotations.Modified)
  {
    ADocument.SaveIncremental(AFileName);
    // saved: clear the "unsaved changes" flag of the user interface
    ADocument.Annotations.ResetModified();
  }
  Console.WriteLine("Unsaved changes: " + ADocument.Annotations.Modified);
  // still true: the next save writes the note again, with the later changes
  Console.WriteLine("Changed since load: " + ADocument.Annotations.ChangedSinceLoad());
}
```

편집된 주석 렌더링

고급 용도로, `TsgcPDFDocument.BuildPageAnnotations(APageIndex)`는 페이지의 주석을 편집 내용을 포함한 현재 상태로, 렌더러의 Annotations 속성에 사용할 TsgcPDFArray로 반환합니다. 배열은 호출한 쪽에서 해제합니다. 뷰어는 주석이 변경된 페이지를 그릴 때 이를 사용하므로 대부분의 애플리케이션은 직접 호출하지 않습니다.

참고 항목

- TsgcPDFDocument
- 북마크
- 증분 저장과 압축 출력
- TsgcPDFViewerControl
- TsgcPDFViewerFMX
- TsgcPDFCommentsView

북마크

북마크(문서 개요)는 `TsgcPDFDocument.Bookmarks`(TsgcPDFBookmarks, 유닛 sgcPDF_Bookmarks)가 관리합니다. `Add`는 최상위 항목을, `TsgcPDFBookmarkItem.AddChild`는 하위 항목을 생성합니다. 각 항목에는 Title, PageIndex, Top, Left, Zoom, ColorR/G/B, Bold, Italic, Open이 있습니다.

```
static void CreateBookmarksPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    for (int i = 0; i <= 4; i++)
      oPDF.AddPage(595, 842).DrawText("Page " + (i + 1), 50, 780, TsgcPDFStandardFont.sfHelvetica, 14);
    oPDF.Bookmarks.Add("Cover", 0);
    TsgcPDFBookmarkItem oChapter = oPDF.Bookmarks.Add("Chapter 1", 1);
    oChapter.Bold = true;
    oChapter.Open = true;
    oChapter.AddChild("Section 1.1", 2);
    oChapter.AddChild("Section 1.2", 3);
    oPDF.Bookmarks.Add("Appendix", 4);
    oPDF.SaveToFile(AFileName);
  }
}
```

북마크 읽기

로드된 문서의 북마크도 동일한 속성으로 확인할 수 있습니다.

```
static void PrintBookmarks(TsgcPDFBookmarkItem AItem, string AIndent)
{
  Console.WriteLine(AIndent + AItem.Title + " -> page " + (AItem.PageIndex + 1));
  for (int i = 0; i < AItem.ChildCount(); i++)
    PrintBookmarks(AItem.GetChild(i), AIndent + "  ");
}

static void ListBookmarks(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    for (int i = 0; i < oPDF.Bookmarks.Count(); i++)
      PrintBookmarks(oPDF.Bookmarks.GetItem(i), "");
  }
}
```

참고 항목

- TsgcPDFDocument
- 주석

레이어

레이어(선택적 콘텐츠 그룹)를 사용하면 리더가 페이지의 일부를 표시하거나 숨길 수 있습니다. `TsgcPDFDocument.Layers`(TsgcPDFLayerManager, 유닛 sgcPDF_Layers)가 관리합니다.

- `AddLayer(AName, ADefaultState)`는 처음에 켜짐(lsOn) 또는 꺼짐(lsOff) 상태인 레이어를 생성합니다. TsgcPDFLayer에는 Intent, Visible, Printable, Locked도 있습니다.
- 콘텐츠는 `BeginLayerMarkedContent(ALayer)`와 `EndLayerMarkedContent` 사이에 기록될 때 해당 레이어에 속합니다. 둘 다 연산자 텍스트를 반환하며, `Content.AddOperator`로 페이지에 추가합니다.
- `AddGroup`은 리더의 레이어 패널에서 레이어를 그룹으로 묶습니다. `FindLayer`, `GetLayer`, `LayerCount`, `RemoveLayer`는 목록을 관리하며, 문서를 로드할 때도 이 목록이 채워집니다.

```
static void CreateLayersPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.Layers.AddLayer("Background", TsgcPDFLayerState.lsOn);
    TsgcPDFLayer oNotes = oPDF.Layers.AddLayer("Notes", TsgcPDFLayerState.lsOff);
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    oPage.DrawText("Always visible", 50, 780, TsgcPDFStandardFont.sfHelvetica, 14);
    // wrap the drawing operators that belong to the layer in marked content
    oPage.Content.AddOperator(oPDF.Layers.BeginLayerMarkedContent(oNotes));
    oPage.DrawText("Only visible when the Notes layer is on", 50, 740,
      TsgcPDFStandardFont.sfHelvetica, 14);
    oPage.Content.AddOperator(oPDF.Layers.EndLayerMarkedContent());
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- TsgcPDFDocument
- 그래픽

폼

대화형 폼(AcroForm)은 `TsgcPDFDocument.AcroForm`(TsgcPDFAcroForm, 유닛 sgcPDF_Forms)이 관리합니다.

필드 만들기

메서드	필드
AddTextField(const AName: AnsiString; APageIndex: Integer; ALeft, ABottom, ARight, ATop: Double)	텍스트 상자입니다.
AddCheckBox(...)	확인란입니다.
AddRadioGroup(const AName: AnsiString) and AddRadioButton(AGroup; APageIndex; ALeft, ABottom, ARight, ATop; const AExportValue)	라디오 버튼입니다.
AddComboBox(...; const AOptions: array of WideString)	드롭다운 목록입니다.
AddListBox(...; const AOptions: array of WideString)	목록 상자입니다.
AddButton(...; const ACaption: WideString)	푸시 버튼입니다.
AddSignatureField(...)	빈 서명 필드입니다.

각 메서드는 Value, DefaultValue, ReadOnly, Required, MaxLen, FontName, FontSize, Alignment를 가진 TsgcPDFFormField를 반환합니다. 리더가 필드 모양을 다시 생성하도록 하려면 `NeedAppearances`를 True로 설정하십시오.

```
static void CreateFormPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    oPage.DrawText("Name:", 50, 737, TsgcPDFStandardFont.sfHelvetica, 12);
    oPage.DrawText("Country:", 50, 697, TsgcPDFStandardFont.sfHelvetica, 12);
    oPage.DrawText("Subscribe:", 50, 657, TsgcPDFStandardFont.sfHelvetica, 12);
    oPDF.AcroForm.AddTextField("name", 0, 160, 730, 400, 755).FontSize = 12;
    oPDF.AcroForm.AddComboBox("country", 0, 160, 690, 400, 715,
      "Germany", "Spain", "United States");
    oPDF.AcroForm.AddCheckBox("subscribe", 0, 160, 650, 178, 668);
    oPDF.AcroForm.NeedAppearances = true;
    oPDF.SaveToFile(AFileName);
  }
}
```

필드 채우기

문서를 로드하면 필드가 AcroForm으로 읽힙니다. FillField, SetCheckBox, GetFieldValue, FindField, GetField, FieldCount를 사용하십시오.

```
static void FillFormPDF(string AInput, string AOutput)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AInput);
```

```
    for (int i = 0; i < oPDF.AcroForm.FieldCount(); i++)
      Console.WriteLine("Field: " + oPDF.AcroForm.GetField(i).FullName);
    oPDF.AcroForm.FillField("name", "John Smith");
    oPDF.AcroForm.FillField("country", "Spain");
    oPDF.AcroForm.SetCheckBox("subscribe", true);
    oPDF.SaveToFile(AOutput);
  }
}
```

평면화

평면화는 현재 필드 값을 일반 페이지 콘텐츠로 그리고 필드를 제거하므로, 값을 더 이상 편집할 수 없게 됩니다. `TsgcPDFAcroForm.FlattenFields`나 `TsgcPDFEditor.FlattenForm`을 사용하십시오.

```
static void FillAndFlatten(string AInput, string AOutput)
{
  using (TsgcPDFEditor oEditor = new TsgcPDFEditor())
  {
    oEditor.LoadFromFile(AInput);
    oEditor.FillFormField("name", "Jane Doe");
    // the field values become ordinary page content, the fields are removed
    oEditor.FlattenForm();
    oEditor.SaveToFile(AOutput);
  }
}
```

참고 항목

- TsgcPDFDocument
- TsgcPDFEditor

암호화

저장하기 전에 `SetEncryption`을 호출하면 표준 보안 핸들러로 문서를 암호화합니다. **TsgcPDFEncryptionMethod**(유닛 sgcPDF_Encryption)로 알고리즘을 선택합니다.

값	알고리즘
emRC4_40	RC4 40비트(PDF 1.1, 레거시 전용)
emRC4_128	RC4 128비트(PDF 1.4)
emAES_128	AES-128(PDF 1.6)
emAES_256	AES-256(PDF 2.0, 권장)

사용자 암호는 파일을 여는 데 필요합니다. 소유자 암호는 모든 권한을 부여합니다. TsgcPDFDocument와 TsgcPDFEditor의 SetEncryption은 모든 권한(인쇄, 수정, 복사, 주석, 폼 채우기, 추출, 조립, 고품질 인쇄)을 부여합니다.

```
static void CreateEncryptedPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.AddPage(595, 842).DrawText("Secret", 50, 780, TsgcPDFStandardFont.sfHelveticaBold, 24);
    // user password opens the file, owner password grants full rights
    oPDF.SetEncryption(TsgcPDFEncryptionMethod.emAES_256, "user123", "owner456");
    oPDF.SaveToFile(AFileName);
  }
}
```

암호화된 파일 열기

`LoadFromFile` 전에 `SetPassword`를 호출하십시오. 사용자 암호나 소유자 암호 중 어느 것으로도 파일을 열 수 있습니다.

```
static void OpenEncryptedPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // set the password before loading
    oPDF.SetPassword("user123");
    oPDF.LoadFromFile(AFileName);
    Console.WriteLine(oPDF.ExtractTextW(0));
  }
}
```

저수준 접근

TsgcPDFEncryptionHandler는 보안 핸들러 자체를 구현합니다. TsgcPDFPermissions 세트를 사용하는 `SetupEncryption`, `Authenticate`, 그리고 문자열과 스트림에 대한 암호화 및 복호화 루틴을 제공합니다. 이 유닛은 sgcPDFMD5, sgcPDFSHA256, sgcPDFSHA384, sgcPDFSHA512, sgcPDFRC4, sgcPDFAESEncryptCBC, sgcPDFAESDecryptCBC도 제공합니다. 이 유닛의 오류는 **EsgcPDFEncryptionError**를 발생시킵니다.

참고 항목

- TsgcPDFDocument
- 전자 서명

전자 서명

TsgcPDFSignatureHandler는 개인 키를 가진 인증서로 PDF에 서명합니다. 인증서는 PFX 파일에서 로드하거나, Windows 인증서 저장소에서 주체 이름이나 지문으로 찾습니다. 서명은 증분 업데이트로 추가되는 분리형 PKCS#7 서명(adbe.pkcs7.detached)이므로, 서명된 파일의 시작 부분에 있는 원본 파일의 바이트는 그대로 유지됩니다.

서명

```
static void SignPDF(string AInput, string AOutput)
{
  using (TsgcPDFSignatureHandler oSigner = new TsgcPDFSignatureHandler())
  {
    oSigner.LoadCertificateFromPFX("signer.pfx", "test123");
    Console.WriteLine("Signing as " + oSigner.GetCertificateInfo().SubjectName);
    oSigner.Reason = "Document approval";
    oSigner.Location = "Barcelona";
    oSigner.HashAlgorithm = TsgcPDFSignatureHashAlgorithm.shaSHA256;
    // visible signature field on page 1: Left, Bottom, Right, Top
    oSigner.SetField("Signature1", 0, 50, 50, 250, 100);
    oSigner.SignDocument(AInput, AOutput);
  }
}
```

표시되는 서명 사각형을 원하면 `SetField`를, 모양이 없는 서명을 원하면 `SetInvisible`을 호출하십시오. `HashAlgorithm`은 SHA-1, SHA-256(기본값), SHA-384, SHA-512 중에서 선택합니다.

검증

TsgcPDFSignatureVerifier는 각 서명에 대해 유효한지, 서명 후 문서가 변경되었는지, 인증서를 Windows가 신뢰하는지를 알려줍니다.

```
static void VerifyPDF(string AFileName)
{
  using (TsgcPDFSignatureVerifier oVerifier = new TsgcPDFSignatureVerifier())
  {
    oVerifier.LoadDocument(AFileName);
    for (int i = 0; i < oVerifier.GetSignatureCount(); i++)
    {
      TsgcPDFCertificateInfo oCert = oVerifier.GetSignatureInfo(i);
      Console.WriteLine("Signer:   " + oCert.SubjectName);
      Console.WriteLine("Valid:    " + oVerifier.VerifySignature(i));
      Console.WriteLine("Modified: " + oVerifier.IsDocumentModified(i));
      Console.WriteLine("Trusted:  " + oVerifier.IsSignatureTrusted(i));
    }
  }
}
```

에디터로 서명하기

`TsgcPDFEditor.SignDocument(AOutputFile, ACertSubjectName, AReason, ALocation)`은 편집한 문서를 Windows 저장소의 인증서로 한 번의 호출로 저장하고 서명합니다.

참고 항목

- 암호화
- PDF/A

PDF/A

PDF/A는 장기 보관을 위한 ISO 표준입니다. sgcPDF는 PDF/A-1, PDF/A-2, PDF/A-3의 **b**(기본) 규격 레벨을 작성하며, **TsgcPDFALevel**(유닛 sgcPDF_PDFA)의 paA1b, paA2b, paA3b로 선택합니다.
`SetPDFALevel`을 설정하면, 규격 레벨을 선언하고 Info 필드를 반복하는 XMP 메타데이터, sRGB 출력 의도, 문서 ID와 함께 문서가 저장됩니다. 콘텐츠도 PDF/A 규칙을 따라야 합니다.

- 모든 폰트를 임베드하십시오. 표준 폰트가 아니라 TrueType 폰트(AddTrueTypeFont와 DrawTextTT)로 텍스트를 그리십시오.
- 문서를 암호화하지 마십시오.
- PDF/A-1b는 투명도를 허용하지 않으므로, 이 레벨에서는 SetOpacity와 투명 그리기를 피하십시오.

```
static void CreatePDFA(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.SetPDFALevel(TsgcPDFALevel.paA2b);
    oPDF.Info.Title = "Archived invoice";
    oPDF.Info.Author = "eSeGeCe";
    // PDF/A requires embedded fonts, so use a TrueType font
    int vFont = oPDF.AddTrueTypeFont("Arial", false, false);
    oPDF.AddPage(595, 842).DrawTextTT("PDF/A-2b document", 50, 780, vFont, 20);
    oPDF.SaveToFile(AFileName);
  }
}
```

검증

TsgcPDFAValidator는 로드된 문서를 레벨에 맞춰 검사합니다. XMP 메타데이터, 폰트 임베드, 색 공간, 투명도, 암호화, 주석, 액션, 파일 구조입니다. `Errors`에 발견된 문제가 나열됩니다. 이는 빠른 자체 검사이며 veraPDF 같은 완전한 규격 검사기를 대체하지는 않습니다.

```
static void ValidatePDFA(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFAValidator oValidator = new TsgcPDFAValidator(TsgcPDFALevel.paA2b);
    oPDF.LoadFromFile(AFileName);
    if (oValidator.Validate(oPDF.Parser))
      Console.WriteLine("No PDF/A-2b problems found");
    else
      Console.WriteLine(oValidator.Errors.Text);
  }
}
```

`TsgcPDFEditor.ConvertToPDFA(ALevel)`은 에디터로 연 문서에 레벨을 설정합니다.

참고 항목

- TsgcPDFDocument
- 텍스트와 폰트

PDF 파일 최적화

TsgcPDFOptimizer는 PDF를 다시 작성하여 크기를 줄입니다. 기본적으로 스트림을 압축하고, 중복 객체를 병합하고, 사용하지 않는 객체를 제거합니다. 이미지 다운샘플링과 메타데이터, 북마크, 주석, JavaScript, 썸네일 제거는 `Options`를 통해 선택적으로 사용할 수 있습니다.

```
static void OptimizePDF(string AInput, string AOutput)
{
  using (TsgcPDFOptimizer oOptimizer = new TsgcPDFOptimizer())
  {
    TsgcPDFOptimizeOptions oOptions = sgcPDF_Optimize.sgcPDFDefaultOptimizeOptions();
    oOptions.DownsampleImages = true;
    oOptions.MaxImageDPI = 150;
    oOptions.RemoveMetadata = true;
    oOptimizer.Options = oOptions;
    oOptimizer.Optimize(AInput, AOutput);
    Console.WriteLine("Saved " + oOptimizer.GetSizeReduction() + " bytes");
    Console.WriteLine(oOptimizer.GetLog().Text);
  }
}
```

sgcPDF 자체로 작성한 파일처럼 이미 압축된 파일은 더 작아지지 않을 수 있습니다. 이때 GetSizeReduction은 0을 반환하며, 로그에는 입력과 출력 크기가 기록됩니다.

참고 항목

- 이미지

페이지 병합, 분할 및 추출

한 번의 호출로 사용하는 헬퍼

유닛 sgcPDF_Editor에는 파일에서 직접 동작하는 함수가 있습니다.

- `sgcPDFMergeFiles(AInputFiles, AOutputFile)`은 지정한 순서대로 파일을 병합합니다.
- `sgcPDFExtractPages(AInputFile, AOutputFile, AFromPage, AToPage)`는 0 기반의 포함 범위 페이지를 새 파일로 복사합니다(분할).
- `sgcPDFGetPageCount(AFileName)`은 페이지 수를 반환합니다.

```
static void MergeAndSplit()
{
  // merge several files into one
  sgcPDF_Editor.sgcPDFMergeFiles(new string[] { @"out\hello.pdf", @"out\bookmarks.pdf", @"out\tables.pdf" },
    @"out\merged.pdf");
  Console.WriteLine("Merged pages: " + sgcPDF_Editor.sgcPDFGetPageCount(@"out\merged.pdf"));
  // extract pages 2 to 4 (zero based indexes 1 to 3) into a new file
  sgcPDF_Editor.sgcPDFExtractPages(@"out\merged.pdf", @"out\pages_2_4.pdf", 1, 3);
}
```

에디터 사용하기

TsgcPDFEditor는 열린 문서에서 페이지 작업을 함께 제공합니다. MergeFrom, ExtractPages, DeletePage, RotatePage, AddBlankPage입니다.

```
static void ReorganizePages(string AInput, string AOutput)
{
  using (TsgcPDFEditor oEditor = new TsgcPDFEditor())
  {
    oEditor.LoadFromFile(AInput);
    oEditor.MergeFrom(@"out\hello.pdf");
    oEditor.RotatePage(0, 90);
    oEditor.DeletePage(1);
    oEditor.AddBlankPage(595, 842);
    oEditor.SaveToFile(AOutput);
  }
}
```

TsgcPDFDocument에서는 AppendDocument, InsertPage, DeletePage를 사용하십시오.

참고 항목

- TsgcPDFEditor

텍스트 추출

sgcPDF는 페이지 콘텐츠 스트림에서 텍스트를 추출합니다. 표준 인코딩(WinAnsi, MacRoman, Standard), 폰트 Differences, ToUnicode 맵, CID 폰트를 디코딩하므로, 임베드된 TrueType이나 CJK 폰트로 작성된 텍스트도 유니코드로 반환됩니다. 이미지만 담고 있는 스캔 페이지는 추출할 텍스트가 없습니다.

일반 텍스트

`TsgcPDFDocument.ExtractTextW`는 페이지의 텍스트를 반환하고, `FindText`는 그 텍스트를 검색합니다.

```
static void PrintAllText(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    for (int i = 0; i < oPDF.PageCount(); i++)
    {
      Console.WriteLine("--- page " + (i + 1) + " ---");
      Console.WriteLine(oPDF.ExtractTextW(i));
    }
    if (oPDF.FindText(0, "Hello"))
      Console.WriteLine("Page 1 contains \"Hello\"");
  }
}
```

위치가 포함된 텍스트

TsgcPDFTextExtractor는 모든 텍스트 조각을 좌표, 폰트, 크기와 함께 반환합니다.

```
static void PrintTextPositions(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    using (TsgcPDFTextExtractor oExtractor = new TsgcPDFTextExtractor(oPDF.Parser,
      oPDF.GetPage(0).PageDict))
    {
      oExtractor.Extract();
      for (int i = 0; i < oExtractor.GetFragmentCount(); i++)
      {
        TsgcPDFTextFragment oFragment = oExtractor.GetFragment(i);
        Console.WriteLine(string.Format("{0:0.0}, {1:0.0}  {2}  ({3} {4:0.0} pt)", oFragment.X,
          oFragment.Y, oFragment.Text, oFragment.FontName, oFragment.FontSize));
      }
    }
  }
}
```

파서 직접 사용하기

sgcPDFExtractPageText, sgcPDFExtractAllText, sgcPDFFindText 함수는 TsgcPDFParser와 함께 동작합니다. 로드한 다음 `Parse`를 호출하십시오.

```
static string GetDocumentText(string AFileName)
{
  using (TsgcPDFParser oParser = new TsgcPDFParser())
  {
    oParser.LoadFromFile(AFileName);
    oParser.Parse();
```

```
    return sgcPDF_TextExtract.sgcPDFExtractAllText(oParser);
  }
}
```

참고 항목

- TsgcPDFTextExtractor

렌더링과 내보내기

페이지는 **TsgcPDFRendererGDI**(유닛 sgcPDF_Renderer_GDI)로 렌더링되며, 이는 페이지 콘텐츠(경로, 텍스트, 이미지, 셰이딩)를 Windows 디바이스 컨텍스트에 그립니다. 직접 사용하는 경우는 드뭅니다.

- TsgcPDFExporter는 페이지를 BMP, PNG, JPEG 파일이나 썸네일로 저장합니다.
- TsgcPDFViewer는 창에 페이지를 표시합니다.
- `TsgcPDFRendererGDI.RenderPage(ADC, AParser, APageDict, AWidth, AHeight)`는 TBitmap 캔버스나 프린터 같은 임의의 HDC에 페이지를 그립니다.

페이지 내보내기

```
static void ExportPages(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  using (TsgcPDFExporter oExporter = new TsgcPDFExporter())
  {
    oPDF.LoadFromFile(AFileName);
    TsgcPDFExportOptions oOptions = sgcPDF_Export.sgcPDFDefaultExportOptions();
    oOptions.Format = TsgcPDFExportFormat.efPNG;
    oOptions.DPI = 150;
    oExporter.ExportPage(oPDF, 0, @"out\page1.png", oOptions);
    oExporter.ExportThumbnail(oPDF, 0, @"out\thumb1.bmp", 200, 200);
  }
  // one call helpers that open the file themselves
  sgcPDF_Export.sgcPDFPageToImage(AFileName, @"out\page1_96.png", 0, 96);
}
```

보기

```
static void ShowViewer(string AFileName)
{
  using (TsgcPDFViewer oViewer = new TsgcPDFViewer())
  {
    oViewer.ViewMode = TsgcPDFViewerMode.vmContinuous;
    // top level window; use CreateWindow(AParent, X, Y, W, H) to embed it
    oViewer.CreateWindow(100, 100, 800, 900, "My PDF Viewer");
    oViewer.LoadFromFile(AFileName);
    oViewer.ZoomFit();
    oViewer.Show();
    oViewer.RunMessageLoop();
  }
}
```

참고 항목

- TsgcPDFExporter
- TsgcPDFViewer

오류 처리 및 악의적인 입력

sgcPDF는 예외를 발생시켜 오류를 알립니다. 손상되었거나 지원되지 않는 파일을 로드하거나, 잘못된 페이지 인덱스, 없는 인증서, 지원되지 않는 이미지를 사용하면 문제를 설명하는 Message를 가진 예외가 발생합니다(예: *PDF parse error: cannot find startxref* 또는 *sgcPDF: Invalid page index*). 암호화 오류는 **EsgcPDFEncryptionError**를 발생시킵니다. 직접 제어하지 않는 파일을 로드할 때는 try..except로 감싸십시오.

```
static void OpenUntrustedPDF(string AFileName)
{
  // cap every decoded stream at 64 MB instead of the default 256 MB
  sgcPDF_Filters.sgcPDFMaxDecodedSize = 64 * 1024 * 1024;
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    try
    {
      oPDF.LoadFromFile(AFileName);
      Console.WriteLine("Pages: " + oPDF.PageCount());
    }
    catch (Exception E)
    {
      Console.WriteLine("Rejected: " + E.Message);
    }
  }
}
```

신뢰할 수 없는 파일에 대한 제한

PDF 파일은 메모리나 스택을 고갈시키도록 조작될 수 있습니다. 예를 들어 압축을 풀면 기가바이트가 되는 작은 압축 스트림이나, 깊게 중첩된 객체가 그렇습니다. 파서와 디코더는 제한을 적용하며, 초과하면 예외를 발생시킵니다.

제한	값	설명
sgcPDFMaxDecodedSize	268435456 (256 MB)	하나의 스트림(Flate, LZW, ASCII85, ASCIIHex, RunLength)의 최대 디코딩 크기입니다. sgcPDF_Filters의 전역 변수이므로, 업로드를 처리하는 서비스에서는 낮추고, 매우 큰 정상 이미지에서는 높일 수 있습니다.
객체 중첩	512단계	파서에서 중첩된 배열과 딕셔너리입니다.
객체 수	8388607	상호 참조 데이터에서 허용하는 최대 객체 번호입니다.
상호 참조 섹션	4096	연결된 xref 섹션과 스트림입니다.
이미지 크기	2^{27}픽셀, 데이터 1GB	디코딩되는 최대 이미지 크기입니다.
콘텐츠 스트림 중첩	64단계	콘텐츠 스트림 파서가 허용하는 중첩 깊이입니다.
페이지 트리	64단계, 노드 1,000,000개	텍스트 추출기가 순회하는 페이지 트리의 깊이와 크기입니다.

변경할 수 있는 값은 sgcPDFMaxDecodedSize뿐입니다. 다른 제한은 고정되어 있습니다. sgcPDFMaxDecodedSize는 프로세스의 모든 문서에 적용되므로 시작 시 한 번만 설정하십시오.

참고 항목

- FAQ

렌더러 아키텍처

sgcPDF는 하나의 콘텐츠 해석기와 여러 그리기 백엔드로 PDF 페이지를 그립니다. 해석기는 페이지 콘텐츠 스트림을 읽고, 폰트, 이미지, 색 공간, 셰이딩, 주석을 해석하며, 모든 좌표를 디바이스 픽셀로 변환합니다. 백엔드는 경로, 래스터, 글리프만 그리면 됩니다. 따라서 같은 페이지는 어떤 백엔드에서도 동일하게 보입니다.

클래스	유닛	역할
TsgcPDFRenderer	sgcPDF_Renderer	콘텐츠 해석기입니다. 플랫폼 독립적이며 순수 Pascal입니다.
TsgcPDFRenderCanvas	sgcPDF_Renderer_Canvas	추상 그리기 백엔드입니다. 경로, 클리핑, 이미지, 래스터, 레이어, 시스템 텍스트입니다.
TsgcPDFRasterCanvas	sgcPDF_Renderer_Raster	안티앨리어싱을 지원하는 소프트웨어 래스터라이저입니다. BGRA 픽셀 버퍼에 그립니다. 모든 플랫폼에서 실행됩니다.
TsgcPDFGDICanvas	sgcPDF_Renderer_GDI	Windows GDI 백엔드입니다. 임의의 디바이스 컨텍스트(화면, 비트맵, 프린터)에 그립니다.
TsgcPDFFMXCanvas	sgcPDF_Renderer_FMX	FireMonkey 비트맵 출력과 FMX 시스템 텍스트를 사용하는 소프트웨어 래스터라이저입니다. Delphi XE5 이상이 필요합니다.

일반적으로 해석기를 직접 사용하지 않습니다. 각 백엔드에는 해석기와 캔버스를 소유하는 완성된 렌더러 클래스가 있습니다.

렌더러	출력	플랫폼
TsgcPDFRendererRaster	PNG 파일 또는 스트림, BMP 파일, BGRA 픽셀	Windows, Linux, macOS, iOS, Android
TsgcPDFRendererGDI	Windows 디바이스 컨텍스트, BMP 파일	Windows
TsgcPDFRendererFMX	FireMonkey TBitmap	모든 FMX 플랫폼

공통 속성

모든 렌더러 클래스는 해석기를 `Renderer`로, 백엔드를 `Canvas`로 노출하며, 다음 속성도 제공합니다.

속성	설명
RenderAnnotations: Boolean	페이지 콘텐츠 다음에 페이지 주석과 폼 필드를 그립니다. 기본값은 True입니다.
AnnotationMode: TsgcPDFAnnotRenderMode	**armView**(기본값)는 NoView 플래그가 있는 주석을 건너뛰고, **armPrint**는 Print 플래그가 있는 주석만 그립니다. 인쇄물과 일치해야 하는 인쇄나 이미지에는 armPrint를 사용하십시오. TsgcPDFRendererFMX에서는 사용할 수 없으며 항상 armView를 사용합니다.

자체 백엔드 작성하기

다른 대상(그래픽 라이브러리, PDF를 SVG로 변환하는 도구, 테스트 레코더)에 그리려면 **TsgcPDFRenderCanvas**에서 파생하여 추상 메서드를 구현한 다음 `TsgcPDFRenderer.Create(ACanvas)`에 전달하십시오. 그런 다음 `RenderPage(AParser,`

`APageDict, AWidth, AHeight)`를 호출하십시오. 해석기는 디바이스 공간에서 캔버스를 호출합니다. 좌표는 Y가 아래로 갈수록 증가하는 픽셀이고, 색상은 RGB 바이트이며, 알파는 0부터 255까지입니다. 래스터 버퍼는 32비트 BGRA이며, 위에서 아래로, 프리멀티플라이되지 않습니다.

참고 항목

- TsgcPDFRendererRaster
- TsgcPDFRendererGDI
- 지원하는 렌더링 기능
- 렌더링과 내보내기

TsgcPDFRendererRaster

TsgcPDFRendererRaster(유닛 sgcPDF_Renderer_Raster)는 GDI, VCL, FMX 없이 PDF 페이지를 이미지로 렌더링합니다. 순수 Pascal로 작성되어 동일한 코드가 Windows 서비스, Linux 웹 서버, macOS의 콘솔 도구에서 실행됩니다. 서버에서 페이지 미리보기와 썸네일을 만들거나, PDF를 PNG로 변환하거나, OCR이나 이미지 처리 코드에 페이지 픽셀을 전달할 때 사용하십시오.

메서드

메서드	설명
RenderPageToPNG(AParser; APageDict; AWidth, AHeight: Integer; const AFileName: string)	흰색 배경에 AWidth x AHeight 픽셀로 페이지를 렌더링하고 PNG로 저장합니다.
RenderPageToBitmap(AParser; APageDict; AWidth, AHeight: Integer; const AFileName: string)	동일하지만 32비트 BMP 파일로 저장합니다.
RenderPageToImage(AParser; APageDict; AWidth, AHeight: Integer)	캔버스에만 렌더링합니다. `Canvas.Bits` 에서 픽셀을 읽거나 캔버스 메서드로 저장하십시오.
RenderPage(AParser; APageDict; ARenderWidth, ARenderHeight: Integer; AOffsetX: Integer = 0; AOffsetY: Integer = 0)	현재 캔버스 콘텐츠 위에 크기를 조정하거나 지우지 않고 그립니다. 먼저 `Canvas.SetSize` 로 캔버스 크기를 지정하십시오.
Render(AContentData: TBytes; AMediaBox: TsgcPDFRect; ...)	원시 콘텐츠 스트림을 렌더링합니다(리소스 없음).

AParser는 로드된 문서의 `TsgcPDFDocument.Parser`이고, APageDict는 `TsgcPDFDocument.Pages[i].PageDict`입니다. 페이지 회전(/Rotate)이 적용됩니다.

TsgcPDFRasterCanvas

`Canvas` 속성은 페이지 래스터를 담고 있습니다.

멤버	설명
Width, Height	래스터의 크기이며, 픽셀 단위입니다.
Bits: TBytes	픽셀입니다. BGRA, 위에서 아래로, 행당 4 * Width 바이트입니다. 알파 채널은 항상 255입니다.
SetSize(AWidth, AHeight)	래스터 크기를 조정하고 흰색으로 채웁니다.
Clear(R, G, B: Byte)	래스터를 색상으로 채웁니다.
SaveToPNGFile, SaveToPNGStream, SaveToBMPFile	래스터를 저장합니다.

예제: 페이지를 PNG로

```
static void RenderPageToPNG(string APDFFile, string APNGFile, int ADPI)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    TsgcPDFPage oPage = oPDF.GetPage(0);
    TsgcPDFRendererRaster oRenderer = new TsgcPDFRendererRaster();
    // page size in points (1/72 inch) to pixels
    oRenderer.RenderPageToPNG(oPDF.Parser, oPage.PageDict,
      (int)Math.Round(oPage.Width * ADPI / 72), (int)Math.Round(oPage.Height * ADPI / 72),
      APNGFile);
  }
}
```

예제: 모든 페이지, 픽셀, 스트림

```
static void RenderAllPagesToPixels(string APDFFile)
{
  // extra folder with TrueType fonts for text in non embedded fonts
  sgcPDF_Renderer_Raster.sgcPDFRasterAddFontDirectory("/opt/myapp/fonts");
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    TsgcPDFRendererRaster oRenderer = new TsgcPDFRendererRaster();
    // annotations are drawn by default; armPrint draws the printable ones
    oRenderer.AnnotationMode = TsgcPDFAnnotRenderMode.armPrint;
    for (int i = 0; i < oPDF.PageCount(); i++)
    {
      oRenderer.RenderPageToImage(oPDF.Parser, oPDF.GetPage(i).PageDict, 800, 1131);
      // Canvas.Bits: BGRA pixels, top-down, 4 * Width bytes per row
      Console.WriteLine(string.Format("page {0}: {1} x {2}, {3} bytes", i + 1,
        oRenderer.Canvas.Width, oRenderer.Canvas.Height, oRenderer.Canvas.Bits.Length));
      using (FileStream oStream = new FileStream(string.Format(@"out\page{0}.png", i + 1),
        FileMode.Create))
        oRenderer.Canvas.SaveToPNGStream(oStream);
    }
  }
}
```

임베드되지 않은 텍스트를 위한 폰트

임베드된 폰트(TrueType, OpenType, CFF, Type 1, Type 3)의 텍스트는 PDF 안의 폰트 프로그램으로 그려집니다. 표준 14가지 폰트처럼 임베드되지 않은 폰트의 텍스트는 같은 이름이나 비슷한 대체 폰트인 시스템의 TrueType 폰트로 그려집니다. 렌더러는 다음 폴더를 검사합니다.

- Windows: %WINDIR%\Fonts와 %LOCALAPPDATA%\Microsoft\Windows\Fonts입니다.
- Linux: /usr/share/fonts, /usr/local/share/fonts, ~/.fonts, ~/.local/share/fonts입니다.
- macOS: /System/Library/Fonts, /Library/Fonts, ~/Library/Fonts입니다.
- Android: /system/fonts입니다.

다른 폴더를 추가하려면 렌더링 전에 `sgcPDFRasterAddFontDirectory(const ADirectory: string)`를 호출하십시오(재귀적으로 검사됩니다). 폰트가 없는 최소 구성의 Linux 서버나 컨테이너에서는 폰트 패키지(예: fonts-dejavu나 fonts-liberation)를 설치하거나, 자체 폰트를 함께 배포하고 해당 폴더를 등록하십시오.

제한 사항

래스터 크기는 거대한 할당으로부터 프로세스를 보호하기 위해 CS_CANVAS_MAX_RASTER 픽셀(67,108,864, 예를 들어 8192 x 8192)로 제한됩니다. 더 큰 출력은 `RenderPage`와 오프셋 매개변수로 타일로 나누십시오.

참고 항목

- 렌더러 아키텍처
- 지원하는 렌더링 기능
- Linux
- TsgcPDFRendererGDI

TsgcPDFRendererGDI

TsgcPDFRendererGDI(유닛 sgcPDF_Renderer_GDI)는 Windows 디바이스 컨텍스트에 PDF 페이지를 렌더링합니다. 래스터 렌더러와 동일한 콘텐츠 해석기를 사용하며 **TsgcPDFGDICanvas**를 통해 그립니다. TsgcPDFViewer, TsgcPDFViewerControl, TsgcPDFThumbnailView, TsgcPDFExporter, 인쇄 기능이 이를 기반으로 합니다. Windows 전용입니다.

메서드

메서드	설명
RenderPage(ADC: HDC; AParser; APageDict; ARenderWidth, ARenderHeight: Integer; AOffsetX: Integer = 0; AOffsetY: Integer = 0)	폼이나 컨트롤 캔버스(`Canvas.Handle`), 메모리 비트맵, 프린터 같은 디바이스 컨텍스트에 페이지를 그립니다.
Render(ADC: HDC; AContentData: TBytes; AMediaBox: TsgcPDFRect; ...)	원시 콘텐츠 스트림을 그립니다.
RenderPageToBitmap(AParser; APageDict; AWidth, AHeight: Integer; const AFileName: string)	페이지를 BMP 파일로 렌더링합니다.
RenderToBitmap(AContentData: TBytes; AMediaBox: TsgcPDFRect; AWidth, AHeight: Integer; const AFileName: string)	원시 콘텐츠 스트림을 BMP 파일로 렌더링합니다.

속성: `Renderer`, `Canvas`, `RenderAnnotations`, `AnnotationMode`입니다. 렌더러 아키텍처를 참조하십시오.

예제: 페이지를 비트맵 파일로

```
static void RenderPageWithGDI(string APDFFile, string ABMPFile)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    using (TsgcPDFRendererGDI oRenderer = new TsgcPDFRendererGDI())
    {
      // skip annotations and form fields, draw the page content only
      oRenderer.RenderAnnotations = false;
      oRenderer.RenderPageToBitmap(oPDF.Parser, oPDF.GetPage(0).PageDict,
        1240, 1754, ABMPFile);
    }
  }
}
```

예제: 디바이스 컨텍스트에 그리기

```
static void PaintPage(IntPtr ADC, TsgcPDFDocument APDF, int APageIndex, int AWidth,
  int AHeight)
{
  using (TsgcPDFRendererGDI oRenderer = new TsgcPDFRendererGDI())
  {
    // draws on any device context: a form canvas, a bitmap or a printer
    oRenderer.RenderPage(ADC, APDF.Parser, APDF.GetPage(APageIndex).PageDict,
      AWidth, AHeight);
  }
```

```
}
```

참고 항목

- 렌더러 아키텍처
- TsgcPDFRendererRaster
- TsgcPDFViewerControl

지원하는 렌더링 기능

이 페이지는 콘텐츠 해석기가 그리는 내용을 나열합니다. 모든 렌더러(래스터, GDI, FMX)와 이를 기반으로 만든 뷰어에 적용됩니다.

그래픽

- 경로: 선, 베지어 곡선, 사각형, nonzero와 even-odd 규칙을 사용한 채우기, 선 그리기, 클리핑을 지원합니다.
- 선 너비, 캡, 조인, 마이터 제한, 대시 패턴을 지원합니다.
- 모든 색 공간에 대한 색상 연산자(cs, CS, sc, SC, scn, SCN, g, rg, k와 그 선 그리기 버전)를 지원합니다.
- 일정한 채우기와 선 알파(ExtGState /ca와 /CA)를 지원합니다.
- 블렌드 모드: Normal과 Multiply는 정확히 합성됩니다. 다른 블렌드 모드는 Normal로 그려집니다.
- 셰이딩: 함수 기반(타입 1), 축 방향(타입 2), 방사형(타입 3), 자유형과 격자형 구로 메시(타입 4와 5), Coons와 텐서곱 패치(타입 6과 7)를 지원하며, sh 연산자와 셰이딩 패턴으로 사용됩니다.
- 채색 및 비채색 타일링 패턴을 지원합니다.
- 매트릭스와 경계 상자를 가진 폼 XObject를 지원합니다.

색 공간

DeviceGray, DeviceRGB, DeviceCMYK, CalGray, CalRGB, Lab, ICCBased(대체 색 공간이나 구성 요소 수를 통해), Indexed, Separation, DeviceN을 지원하며, 샘플형(타입 0), 지수형(타입 2), PostScript 계산기형(타입 4) 색조 변환 함수를 지원합니다. CMYK는 ICC 색상 관리 없이 간단한 공식으로 RGB로 변환됩니다.

이미지

- 이미지 XObject와 인라인 이미지(BI, ID, EI)를 지원합니다.
- 필터: DCTDecode(JPEG 베이스라인 및 프로그레시브), JPXDecode(JPEG 2000), CCITTFaxDecode(Group 3과 Group 4), JBIG2Decode, 예측기가 있는 FlateDecode와 LZWDecode, RunLengthDecode, ASCII85Decode, ASCIIHexDecode를 지원합니다. 이미지 디코더를 참조하십시오.
- 구성 요소당 1, 2, 4, 8, 16비트, /Decode 배열, 이미지 마스크(스텐실 마스크), 색상 키 마스크, /Mask 스트림, /Matte가 있는 /SMask 소프트 마스크를 지원합니다.
- 회전, 기울이기, 뒤집기 상태에서의 배치와 부드러운 축소를 지원합니다.

텍스트와 폰트

- 임베드된 TrueType 폰트(FontFile2), CFF와 OpenType CFF 폰트(FontFile3), Type 1 폰트(FontFile), 그리고 Type 3 폰트(콘텐츠 스트림으로 그려지는 글리프)를 지원합니다.
- WinAnsi, MacRoman, Standard, /Differences 인코딩을 사용하는 단순 폰트, 심볼 폰트, Identity-H와 Identity-V, 사전 정의된 CJK CMap을 사용하는 합성(Type0) 폰트를 지원합니다.
- 세로쓰기(Identity-V와 세로 CMap, /W2 메트릭 포함)를 지원합니다.
- 문자 및 단어 간격, 가로 배율, 상승, 텍스트 렌더링 모드(채우기, 선 그리기, 클리핑, 숨김)를 지원합니다.
- 임베드되지 않은 폰트는 같은 이름의 설치된 시스템 폰트나 대체 폰트로 그려집니다. 작은 텍스트는 슈퍼샘플링되어 낮은 해상도에서도 읽을 수 있게 유지됩니다.

주석과 폼

- 주석 모양 스트림(/AP /N)을 지원하며, 확인란과 라디오 버튼의 /AS 상태, Hidden, NoView, Print 플래그, 모양 경계 상자를 주석 /Rect에 매핑하는 기능도 지원합니다.
- 모양 스트림이 없는 일반적인 주석 유형에 대한 간단한 대체 그리기를 지원합니다.
- 폼이 **/NeedAppearances**를 true로 설정하면, 일반 모양이 없는 텍스트 필드, 선택 필드, 버튼은 필드 값으로부터 그려집니다. `sgcPDFWidgetAppearanceGenerated(AParser, AWidget)`은 위젯이 이렇게 그려지는지 알려줍니다.
- 페이지 콘텐츠만 그리려면 `RenderAnnotations := False`로 설정하고, 프린터가 인쇄할 내용을 그리려면 `AnnotationMode := armPrint`로 설정하십시오.

지원하지 않는 기능

- ExtGState의 소프트 마스크 딕셔너리(/None이 아닌 /SMask)는 무시되며, 투명도 그룹은 분리되거나 녹아웃되지 않습니다.
- 선택적 콘텐츠(레이어)는 평가되지 않습니다. 숨겨진 레이어의 콘텐츠도 그려집니다.
- Normal과 Multiply 이외의 블렌드 모드는 Normal로 근사됩니다.

참고 항목

- 렌더러 아키텍처
- 이미지 디코더
- OpenType 폰트

TsgcPDFViewerControl

TsgcPDFViewerControl(유닛 sgcPDF_ViewerVCL)은 폼 안에 PDF 문서를 표시하는 VCL 컨트롤입니다. 폼에 놓거나 런타임에 만든 다음 파일을 로드합니다. 페이지는 TsgcPDFRendererGDI가 백그라운드에서 크기가 제한된 페이지 캐시로 렌더링하므로, 큰 문서도 빠르게 열리고 부드럽게 스크롤됩니다. **sgcPDF** 팔레트 페이지에 함께 쓰는 컨트롤인 TsgcPDFThumbnailView, TsgcPDFOutlineView, TsgcPDFCommentsView와 함께 있습니다.
기능: 연속 또는 단일 페이지 레이아웃, 확대/축소 모드, 키보드, 마우스 휠, 손 도구 탐색, 텍스트 선택과 복사, 강조 표시가 있는 검색, 링크 탐색, 대화형 폼 채우기, 주석 검토와 편집, 인쇄. Delphi 10.3 이상에서는 고DPI를 지원합니다.

공개 속성

속성	설명
Document: TsgcPDFDocument	직접 소유한 TsgcPDFDocument를 표시합니다. 컨트롤이 스스로 파일을 로드하게 하려면 비워 두십시오.
FileName: string	파일을 로드합니다(디자인 타임에도 가능).
PageIndex: Integer	현재 페이지이며, 0부터 시작합니다.
Zoom: Double	백분율로 표시한 확대/축소 배율입니다. 설정하면 ZoomMode가 zmCustom으로 전환됩니다.
ZoomMode: TsgcPDFViewerZoomMode	zmFitWidth(기본값), zmFitPage, zmActualSize 또는 zmCustom입니다.
ViewMode: TsgcPDFViewerViewMode	pvmContinuous(기본값, 모든 페이지를 아래로 이어서 표시) 또는 pvmSinglePage입니다.
PageGap: Integer	페이지 사이의 간격이며, 픽셀 단위입니다. 기본값은 10입니다.
BackgroundColor: TColor	페이지 주위의 색상입니다. 기본값은 clAppWorkSpace입니다.
ShowPageShadow: Boolean	모든 페이지 아래에 그림자를 그립니다. 기본값은 True입니다.
CacheSize: Integer	메모리에 유지하는 렌더링된 페이지 수입니다. 기본값은 12입니다.
SelectionMode: TsgcPDFViewerSelectionMode	psmText(기본값)는 마우스로 텍스트를 선택하고, psmHand는 페이지를 드래그합니다.
SelectionColor, SearchColor, SearchCurrentColor: TColor	텍스트 선택, 모든 검색 일치 항목, 현재 일치 항목의 색상입니다.
FormFilling: Boolean	사용자가 텍스트 필드, 확인란, 라디오 버튼, 목록 상자, 콤보 상자를 채울 수 있게 합니다. 기본값은 True입니다.
OpenLinksInBrowser: Boolean	http, https, mailto 링크를 기본 애플리케이션으로 엽니다. 기본값은 False입니다.
AnnotationEditing: Boolean	사용자가 주석을 선택, 이동, 크기 조정, 삭제하고, 텍스트를 편집하고, 새 주석을 추가할 수 있게 합니다. 이때 링크는 열리지 않고 선택됩니다. False로 설정하면 선택이 지워지지만 도구는 유지됩니다. AnnotationEditing이 False일 때도 도구가 동작하기 때문입니다. 기본값은 False입니다.
AnnotationTool: TsgcPDFViewerAnnotationTool	페이지에 놓을 다음 주석의 도구이며, AnnotationEditing이 True이든 False이든 적용됩니다. vatNone(기본값), vatNote, vatFreeText, vatHighlight,

속성	설명
	vatUnderline, vatStrikeOut, vatLine, vatArrow, vatSquare, vatCircle, vatInk, vatStamp.
AnnotationAuthor: string	사용자가 추가하는 주석과 답글의 작성자.
AnnotationColor: TColor	새 주석의 색상. clDefault(기본값)이면 각 도구의 일반 색상을 사용합니다.
AnnotationStampName: string	vatStamp로 놓는 스탬프의 이름. Approved(기본값), Draft, Confidential 등입니다.
ShowAnnotationPopups: Boolean	내용 또는 답글이 있는 주석을 클릭하면 팝업을 엽니다. 기본값은 True입니다.

이벤트

이벤트	설명
OnPageChange: TNotifyEvent	현재 페이지가 변경되었습니다.
OnZoomChange: TNotifyEvent	확대/축소 배율이 변경되었습니다.
OnLinkClick(Sender; APageIndex: Integer; const AURI: string; var AHandled: Boolean)	링크를 클릭했습니다. 문서의 페이지로 연결되는 링크는 AURI가 비어 있습니다. 기본 동작을 취소하려면 AHandled를 True로 설정하십시오.
OnFieldChange(Sender; AField: TsgcPDFFormField)	사용자가 폼 필드의 값을 변경했습니다.
OnSearchProgress(Sender; APageIndex, APageCount: Integer; var ACancel: Boolean)	FindText가 검색하는 동안 모든 페이지에 대해 호출됩니다. 중단하려면 ACancel을 설정하십시오.
OnAnnotationClick(Sender; AAnnotation: TsgcPDFAnnotation; var AHandled: Boolean)	내용 또는 답글이 있는 주석을 클릭했습니다. AHandled를 True로 설정하면 팝업이 표시되지 않습니다.
OnAnnotationSelect(Sender; AAnnotation: TsgcPDFAnnotation)	선택된 주석이 변경되었습니다. 선택이 지워진 경우 AAnnotation은 nil입니다.
OnAnnotationAdd(Sender; AAnnotation: TsgcPDFAnnotation)	사용자가 도구로 주석을 추가했거나 팝업에서 답글을 추가했습니다.
OnAnnotationChange(Sender; AAnnotation: TsgcPDFAnnotation)	사용자가 주석을 이동, 크기 조정 또는 편집했습니다.
OnAnnotationDelete(Sender; AAnnotation: TsgcPDFAnnotation; var AAllow: Boolean)	주석이 답글과 함께 삭제되려 하고 있습니다. AAllow를 False로 설정하면 삭제되지 않습니다.

공개 메서드

메서드	설명
LoadFromFile, LoadFromStream, Close	문서를 열고 닫습니다.
GotoPage, NextPage, PrevPage, FirstPage, LastPage	탐색입니다.
ZoomIn, ZoomOut	확대/축소를 한 단계씩 변경합니다.

메서드	설명
Print(AFromPage: Integer = 0; AToPage: Integer = -1; const ATitle: string = '')	현재 프린터로 페이지 범위(0 기반, -1은 마지막 페이지)를 인쇄하며, 주석은 인쇄 모드로 처리합니다.
RenderPageToBitmap(APageIndex: Integer; ABitmap: TBitmap; AZoomPercent: Double = 100; ADPI: Integer = 0)	페이지를 VCL 비트맵으로 렌더링합니다.
FindText(const AText: WideString; ACaseSensitive: Boolean = False; AWholeWord: Boolean = False; ABackward: Boolean = False): Boolean	문서에서 모든 일치 항목을 찾아 강조 표시하고 첫 번째 항목으로 스크롤합니다.
FindNext, FindPrevious, ClearSearch, GetSearchMatch(AIndex)	일치 항목 사이를 이동합니다. SearchResults는 일치 항목 수이고, SearchIndex는 현재 항목입니다.
SelectText(APageIndex, AFirstChar, ACharCount), SelectPage, SelectAll, ClearSelection, CopyToClipboard	텍스트 선택입니다. SelectedText와 HasSelection으로 선택 항목을 읽습니다.
PageText(APageIndex): TsgcPDFTextExtractor	문자 위치가 포함된 페이지의 텍스트입니다(뷰어가 소유).
DeviceToPage, PageToDevice, GetPageRect, PageRotation	클라이언트 픽셀과 PDF 사용자 공간 사이를 변환합니다.
WidgetCount, GetWidget(AIndex), EditWidget	폼 필드 위젯에 접근하고 편집을 시작합니다.
ClearCache, RenderVisiblePages, CachedPageCount	페이지 캐시 제어입니다.
AnnotationAt(X, Y: Integer): TsgcPDFAnnotation	클라이언트 좌표의 한 점에 있는 가장 위쪽 주석이며, 없으면 nil입니다. 답글, 팝업, 폼 위젯, 숨겨진 주석은 건너뛰며, 링크는 AnnotationEditing이 True일 때만 반환됩니다.
SelectedAnnotation, SelectAnnotation(AAnnotation), DeleteSelectedAnnotation: Boolean	선택된 주석. SelectAnnotation(nil)은 선택을 지웁니다. DeleteSelectedAnnotation은 AnnotationEditing이 True일 때만 동작하고, OnAnnotationDelete를 발생시키며, 주석이 삭제되면 True를 반환합니다.
ShowAnnotationPopup(AAnnotation), CloseAnnotationPopup	주석의 팝업을 코드에서 열고 닫습니다.
GotoAnnotation(AAnnotation)	주석이 보이도록 스크롤합니다. AnnotationEditing이 True이면 그 주석도 선택됩니다.
AnnotationsChanged	코드에서 Document.Annotations를 변경한 뒤에 호출합니다. 뷰어는 다시 그리고, 제거된 주석의 선택 또는 팝업을 버립니다.

마우스와 키보드

- 마우스 휠은 스크롤하고, Ctrl + 휠은 확대/축소합니다.
- 화살표 키, Page Up, Page Down, Space, Home, End는 스크롤하고 페이지를 변경합니다.
- Ctrl+A는 문서의 모든 텍스트를 선택하고, Ctrl+C와 Ctrl+Insert는 선택 항목을 복사합니다.
- 폼 채우기 중에는 Tab으로 폼 필드 사이를 이동합니다.
- 내용 또는 답글이 있는 주석을 클릭하면 팝업이 열립니다. AnnotationEditing이 True일 때는 클릭하면 주석이 선택되고, 끌면 이동하고, 핸들로 크기를 조정하고, Delete로 삭제하고, 화살표 키로 이동합니다(Shift를 누르면 10포인트).
- 도구, 이동, 크기 조정 드래그 중의 Esc는 해당 드래그만 취소합니다. 그 외에는 주석 팝업을 닫고, 선택을 지우고, 주석 도구를 취소합니다.

예제: 썸네일과 북마크가 있는 뷰어

```
public TMainForm()
{
  Width = 1100;
  Height = 800;
  // page thumbnails on the left
  FThumbnails = new TsgcPDFThumbnailView();
  FThumbnails.Parent = this;
  FThumbnails.Dock = DockStyle.Left;
  FThumbnails.Width = 160;
  FThumbnails.ThumbnailWidth = 120;
  // bookmarks (outline) tree on the right
  FOutline = new TsgcPDFOutlineView();
  FOutline.Parent = this;
  FOutline.Dock = DockStyle.Right;
  FOutline.Width = 220;
  // the viewer fills the rest of the form
  FViewer = new TsgcPDFViewerControl();
  FViewer.Parent = this;
  FViewer.Dock = DockStyle.Fill;
  FViewer.BringToFront();
  FViewer.ZoomMode = TsgcPDFViewerZoomMode.zmFitWidth;
  FViewer.ViewMode = TsgcPDFViewerViewMode.pvmContinuous;
  FViewer.OnPageChange = OnPageChangeEvent;
  FViewer.OnLinkClick = OnLinkClickEvent;
  FViewer.OnFieldChange = OnFieldChangeEvent;
  FViewer.OnSearchProgress = OnSearchProgressEvent;
  // link the side panels to the viewer
  FThumbnails.Viewer = FViewer;
  FOutline.Viewer = FViewer;
  FViewer.LoadFromFile("sample.pdf");
}
```

예제: 이벤트

```
void OnPageChangeEvent(object Sender)
{
  Text = string.Format("Page {0} of {1}, zoom {2:F0}%",
    FViewer.PageIndex + 1, FViewer.PageCount, FViewer.Zoom);
}

void OnLinkClickEvent(object Sender, int APageIndex, string AURI, ref bool AHandled)
{
  // AURI is empty for links to a page of the same document
  if (AURI != "")
    AHandled = MessageBox.Show("Open " + AURI + "?", Text, MessageBoxButtons.YesNo,
      MessageBoxIcon.Question) != DialogResult.Yes;
}

void OnFieldChangeEvent(object Sender, TsgcPDFFormField AField)
{
  // a form field was edited in the viewer: save the filled form
  FViewer.Document.SaveToFile("filled.pdf");
}

void OnSearchProgressEvent(object Sender, int APageIndex, int APageCount,
  ref bool ACancel)
{
  Text = string.Format("Searching page {0} of {1}", APageIndex + 1, APageCount);
}
```

예제: 검색과 복사

```
public void SearchAndCopy()
{
  // every match of the document is highlighted, the first one is shown
  if (FViewer.FindText("invoice", false, true))
  {
    MessageBox.Show(string.Format("{0} matches", FViewer.SearchResults));
    FViewer.FindNext();
  }
  // select the text of the current page and copy it to the clipboard
  FViewer.SelectPage(FViewer.PageIndex);
  if (FViewer.HasSelection)
    FViewer.CopyToClipboard();
}
```

예제: 인쇄

```
public void PrintPages()
{
  // pages 1 to 3 (zero based indexes) on the default printer
  FViewer.Print(0, 2, "My document");
}
```

주석

뷰어는 `Document.Annotations`의 주석을 그립니다. 내용 또는 답글이 있는 주석을 클릭하면 작성자, 날짜, 내용, 답글을 보여 주는 팝업이 열립니다. 자체 사용자 인터페이스를 표시하려면 OnAnnotationClick을 처리하거나 ShowAnnotationPopups를 False로 설정하십시오.
AnnotationEditing을 True로 설정하면 사용자가 주석을 변경할 수도 있습니다.

- 클릭하면 주석이 선택되고 핸들이 표시됩니다. 주석을 끌어서 이동하거나 핸들을 끌어서 크기를 조정합니다.
- Delete는 선택된 주석을 답글과 함께 삭제합니다. OnAnnotationDelete로 거부할 수 있습니다.
- 화살표 키는 선택된 주석을 1포인트, Shift를 누르면 10포인트 이동합니다.
- 팝업의 텍스트를 편집할 수 있으며, 답글 입력란에서 AnnotationAuthor를 작성자로 하는 답글을 추가할 수 있습니다.
- 도구, 이동, 크기 조정 드래그 중의 Esc는 해당 드래그만 취소합니다. 그 외에는 팝업을 닫고, 선택을 지우고, 현재 도구를 취소합니다.

AnnotationTool을 설정하면 새 주석 하나를 놓을 수 있습니다. 페이지를 클릭하거나 그 위를 끌면 AnnotationAuthor와 AnnotationColor로 주석이 만들어집니다. OnAnnotationAdd가 발생하고 도구는 vatNone으로 돌아갑니다.

도구	놓이는 주석
vatNone	도구 없음(기본값). 클릭하면 주석을 선택합니다.
vatNote	스티키 노트(TsgcPDFTextAnnotation).
vatFreeText	텍스트 상자(TsgcPDFFreeTextAnnotation).
vatHighlight, vatUnderline, vatStrikeOut	마우스로 선택한 텍스트 위의 텍스트 마크업(TsgcPDFHighlightAnnotation).
vatLine, vatArrow	선 또는 끝에 화살표가 있는 선(TsgcPDFLineAnnotation).
vatSquare, vatCircle	사각형 또는 타원(TsgcPDFShapeAnnotation).
vatInk	자유 곡선 그리기(TsgcPDFInkAnnotation).
vatStamp	AnnotationStampName 이름의 스탬프(TsgcPDFStampAnnotation).

저장은 애플리케이션이 담당합니다. 뷰어는 메모리의 문서만 변경하며, 저장할 내용이 있는지는 `Document.Annotations.Modified`로 알 수 있습니다. 변경 내용을 덧붙여 기존 서명을 유효하게 유지하려면

Document.SaveIncremental을, 전체 저장에는 Document.SaveToFile을 호출하십시오. 주석을 참고하십시오. 코드에서 Document.Annotations를 변경한 뒤에는 AnnotationsChanged를 호출하십시오. 주석을 코멘트 목록으로 뷰어 옆에 표시하려면 TsgcPDFCommentsView를 연결하십시오.

예제: 주석

```
void SetupAnnotations()
{
  FViewer.AnnotationEditing = true;
  FViewer.AnnotationAuthor = "Reviewer";
  // Color.Empty (the default) uses the usual colour of each tool
  FViewer.AnnotationColor = Color.Red;
  FViewer.AnnotationStampName = "Draft";
  FViewer.OnAnnotationSelect = OnAnnotationSelectEvent;
  FViewer.OnAnnotationAdd = OnAnnotationAddEvent;
  FViewer.OnAnnotationChange = OnAnnotationChangeEvent;
  FViewer.OnAnnotationDelete = OnAnnotationDeleteEvent;
  FViewer.OnAnnotationClick = OnAnnotationClickEvent;
}

void NoteButtonClick(object Sender, EventArgs e)
{
  // the next click on a page places a sticky note
  FViewer.AnnotationTool = TsgcPDFViewerAnnotationTool.vatNote;
}

void DeleteButtonClick(object Sender, EventArgs e)
{
  FViewer.DeleteSelectedAnnotation();
}

void SaveButtonClick(object Sender, EventArgs e)
{
  if (FViewer.Document.Annotations.Modified)
    // the original bytes are kept, and so are the signatures of the file
    FViewer.Document.SaveIncremental(Path.ChangeExtension(FViewer.FileName, null) + "_reviewed.pdf");
}

void OnAnnotationSelectEvent(object Sender, TsgcPDFAnnotation AAnnotation)
{
  // AAnnotation is null when the selection is cleared
  DeleteButton.Enabled = AAnnotation != null;
}

void OnAnnotationAddEvent(object Sender, TsgcPDFAnnotation AAnnotation)
{
  StatusLabel.Text = string.Format("{0} added on page {1}", AAnnotation.GetType().Name,
    AAnnotation.PageIndex + 1);
}

void OnAnnotationChangeEvent(object Sender, TsgcPDFAnnotation AAnnotation)
{
  StatusLabel.Text = "Changed: " + AAnnotation.Contents;
}

void OnAnnotationDeleteEvent(object Sender, TsgcPDFAnnotation AAnnotation, ref bool AAllow)
{
  AAllow = MessageBox.Show("Delete this annotation and its replies?", Text, MessageBoxButtons.YesNo,
    MessageBoxIcon.Question) == DialogResult.Yes;
}

void OnAnnotationClickEvent(object Sender, TsgcPDFAnnotation AAnnotation, ref bool AHandled)
{
  // no popup for stamps in this application
  AHandled = AAnnotation is TsgcPDFStampAnnotation;
```

```
}
```

폼 채우기

FormFilling이 True이면, 텍스트나 선택 필드를 클릭하면 필드 위에 에디터가 열리고, 확인란이나 라디오 버튼을 클릭하면 전환됩니다. 필드 값은 문서(TsgcPDFDocument.AcroForm)에 기록되고 페이지가 다시 그려집니다. 반응하려면 OnFieldChange를 처리하고, 채워진 값을 유지하려면 `Document.SaveToFile`로 문서를 저장하십시오.

참고 항목

- TsgcPDFThumbnailView
- TsgcPDFOutlineView
- TsgcPDFCommentsView
- TsgcPDFViewerFMX
- TsgcPDFViewer
- 주석

TsgcPDFThumbnailView

TsgcPDFThumbnailView(유닛 sgcPDF_ViewerVCL)는 TsgcPDFViewerControl의 페이지 썸네일을 스크롤 가능한 열로 표시합니다. 썸네일을 클릭하면 해당 페이지로 이동합니다. 현재 페이지는 강조 표시되며, 썸네일은 필요할 때 렌더링되어 캐시됩니다.

멤버	설명
Viewer: TsgcPDFViewerControl	문서가 표시되는 뷰어입니다. 썸네일은 이 뷰어의 문서와 현재 페이지를 따라갑니다.
ThumbnailWidth: Integer	썸네일의 너비이며, 픽셀 단위입니다. 기본값은 120입니다.
ItemCount	썸네일(페이지) 수입니다.
GetItemRect(AIndex): TRect, ItemAtPos(X, Y): Integer	클라이언트 좌표로 표현한 썸네일의 레이아웃입니다.
RenderVisibleThumbnails, CachedThumbnailCount	렌더링과 캐시 제어입니다.

일반적인 VCL 속성(Align, Color, Font, PopupMenu...)과 OnClick, OnMouseDown, OnResize가 공개되어 있습니다. TsgcPDFViewerControl의 예제를 참조하십시오.

참고 항목

- TsgcPDFViewerControl
- TsgcPDFOutlineView

TsgcPDFOutlineView

TsgcPDFOutlineView(유닛 sgcPDF_ViewerVCL)는 TsgcPDFViewerControl 문서의 북마크(개요)를 표시하는 트리 뷰입니다. 노드를 선택하면 뷰어가 북마크 대상으로 스크롤됩니다.

멤버	설명
Viewer: TsgcPDFViewerControl	따라갈 뷰어입니다. 뷰어가 다른 문서를 로드하면 트리가 다시 만들어집니다.
RebuildOutline	문서로부터 트리를 다시 로드합니다.
NavigateToNode(ANode: TTreeNode)	노드의 대상으로 이동합니다.

TCustomTreeView 에서 파생되며, 일반적인 트리 뷰 속성(Items, Indent, ShowButtons, ShowLines, ShowRoot, HideSelection...)과 OnClick, OnChange 이벤트를 공개합니다.

참고 항목

- TsgcPDFViewerControl
- TsgcPDFCommentsView
- 북마크

TsgcPDFCommentsView

TsgcPDFCommentsView(유닛 sgcPDF_ViewerVCL)는 TsgcPDFViewerControl의 문서에 있는 코멘트를 나열하는 트리 뷰로, PDF 리더의 코멘트 패널과 같습니다. 폼 필드, 링크, 팝업을 제외한 모든 주석이 나열됩니다.

멤버	설명
Viewer: TsgcPDFViewerControl	따라갈 뷰어. 뷰어가 문서를 로드하거나 닫을 때와 주석이 변경될 때 목록이 다시 만들어집니다.
GroupByPage: Boolean	코멘트를 페이지마다 하나의 노드로 묶어 Page N (count) 텍스트로 표시합니다. False이면 코멘트가 페이지 순서의 단일 목록이 됩니다. 기본값은 True입니다.
ShowReplies: Boolean	코멘트의 답글을 그 자식 노드로 표시합니다. False이면 최상위 코멘트만 나열됩니다. 기본값은 True입니다.
RebuildComments	뷰어의 문서에서 목록을 다시 로드합니다. 목록은 이미 뷰어를 따라가며, 코드에서 Document.Annotations를 변경한 뒤에는 뷰어의 AnnotationsChanged 메서드도 목록을 다시 만듭니다.
SelectedAnnotation: TsgcPDFAnnotation	읽기 전용. 선택된 노드의 주석이며, 선택된 노드가 없거나 페이지 노드가 선택되어 있으면 nil입니다.

각 노드는 `Author, date: contents` 형식으로 표시됩니다. 내용은 한 줄로 표시되며 긴 텍스트는 120자에서 잘립니다. 내용이 비어 있으면 대신 Highlight나 Stamp (Approved)처럼 주석의 유형이 표시됩니다. 작성자가 없는 주석은 (no author)로 표시되고, 날짜가 없는 주석은 날짜를 생략합니다.

- 노드를 선택하면 뷰어가 GotoAnnotation으로 해당 주석까지 스크롤합니다. 뷰어의 AnnotationEditing이 True이면 그 주석이 뷰어에서도 선택됩니다.
- 더블 클릭하면 해당 주석의 팝업이 열립니다.
- 뷰어의 AnnotationEditing이 True일 때 Delete 키는 선택된 주석과 그 답글을 삭제합니다. 뷰어의 OnAnnotationDelete로 거부할 수 있습니다.
- 사용자가 뷰어에서 주석을 선택하면 그 노드가 목록에서 선택됩니다.

TCustomTreeView 에 서 파 생 되 며 일 반 적 인 트 리 뷰 속 성(Items, Indent, ShowButtons, ShowLines, ShowRoot, HideSelection 등)과 OnClick, OnChange 이벤트를 게시합니다. 트리는 읽기 전용이며 HideSelection의 기본값은 False이므로 선택된 코멘트가 계속 보입니다.

예제: 코멘트 패널

```
void CreateCommentsPanel()
{
  FComments = new TsgcPDFCommentsView();
  FComments.Parent = this;
  FComments.Dock = DockStyle.Right;
  FComments.Width = 280;
  // one flat list instead of the "Page N (count)" groups
  FComments.GroupByPage = false;
  FComments.ShowReplies = true;
  // AfterSelect of the TreeView: OnChange in Delphi
  FComments.AfterSelect += OnCommentChangeEvent;
  // from now on the list follows the document, the annotations and the
  // selection of the viewer
  FComments.Viewer = FViewer;
  // a selected comment is also selected in the viewer, Delete removes it
  FViewer.AnnotationEditing = true;
}

void OnCommentChangeEvent(object Sender, TreeViewEventArgs e)
{
  // null for a page group node
```

```
  TsgcPDFAnnotation oAnnot = FComments.SelectedAnnotation;
  if (oAnnot != null)
    StatusLabel.Text = string.Format("{0}, page {1}", oAnnot.Author, oAnnot.PageIndex + 1);
}
```

참고 항목

- TsgcPDFViewerControl
- TsgcPDFOutlineView
- TsgcPDFCommentsViewFMX
- 주석

TsgcPDFViewerFMX

TsgcPDFViewerFMX(유닛 sgcPDF_ViewerFMX)는 Windows, macOS, iOS, Android, Linux(FMXLinux)에서 PDF 문서를 표시하는 FireMonkey 컨트롤입니다. 페이지는 순수 Pascal 소프트웨어 래스터라이저(FMX 비트맵 출력을 사용하는 TsgcPDFRendererRaster)로 연속 세로 레이아웃에 그려지며, 제한된 페이지 캐시에 유지됩니다. Delphi XE5 이상이 필요하며, FMX 패키지로 설치되어 **sgcPDF** 팔레트 페이지에 나타납니다.

공개 속성

속성	설명
Document: TsgcPDFDocument	직접 소유한 TsgcPDFDocument를 표시합니다.
PageIndex: Integer	현재 페이지이며, 0부터 시작합니다.
Zoom: Single	확대/축소 비율이며, 1은 100%입니다. 설정하면 ZoomMode가 fzmCustom으로 전환됩니다.
ZoomMode: TsgcPDFViewerFMXZoomMode	fzmFitWidth(기본값), fzmFitPage 또는 fzmCustom입니다.
PageGap: Single	페이지 사이의 간격입니다.
BackgroundColor: TAlphaColor	페이지 주위의 색상입니다. 기본값은 $FF808080입니다.
CacheSize: Integer	메모리에 유지하는 렌더링된 페이지 수입니다. 기본값은 8입니다.
OnPageChange, OnZoomChange, OnLayerChange: TNotifyEvent	페이지, 확대/축소, 레이어 표시 알림.
OnLinkClick(Sender; APageIndex: Integer; const AURI: string; var AHandled: Boolean)	링크를 탭하거나 클릭했습니다. AHandled를 설정하지 않으면 문서의 페이지로 연결되는 링크를 따라갑니다.
AnnotationEditing: Boolean	사용자가 주석을 선택, 이동, 크기 조정, 삭제하고, 텍스트를 편집하고, 새 주석을 추가할 수 있게 합니다. False로 설정하면 선택이 지워지지만 도구는 유지됩니다. AnnotationEditing이 False일 때도 도구가 동작하기 때문입니다. 열려 있는 팝업은 새 모드로 다시 열립니다. 기본값은 False입니다.
AnnotationTool: TsgcPDFViewerAnnotationTool	페이지에 놓을 다음 주석의 도구이며, AnnotationEditing이 True이든 False이든 적용됩니다. vatNone(기본값), vatNote, vatFreeText, vatHighlight, vatUnderline, vatStrikeOut, vatLine, vatArrow, vatSquare, vatCircle, vatInk, vatStamp.
AnnotationAuthor: string	사용자가 추가하는 주석과 답글의 작성자.
AnnotationColor: TAlphaColor	새 주석의 색상. TAlphaColorRec.Null(기본값)이면 각 도구의 일반 색상을 사용합니다.
AnnotationStampName: string	vatStamp로 놓는 스탬프의 이름. 기본값은 Approved입니다.
ShowAnnotationPopups: Boolean	내용 또는 답글이 있는 주석을 탭하거나 클릭하면 팝업을 엽니다. 기본값은 True입니다.

속성	설명
OnAnnotationClick(Sender; AAnnotation: TsgcPDFAnnotation; var AHandled: Boolean)	내용 또는 답글이 있는 주석을 탭하거나 클릭했습니다. AHandled를 True로 설정하면 팝업이 표시되지 않습니다.
OnAnnotationSelect(Sender; AAnnotation: TsgcPDFAnnotation)	선택된 주석이 변경되었습니다. 선택이 지워진 경우 AAnnotation은 nil입니다.
OnAnnotationAdd(Sender; AAnnotation: TsgcPDFAnnotation)	사용자가 도구로 주석을 추가했거나 팝업에서 답글을 추가했습니다.
OnAnnotationChange(Sender; AAnnotation: TsgcPDFAnnotation)	사용자가 주석을 이동, 크기 조정 또는 편집했습니다.
OnAnnotationDelete(Sender; AAnnotation: TsgcPDFAnnotation; var AAllow: Boolean)	주석이 답글과 함께 삭제되려 하고 있습니다. AAllow를 False로 설정하면 삭제되지 않습니다.

공개 메서드

LoadFromFile, LoadFromStream, Close, GotoPage, NextPage, PrevPage, FirstPage, LastPage, ZoomIn, ZoomOut, ClearCache, SetScrollPosition, GetPageRect, DeviceToPage, PageToDevice, PageRotation, CachedPageCount. 레이어(선택적 콘텐츠): GetLayers, SetLayerVisible, LayerVisible. TsgcPDFViewerControl과 같습니다. 읽기 전용: PageCount, ScrollX, ScrollY, ContentWidth, ContentHeight, FileName.
주석: AnnotationAt(X, Y: Single), SelectedAnnotation(읽기 및 쓰기), SelectAnnotation, DeleteSelectedAnnotation(주석이 삭제되면 True를 반환), ShowAnnotationPopup, CloseAnnotationPopup, GotoAnnotation, AnnotationsChanged. AnnotationsChanged는 코드에서 Document.Annotations를 변경한 뒤에 호출합니다. 이들은 TsgcPDFViewerControl과 같은 방식으로 동작합니다. DeleteSelectedAnnotation은 AnnotationEditing이 True일 때만 동작하고, GotoAnnotation은 주석이 보이도록 스크롤하며 AnnotationEditing이 True이면 그 주석도 선택합니다.

터치, 마우스, 키보드

- 손가락이나 마우스로 드래그하여 스크롤하고 이동합니다.
- 핀치로 확대/축소합니다(컨트롤이 Zoom 대화형 제스처를 활성화합니다).
- 마우스 휠은 스크롤하고, Shift + 휠은 가로로 스크롤하고, Ctrl + 휠은 확대/축소합니다.
- Page Up, Page Down, Home, End, 화살표 키로 탐색합니다.
- AnnotationEditing이 False일 때는 링크를 탭하면 해당 링크로 이동하고, 내용 또는 답글이 있는 주석을 탭하면 팝업이 열립니다.
- AnnotationEditing이 True일 때는 탭하면 주석이 선택되고, 빈 곳을 탭하면 선택이 지워집니다. 더블 클릭하면 팝업이 열립니다. 선택된 주석을 끌면 이동하고, 핸들을 끌면 크기가 조정되며, 그 밖의 곳을 끌면 화면이 이동합니다. Delete는 선택된 주석을 삭제하고, 화살표 키는 1포인트, Shift를 누르면 10포인트 이동합니다.
- 도구, 이동, 크기 조정 드래그 중의 Esc는 해당 드래그만 취소합니다. 그 외에는 주석 팝업을 닫고, 선택을 지우고, 주석 도구를 취소합니다.

예제

```
// .NET has no FireMonkey viewer: the WinForms TsgcPDFViewerControl is used
public TMainForm()
{
  FViewer = new TsgcPDFViewerControl();
  FViewer.Parent = this;
  FViewer.Dock = DockStyle.Fill;
  FViewer.ZoomMode = TsgcPDFViewerZoomMode.zmFitWidth;
  FViewer.BackColor = Color.LightGray;
  FViewer.OnPageChange = OnPageChangeEvent;
  FViewer.LoadFromFile("sample.pdf");
}

void OnPageChangeEvent(object Sender)
```

```
{
  Text = string.Format("Page {0} of {1}", FViewer.PageIndex + 1, FViewer.PageCount);
}
```

주석

FMX 뷰어는 TsgcPDFViewerControl과 동일한 주석 기능을 같은 도구와 이벤트로 제공합니다. 내용 또는 답글이 있는 주석을 탭하거나 클릭하면 작성자, 날짜, 내용, 답글을 보여 주는 팝업이 열립니다. AnnotationEditing이 True일 때는 팝업의 텍스트를 편집할 수 있으며, 답글 입력란에서 AnnotationAuthor를 작성자로 하는 답글을 추가할 수 있습니다.

- AnnotationEditing이 True일 때는 탭하거나 클릭하면 주석이 선택되고, 더블 클릭하면 팝업이 열립니다. 선택된 주석을 끌면 이동하고, 핸들로 크기를 조정하며, Delete로 답글과 함께 삭제합니다. 화살표 키는 선택된 주석을 1포인트, Shift를 누르면 10포인트 이동합니다. 변경할 때마다 OnAnnotationChange가 발생합니다.
- 도구, 이동, 크기 조정 드래그 중의 Esc는 해당 드래그만 취소합니다. 그 외에는 팝업을 닫고, 선택을 지우고, 현재 도구를 취소합니다.
- AnnotationTool은 탭하거나 끌어서 주석 하나를 놓은 다음 vatNone으로 돌아갑니다. AnnotationEditing이 True이면 새 주석이 선택됩니다. 새 노트는 ShowAnnotationPopups가 True일 때 텍스트를 입력할 수 있도록 팝업을 엽니다. 새 자유 텍스트는 제자리 편집기를 엽니다. Ctrl+Enter 또는 편집기 바깥을 탭하면 입력한 텍스트로 주석이 추가됩니다. Esc를 누르거나 텍스트가 비어 있으면 버려집니다.
- FMX 뷰어에는 텍스트 선택이 없으므로, vatHighlight, vatUnderline, vatStrikeOut은 끌어서 만든 사각형 안의 텍스트를 한 줄에 사각형 하나씩 표시합니다. 사각형에 텍스트가 없으면 아무것도 추가되지 않으며 도구는 활성 상태로 유지됩니다.
- 위치는 컨트롤 좌표(Single)로 지정합니다. AnnotationAt(X, Y)와 PageToDevice(APageIndex, APDFX, APDFY): TPointF.

저장은 VCL 뷰어와 마찬가지로 애플리케이션이 담당합니다. `Document.Annotations.Modified`가 True일 때 `Document.SaveIncremental` 또는 `Document.SaveToFile`을 호출하십시오. 저장한 뒤에는 `Document.Annotations.ResetModified`로 그 플래그를 지우며, 다음 저장에서도 문서를 로드한 이후에 이루어진 모든 변경이 기록됩니다. 주석을 참고하십시오. 주석을 코멘트 목록으로 뷰어 옆에 표시하려면 TsgcPDFCommentsViewFMX를 연결하십시오.

예제: 주석

```
// .NET has no FireMonkey viewer: the WinForms TsgcPDFViewerControl is used
void SetupAnnotations()
{
  FViewer.AnnotationEditing = true;
  FViewer.AnnotationAuthor = "Reviewer";
  // Color.Empty (the default) uses the usual colour of each tool
  FViewer.AnnotationColor = Color.Red;
  FViewer.OnAnnotationAdd = OnAnnotationAddEvent;
  FViewer.OnAnnotationChange = OnAnnotationChangeEvent;
  FViewer.OnAnnotationDelete = OnAnnotationDeleteEvent;
}

void HighlightButtonClick(object Sender, EventArgs e)
{
  // select text with the mouse to highlight it
  FViewer.AnnotationTool = TsgcPDFViewerAnnotationTool.vatHighlight;
}

void SaveButtonClick(object Sender, EventArgs e)
{
  if (FViewer.Document.Annotations.Modified)
    FViewer.Document.SaveIncremental(Path.Combine(
      Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments), "reviewed.pdf"));
}

void OnAnnotationAddEvent(object Sender, TsgcPDFAnnotation AAnnotation)
{
  Text = string.Format("{0} added on page {1}", AAnnotation.GetType().Name,
    AAnnotation.PageIndex + 1);
}

void OnAnnotationChangeEvent(object Sender, TsgcPDFAnnotation AAnnotation)
```

```
{
  Text = "Changed: " + AAnnotation.Contents;
}

void OnAnnotationDeleteEvent(object Sender, TsgcPDFAnnotation AAnnotation, ref bool AAllow)
{
  // only the annotations of this user can be deleted
  AAllow = AAnnotation.Author == FViewer.AnnotationAuthor;
}
```

FMX 비트맵으로 렌더링하기

TsgcPDFRendererFMX(유닛 sgcPDF_Renderer_FMX)는 예를 들어 사용자 지정 미리보기나 페이지 이미지 공유를 위해 페이지를 FMX TBitmap으로 렌더링합니다. 임베드되지 않았고 TrueType 파일로도 사용할 수 없는 폰트의 텍스트는 플랫폼의 FMX 텍스트 레이아웃으로 그려집니다.

```
// .NET has no FireMonkey renderer: TsgcPDFRendererGDI draws on a System.Drawing bitmap
public static void RenderPageToBitmap(TsgcPDFDocument ADocument, int APageIndex,
  Bitmap ABitmap)
{
  using (TsgcPDFRendererGDI oRenderer = new TsgcPDFRendererGDI())
  using (Graphics oGraphics = Graphics.FromImage(ABitmap))
    oRenderer.RenderPage(oGraphics, ADocument.Parser,
      ADocument.GetPage(APageIndex).PageDict, ABitmap.Width, ABitmap.Height);
}
```

텍스트 선택, 검색, 폼 채우기는 VCL 뷰어에서만 사용할 수 있습니다.

참고 항목

- TsgcPDFViewerControl
- TsgcPDFCommentsViewFMX
- macOS, iOS, Android
- 렌더러 아키텍처
- 주석

TsgcPDFCommentsViewFMX

TsgcPDFCommentsViewFMX(유닛 sgcPDF_ViewerFMX)는 TsgcPDFCommentsView의 FireMonkey 버전입니다. TsgcPDFViewerFMX의 문서에 있는 코멘트를 나열하는 TTreeView입니다. 폼 필드, 링크, 팝업을 제외한 모든 주석이 나열됩니다. Delphi XE5 이상이 필요하며 FMX 패키지가 **sgcPDF** 팔레트 페이지에 설치합니다.

멤버	설명
Viewer: TsgcPDFViewerFMX	따라갈 뷰어. 뷰어가 문서를 로드하거나 닫을 때와 주석이 변경될 때 목록이 다시 만들어집니다.
GroupByPage: Boolean	코멘트를 페이지마다 하나의 항목으로 묶어 Page N (count) 텍스트로 표시합니다. False이면 코멘트가 페이지 순서의 단일 목록이 됩니다. 기본값은 True입니다.
ShowReplies: Boolean	코멘트의 답글을 그 자식 항목으로 표시합니다. False이면 최상위 코멘트만 나열됩니다. 기본값은 True입니다.
RebuildComments	뷰어의 문서에서 목록을 다시 로드합니다. 목록은 이미 뷰어를 따라가며, 코드에서 Document.Annotations를 변경한 뒤에는 뷰어의 AnnotationsChanged 메서드도 목록을 다시 만듭니다.
SelectedAnnotation: TsgcPDFAnnotation	읽기 전용. 선택된 항목의 주석이며, 선택된 항목이 없거나 페이지 항목이 선택되어 있으면 nil입니다.

각 항목은 `Author, date: contents` 형식으로 표시됩니다. 내용은 한 줄로 표시되며 긴 텍스트는 줄여서 표시됩니다. 내용이 비어 있으면 대신 Highlight처럼 주석의 유형이 표시됩니다. 작성자가 없는 주석은 (no author)로 표시되고, 날짜가 없는 주석은 날짜를 생략합니다.

- 항목을 선택하면 뷰어가 GotoAnnotation으로 해당 주석까지 스크롤합니다. 뷰어의 AnnotationEditing이 True이면 그 주석이 뷰어에서도 선택됩니다.
- 더블 클릭하면 해당 주석의 팝업이 열립니다.
- 뷰어의 AnnotationEditing이 True일 때 Delete 키는 선택된 주석과 그 답글을 삭제합니다. 뷰어의 OnAnnotationDelete로 거부할 수 있습니다.
- 사용자가 뷰어에서 주석을 선택하면 그 항목이 목록에서 선택됩니다.

Align, Width, OnChange 같은 그 밖의 속성과 이벤트는 FMX TTreeView의 것입니다. 항목은 문서로부터 만들어지며 폼에 저장되지 않습니다.

예제: 코멘트 패널

```
// .NET has no FireMonkey viewer: the WinForms TsgcPDFCommentsView is used
void CreateCommentsPanel()
{
  FComments = new TsgcPDFCommentsView();
  FComments.Parent = this;
  FComments.Dock = DockStyle.Right;
  FComments.Width = 280;
  FComments.GroupByPage = true;
  FComments.ShowReplies = true;
  FComments.AfterSelect += OnCommentChangeEvent;
  // from now on the list follows the document, the annotations and the
  // selection of the viewer
  FComments.Viewer = FViewer;
  // a selected comment is also selected in the viewer, Delete removes it
  FViewer.AnnotationEditing = true;
}

void OnCommentChangeEvent(object Sender, TreeViewEventArgs e)
{
  // null for a page group item
  TsgcPDFAnnotation oAnnot = FComments.SelectedAnnotation;
```

```
  if (oAnnot != null)
    Text = string.Format("{0}, page {1}", oAnnot.Author, oAnnot.PageIndex + 1);
}
```

참고 항목

- TsgcPDFViewerFMX
- TsgcPDFCommentsView
- 주석

OpenType 폰트

`TsgcPDFDocument.AddTrueTypeFont`(설치된 폰트)와 `AddTrueTypeFontFromFile`(.ttf, .otf 또는 .ttc 파일의 첫 번째 폰트)은 두 가지 OpenType 폰트 형식을 모두 지원합니다.

폰트 윤곽선	임베드 형태
TrueType(glyf 테이블, .ttf 및 대부분의 .otf 파일)	CIDFontType2 하위 폰트와 서브셋 FontFile2 프로그램을 가진 Type0 폰트입니다.
PostScript / CFF(CFF 테이블, OTTO로 시작하는 .otf 파일)	CIDFontType0 하위 폰트와 서브타입 CIDFontType0C의 서브셋 FontFile3 프로그램을 가진 Type0 폰트입니다.

두 경우 모두 문서에서 사용된 글리프만 임베드되며, 텍스트를 복사하고 검색할 수 있도록 ToUnicode CMap이 기록되고, 글리프 너비는 /W에 기록됩니다. `TsgcTrueTypeFont.IsCFF`는 로드된 폰트가 어느 형식인지 알려줍니다. 텍스트는 `TsgcPDFPage.DrawTextTT`나 `DrawTextTTColor`로 그리며, 이 메서드들은 복합 스크립트와 셰이핑에서 설명한 OpenType 레이아웃 기능도 적용합니다.

예제

```
static void CreateOpenTypeCFFPDF(string AOTFFile, string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // an .otf font with PostScript (CFF) outlines is embedded as a
    // CIDFontType0 font with a subset FontFile3 /CIDFontType0C program
    int vFont = oPDF.AddTrueTypeFontFromFile(AOTFFile);
    oPage.DrawTextTT("OpenType CFF text, subset and embedded", 50, 780, vFont, 16);
    oPDF.SaveToFile(AFileName);
  }
}
```

AddTrueTypeFontFromFile은 폰트가 설치되어 있지 않을 수 있는 서버, Linux, 모바일 플랫폼에서 폰트를 사용하는 방법입니다. `FindTrueTypeFont(AFamilyName, ABold, AItalic)`은 이미 추가된 폰트의 인덱스를 반환하며, 없으면 -1을 반환합니다.

참고 항목

- 복합 스크립트와 셰이핑
- 중국어, 일본어, 한국어 텍스트
- 텍스트와 폰트

복합 스크립트와 셰이핑

임베드된 TrueType이나 OpenType 폰트로 그린 텍스트(`DrawTextTT`, `DrawTextTTColor`, TrueType 폰트를 사용하는 TsgcPDFCanvas 와 TsgcPDFHTMLRenderer, 그리고 폼 필드 모양)는 sgcPDF 의 OpenType 셰이퍼(유닛 sgcPDF_Shaping, 클래스 TsgcPDFShaper)를 거칩니다. 순수 Pascal로 작성되어 모든 플랫폼에서 동작합니다. 텍스트는 입력한 그대로 논리적 순서로 전달하면 셰이퍼가 시각적 순서로 글리프를 생성합니다.

- **아랍어**: 연결 형태(isol, init, medi, fina), lam alef 같은 필수 리가처(rlig), 마크 배치입니다.
- **히브리어**와 그 밖의 오른쪽에서 왼쪽으로 쓰는 텍스트이며, 괄호 같은 거울 문자도 처리합니다.
- **양방향 텍스트**: 유니코드 양방향 알고리즘(UAX #9)이 왼쪽에서 오른쪽과 오른쪽에서 왼쪽으로 섞인 런을 숫자까지 포함하여 정렬합니다. 문단 방향은 첫 번째 강한 문자에서 결정됩니다.
- **리가처와 문맥 형태**: GSUB 테이블의 ccmp, locl, liga, clig, calt입니다.
- **커닝과 마크 배치**: GPOS 테이블의 kern, mark, mkmk, curs이며, 폰트에 GPOS 커닝이 없으면 예전 kern 테이블을 사용합니다.

텍스트에 오른쪽에서 왼쪽으로 쓰는 런이 포함되면 논리적 텍스트도 /ActualText로 함께 기록되므로, 복사 및 붙여넣기와 텍스트 추출은 입력한 순서 그대로 텍스트를 반환합니다.

예제: 아랍어, 히브리어, 커닝

```
static void CreateMultilingualPDF(string AFontFile, string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // a TrueType or OpenType font file; the used glyphs are embedded
    int vFont = oPDF.AddTrueTypeFontFromFile(AFontFile);
    // Latin with kerning and ligatures from the GPOS / GSUB tables
    oPage.DrawTextTT("Office affinity, AVATAR", 50, 780, vFont, 18);
    // Arabic: joining forms and right to left order are applied
    oPage.DrawTextTT("\u0645\u0631\u062D\u0628\u0627", 50, 740, vFont, 18);
    // Hebrew mixed with numbers (Unicode bidirectional algorithm)
    oPage.DrawTextTT("\u05E9\u05DC\u05D5\u05DD 2026", 50, 700, vFont, 18);
    oPDF.SaveToFile(AFileName);
  }
}
```

셰이퍼 직접 사용하기

`TsgcTrueTypeFont.ShapeText(const AText: WideString; ABaseLevel: Integer = -1): TsgcPDFShapedGlyphs`는 배치된 글리프(GlyphID, Cluster, Text, XAdvance, YAdvance, XOffset, YOffset, 양방향 Level)를 시각적 순서로, 폰트 단위의 간격과 함께 반환합니다. ABaseLevel은 0이면 왼쪽에서 오른쪽, 1이면 오른쪽에서 왼쪽, -1이면 자동 감지입니다. `EncodeShapedText(AText, AFontSize)`는 셰이핑된 텍스트의 콘텐츠 스트림 연산자를 반환합니다. 유닛 sgcPDF_Shaping은 양방향 헬퍼 함수인 `sgcPDFBidiVisual`, `sgcPDFBidiLevels`, `sgcPDFBidiReorder`도 제공합니다.

```
static void ShowShapedGlyphs(string AText)
{
  TsgcTrueTypeFont oFont = new TsgcTrueTypeFont();
  oFont.LoadFromFile(@"C:\Windows\Fonts\arial.ttf");
  // -1: the paragraph direction comes from the first strong character
  TsgcPDFShapedGlyph[] vGlyphs = oFont.ShapeText(AText, -1);
  for (int i = 0; i < vGlyphs.Length; i++)
    Console.WriteLine(string.Format("glyph {0}  cluster {1}  advance {2}  level {3}",
      vGlyphs[i].GlyphID, vGlyphs[i].Cluster, vGlyphs[i].XAdvance, vGlyphs[i].Level));
}
```

폼 필드

표준 폰트 인코딩을 벗어난 문자(키릴 문자, 그리스 문자, CJK, 아랍어, 히브리어)를 담은 폼 필드 값은 임베드된 TrueType 폰트로 모양이 그려지며 동일한 방식으로 셰이핑됩니다. 폰트를 선택하려면 `TsgcPDFAcroForm.AppearanceFontFile`이나

`AppearanceFontName`을 설정하십시오. 둘 다 비어 있으면 해당 문자를 지원하는 첫 번째 설치된 폰트(Arial Unicode MS, Segoe UI, Arial, Yu Gothic, MS Gothic, Microsoft YaHei, Malgun Gothic, Noto Sans)가 사용됩니다.

제한 사항

- 인도 문자(데바나가리, 벵골어, 타밀어)나 동남아시아 문자(태국어, 크메르어)처럼 전용 셰이핑 엔진이 필요한 문자는 음절 재정렬 없이 일반 기능만으로 그려집니다.
- 명시적 양방향 임베딩 및 재정의 문자(LRE, RLE, LRO, RLO, 격리 문자)는 무시됩니다.

참고 항목

- OpenType 폰트
- 중국어, 일본어, 한국어 텍스트
- 폼

중국어, 일본어, 한국어 텍스트

중국어, 일본어, 한국어 텍스트를 쓰려면 `AddTrueTypeFont`나 `AddTrueTypeFontFromFile`로 해당 문자를 포함한 폰트를 추가하고 `DrawTextTT`로 그리십시오. 폰트는 서브셋(사용된 글리프만)으로 임베드되므로 큰 CJK 폰트를 사용해도 출력이 작게 유지되며, 모든 컴퓨터에서 동일하게 표시됩니다. TrueType 컬렉션(.ttc)도 지원되며, 컬렉션의 첫 번째 폰트가 사용됩니다.

```
static void CreateJapanesePDF(string AFontFile, string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // for example C:\Windows\Fonts\msgothic.ttc (the first face is used)
    int vFont = oPDF.AddTrueTypeFontFromFile(AFontFile);
    oPage.DrawTextTT("\u65E5\u672C\u8A9E\u306E\u30C6\u30AD\u30B9\u30C8",
      50, 780, vFont, 20);
    oPDF.SaveToFile(AFileName);
  }
}
```

일반적인 Windows 폰트: MS Gothic, Yu Gothic, Meiryo(일본어), SimSun, Microsoft YaHei(중국어 간체), MingLiU, Microsoft JhengHei(중국어 번체), Batang, Gulim, Malgun Gothic(한국어)입니다. Linux에서는 예를 들어 Noto Sans CJK나 WenQuanYi 폰트를 AddTrueTypeFontFromFile로 사용할 수 있습니다.

CJK 문서 읽기와 렌더링

텍스트 추출과 렌더러는 다른 도구로 만든 CJK 문서도 지원합니다. Identity-H와 Identity-V 인코딩을 사용하는 합성 폰트와 사전 정의된 CJK CMap, 임베드된 CID 폰트(TrueType과 CFF), 세로쓰기를 지원합니다. CJK 폰트를 임베드하지 않고 사용한 문서는 시스템에 설치된 폰트로 렌더링됩니다.

TsgcPDFCJKFontManager

유닛 sgcPDF_Fonts_CJK(Windows 전용)에는 예전의 저수준 클래스인 TsgcPDFCIDFont와 TsgcPDFCJKFontManager, 그리고 헬퍼 함수 `sgcPDFIsCJKCodePoint`와 `TsgcPDFCJKFontManager.ContainsCJK`가 있습니다. 이 클래스들은 TsgcPDFDocument와 연결되어 있지 않습니다. 새 코드에서는 AddTrueTypeFont와 DrawTextTT를 사용하십시오.

참고 항목

- OpenType 폰트
- 복합 스크립트와 셰이핑

텍스트 추출: 문자 위치

일반 텍스트와 조각뿐만 아니라, **TsgcPDFTextExtractor**(유닛 sgcPDF_TextExtract)는 읽기 순서대로 단어와 줄로 묶은 모든 문자를 위치와 함께 반환합니다. VCL 뷰어가 텍스트 선택과 검색 강조 표시에 사용하는 것과 같은 기능이며, 좌표가 있는 텍스트 검색, 고정 레이아웃에서의 데이터 추출, 찾은 텍스트의 마스킹에도 사용할 수 있습니다.

문자, 단어, 줄 API

메서드	설명
GetCharCount, GetChar(AIndex): TsgcPDFTextChar	GetText와 같은 읽기 순서의 문자입니다. TsgcPDFTextChar에는 글리프 사각형(텍스트 방향을 따르는 X1, Y1부터 X4, Y4까지), 경계 상자(Left, Bottom, Right, Top), 원점(OriginX, OriginY), FontSize, LineIndex, WordIndex(공백은 -1), Synthetic(추출기가 두 텍스트 런 사이에 삽입한 공백)이 있습니다.
GetWordCount, GetWord(AIndex): TsgcPDFTextWord	문자 범위(FirstChar, CharCount)로 표현되는 단어이며, LineIndex와 경계 상자를 포함합니다.
GetLineCount, GetLine(AIndex): TsgcPDFTextLine	경계 상자를 포함한 문자 범위로 표현되는 줄입니다.
GetCharsText: WideString	줄 사이를 CRLF로 연결한 문자입니다. GetText와 동일한 텍스트입니다.
CharAt(AX, AY: Double; ATolerance: Double = 0): Integer	사용자 공간의 한 지점에 있는 문자의 인덱스이며, 없으면 -1입니다. ATolerance는 문자 상자를 넓힙니다.
IncludeAnnotations: Boolean	표시되는 주석 모양 스트림(폼 필드 값, 자유 텍스트, 스탬프)의 텍스트도 추출합니다. 기본값은 False입니다.

`sgcPDFCreatePageTextExtractor(AParser, APageIndex)`는 페이지용 추출기를 생성하고 Extract를 실행합니다(직접 해제해야 하며, 페이지가 없으면 nil을 반환). 모든 위치는 PDF 사용자 공간(포인트 단위, 원점은 왼쪽 아래 모서리) 기준입니다.

예제

```
static void ListCharacterPositions(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    using (TsgcPDFTextExtractor oText = new TsgcPDFTextExtractor(oPDF.Parser,
      oPDF.GetPage(0).PageDict))
    {
      // also read the text shown by annotation appearance streams
      oText.IncludeAnnotations = true;
      oText.Extract();
      for (int i = 0; i < oText.GetWordCount(); i++)
      {
        TsgcPDFTextRange vWord = oText.GetWord(i);
        Console.WriteLine(string.Format("word {0}: line {1}, box {2:F1} {3:F1} {4:F1} {5:F1}",
          i, vWord.LineIndex, vWord.Left, vWord.Bottom, vWord.Right, vWord.Top));
      }
      if (oText.GetCharCount() > 0)
      {
        TsgcPDFTextChar vChar = oText.GetChar(0);
```

```
        Console.WriteLine(string.Format("first char at {0:F1}, {1:F1} size {2:F1}",
          vChar.OriginX, vChar.OriginY, vChar.FontSize));
        // hit test: the character under a point of the page (user space)
        Console.WriteLine("char at point: " + oText.CharAt(vChar.OriginX + 1,
          vChar.OriginY + 2, 1));
      }
    }
  }
}
```

향상된 추출

- 폼 XObject 안의 텍스트도 추출되며, 폼 매트릭스가 위치에 적용됩니다.
- 마크된 콘텐츠의 /ActualText는 자신이 덮는 글리프를 대체하므로, 리가처, 셰이핑된 아랍어, 하이픈 표시도 원본 텍스트로 추출됩니다.
- ToUnicode CMap이 없는 폰트는 인코딩, /Differences, 글리프 이름을 통해 매핑됩니다.

참고 항목

- TsgcPDFTextExtractor
- 텍스트 추출
- TsgcPDFViewerControl

신뢰할 수 없는 파일에 대한 제한

sgcPDF는 업로드된 파일, 이메일 첨부 파일, 스캔한 문서처럼 외부에서 들어온 파일을 여는 데 자주 사용됩니다. 모든 파서와 디코더는 입력을 검사하며, 손상된 데이터에 대해서는 범위를 벗어나 읽는 대신 예외를 발생시킵니다. 할당량은 다음 전역 변수로 제한되며, 문서를 로드하기 전에 값을 낮추거나 높일 수 있습니다.

변수	유닛	기본값	제한
sgcPDFMaxDecodedSize: Int64	sgcPDF_Filters	268435456 (256 MB)	하나의 스트림(Flate, LZW, RunLength, ASCII85, ASCIIHex, CCITT, JBIG2)의 최대 디코딩 크기입니다. 압축 폭탄으로부터 보호합니다.
sgcPDFJPEGMaxPixels: Int64	sgcPDF_JPEG	67108864	최대 JPEG 프레임 크기입니다(너비 x 높이).
sgcPDFJPEGMaxMemory: Int64	sgcPDF_JPEG	1073741824 (1 GB)	이미지 하나당 JPEG 디코더의 작업 메모리입니다.
sgcPDFJPXMaxPixels: Int64	sgcPDF_JPX	67108864	최대 JPEG 2000 이미지 크기입니다(너비 x 높이).
CS_CANVAS_MAX_RASTER (constant)	sgcPDF_Renderer_Canvas	67108864	렌더러가 생성하는 최대 래스터 크기이며, 픽셀 단위입니다.

```
static void ConfigureLimits()
{
  // largest decoded stream (default 256 MB); larger streams raise
  sgcPDF_Filters.sgcPDFMaxDecodedSize = 64 * 1024 * 1024;
}
```

그 밖에도 여러 보호 장치가 내장되어 있습니다. 중첩된 폼 XObject, 패턴, Type 3 글리프, 페이지 트리, 개요에 대한 재귀 제한, 객체 참조의 순환 감지, 뷰어의 캐시 크기 제한입니다. 손상된 이미지는 건너뛰거나 부분적으로만 그려지며, 페이지의 나머지 부분은 계속 렌더링됩니다.
이 변수들은 프로세스 전역입니다. 다른 스레드가 문서를 로드하는 중이 아니라 시작 시 한 번만 설정하십시오.

참고 항목

- 오류 처리
- 이미지 디코더

이미지 디코더

sgcPDF에는 PDF 파일에서 사용하는 모든 이미지 압축용 순수 Pascal 디코더가 포함되어 있습니다. 렌더러, 뷰어, TsgcPDFImageExtractor, 옵티마이저가 이 디코더를 사용하며, 직접 호출할 수도 있습니다. 모든 플랫폼에서 실행되며 DLL이 필요 없습니다.

형식	유닛과 함수	지원 내용
JPEG (DCTDecode)	sgcPDF_JPEG: sgcPDFDecodeJPEG, TsgcPDFJPEGDecoder	베이스라인과 프로그레시브 허프만, 8비트, 1개, 3개 또는 4개 구성 요소, 임의의 샘플링 비율, 재시작 마커, JFIF와 Adobe 변환(YCbCr, YCCK, CMYK)을 지원합니다. 산술 코딩, 무손실, 12비트 JPEG는 EsgcPDFJPEGError를 발생시킵니다.
JPEG 2000 (JPXDecode)	sgcPDF_JPX: sgcPDFDecodeJPX	JP2 파일과 원시 코드스트림, 타일, 모든 진행 순서, 가역 5/3 및 비가역 9/7 웨이블릿, 구성 요소 서브샘플링, 1비트부터 16비트까지, 팔레트, 알파 채널, sYCC를 지원합니다.
CCITT Group 3 and Group 4 (CCITTFaxDecode)	sgcPDF_CCITT: sgcPDFCCITTFaxDecode, sgcPDFCCITTFaxDecodeParms	1차원 Group 3(MH), 2차원 Group 3(MR), Group 4(MMR), EndOfLine, EncodedByteAlign, EndOfBlock, BlackIs1을 지원합니다.
JBIG2 (JBIG2Decode)	sgcPDF_JBIG2: sgcPDFJBIG2Decode	일반 영역(산술 및 MMR), 산술 코딩을 사용하는 기호 사전과 텍스트 영역, /JBIG2Globals를 지원합니다. 하프톤, 패턴, 정제, 허프만 코딩된 기호 및 텍스트 세그먼트는 지원되지 않으며 예외를 발생시킵니다.

함수 시그니처

- `sgcPDFDecodeJPEG(const AData: TBytes; out AWidth, AHeight, AComponents: Integer): TBytes`와 `AColorTransform: Integer`(PDF의 /ColorTransform 값, 없으면 -1)를 받는 오버로드가 있습니다. 인터리브된 8비트 샘플을 반환합니다.
- `sgcPDFDecodeJPX(const AData: TBytes; out AInfo: TsgcPDFJPXInfo; AKeep16: Boolean = False): TBytes`입니다. AInfo는 Width, Height, Components, ColorComponents, HasAlpha, BitsPerComponent, ColorSpace(sgcJPXColorGray, sgcJPXColorRGB, sgcJPXColorCMYK, sgcJPXColorICC), ICC 프로필을 제공합니다.
- `sgcPDFCCITTFaxDecode(const AData: TBytes; AK, AColumns, ARows: Integer; AEndOfLine, AEncodedByteAlign, AEndOfBlock, ABlackIs1: Boolean; ADamagedRowsBeforeError: Integer = 0): TBytes`입니다. AK가 < 0이면 Group 4, 0이면 Group 3 MH, > 0이면 Group 3 MR입니다. 바이트 단위로 패딩된 1비트 행을 반환합니다.
- `sgcPDFJBIG2Decode(const AData, AGlobals: TBytes; out AWidth, AHeight: Integer): TBytes`입니다. PDF 규칙(0이 검은색)에 따른 1비트 행을 반환합니다.

예제

```
static void DecodeJPEGFile(string AFileName)
{
  int vWidth, vHeight, vComponents;
  // 1 (gray), 3 (RGB) or 4 (CMYK) bytes per pixel
  byte[] vPixels = sgcPDF_JPEG.sgcPDFDecodeJPEG(File.ReadAllBytes(AFileName), out vWidth,
    out vHeight, out vComponents);
  Console.WriteLine(string.Format("{0} x {1}, {2} components", vWidth, vHeight, vComponents));
}
```

모든 디코더는 손상된 데이터를 예외로 거부하며 할당량을 제한합니다. 신뢰할 수 없는 파일에 대한 제한을 참조하십시오.

참고 항목

- TsgcPDFImageExtractor
- TIFF 가져오기
- 지원하는 렌더링 기능

TIFF 가져오기

TsgcPDFImage와 TsgcPDFDocument.AddImage는 JPEG, PNG, BMP뿐만 아니라 TIFF 파일도 로드합니다. 덕분에 스캔한 문서나 팩스 파일을 손쉽게 PDF로 변환할 수 있습니다.

지원하는 TIFF 파일

- 리틀 엔디언과 빅 엔디언(II와 MM), strip과 tile을 지원합니다.
- 압축: 없음, PackBits, LZW, Deflate(수평 예측기 포함), CCITT Group 3과 Group 4, JPEG를 지원합니다.
- 이진, 그레이스케일, RGB, 팔레트, CMYK 이미지를 지원하며, 샘플당 1비트부터 16비트까지 지원합니다(16비트는 8비트로 축소됨).
- 다중 페이지 파일: 모든 페이지(IFD)를 추가할 수 있습니다.

단일 strip CCITT와 JPEG 페이지는 CCITTFaxDecode와 DCTDecode 스트림으로 다시 인코딩하지 않고 임베드되므로, 스캔한 팩스 페이지는 원본 크기와 품질을 유지합니다. 그 밖의 페이지는 Flate로 압축됩니다.

API

멤버	설명
TsgcPDFDocument.AddImage(const AFileName: string; APageIndex: Integer): Integer	다중 페이지 TIFF 파일의 APageIndex(0 기반) 페이지를 추가합니다. 다른 형식은 페이지 0만 허용합니다.
TsgcPDFImage.TIFFPageCount(const AData: TBytes): Integer	클래스 함수입니다. TIFF 파일의 페이지 수를 반환합니다.
TsgcPDFImage.LoadTIFF(const AData: TBytes; APageIndex: Integer = 0)	한 페이지를 TsgcPDFImage로 로드합니다(이후 TsgcPDFDocument.AddImageObject로 추가).
TsgcPDFImage.IsCCITT: Boolean	페이지가 다시 인코딩하지 않고 CCITT 데이터로 임베드되어 있으면 True입니다.

예제: 다중 페이지 TIFF를 PDF로

```
static void TIFFToPDF(string ATIFFFile, string APDFFile)
{
  // number of pages (IFDs) of the TIFF file
  int vPages = TsgcPDFImage.TIFFPageCount(File.ReadAllBytes(ATIFFFile));
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    for (int i = 0; i < vPages; i++)
    {
      TsgcPDFPage oPage = oPDF.AddPage(595, 842);
      // APageIndex selects the TIFF page; CCITT G3/G4 and JPEG strips are
      // embedded without re-encoding
      int vImage = oPDF.AddImage(ATIFFFile, i);
      oPage.DrawImageFromIndex(vImage, 50, 300, 495, 371);
    }
    oPDF.SaveToFile(APDFFile);
  }
}
```

참고 항목

- 이미지
- 이미지 디코더

TsgcPDFImageExtractor

TsgcPDFImageExtractor(유닛 sgcPDF_ImageExtract)는 PDF의 이미지 목록을 확인하여 PNG, JPEG, BMP 파일로 저장하거나 픽셀을 반환합니다. 이미지 XObject, 인라인 이미지, 폼 XObject 안의 이미지(선택적으로 주석 안의 이미지도)를 각 이미지가 그려진 위치와 함께 찾습니다. sgcPDF 팔레트 페이지에 있습니다.

속성

속성	설명
Document: TsgcPDFDocument	로드된 문서입니다.
Parser: TsgcPDFParser	명시적인 파서입니다. Document보다 우선합니다.
IncludeInline: Boolean	인라인 이미지(BI, ID, EI)를 나열합니다. 기본값은 True입니다.
IncludeAnnotations: Boolean	주석 모양 스트림으로 그려진 이미지를 나열합니다. 기본값은 False입니다.
IncludeUnused: Boolean	콘텐츠가 그리지 않는 페이지 리소스의 이미지도 나열합니다. 기본값은 True입니다.
UniqueOnly: Boolean	모든 이미지 객체를 한 번씩만 나열합니다. UseCount는 그려진 횟수를 알려줍니다. 기본값은 False입니다.
IncludeAlpha: Boolean	/SMask나 /Mask 투명도를 알파 채널로 추가합니다. 기본값은 True입니다.
MaxFormDepth: Integer	폼 XObject의 최대 중첩 수준입니다. 기본값은 16입니다.
FileNamePattern: string	ExtractAll에서 확장자 없이 사용하는 파일 이름입니다. %p는 페이지 번호, %i는 이미지 번호, %n은 페이지 내 이미지 번호, %o는 객체 번호, %r은 리소스 이름입니다.
Count, Items[AIndex]: TsgcPDFImageItem	찾은 이미지입니다. 스캔은 필요할 때 실행됩니다.
Errors: TStringList	ExtractAll이 저장하지 못한 이미지입니다.

메서드

메서드	설명
Scan, ScanPage(APageIndex), Clear	모든 페이지나 한 페이지의 이미지를 나열합니다.
SaveToFile(AIndex; const AFileName: string)	확장자로 형식을 선택합니다. .png, .jpg, .jpeg, .bmp입니다. .jp2, .jpx, .j2k, .jb2, .bin, .raw 확장자는 저장된 스트림 데이터를 그대로 저장합니다.
SaveAsPNG, SaveAsJPEG, SaveAsBMP, SaveToStream(AIndex; AStream; AFormat)	이미지 하나를 저장합니다. AFormat: pifAuto(JPEG 이미지는 JPEG로, JPEG 2000은 저장된 그대로, 나머지는 PNG로), pifPNG, pifJPEG, pifBMP 또는 pifRaw입니다.
ExtractAll(const AFolder: string; AFormat = pifAuto): Integer	나열된 모든 이미지를 저장하고 기록된 파일 수를 반환합니다.
GetPixels(AIndex; out AWidth, AHeight, AChannels: Integer): TBytes	위에서 아래로 디코딩된 8비트 픽셀입니다: 1은 Gray, 2는 Gray와 alpha, 3은 RGB, 4는 RGBA입니다.

메서드	설명
GetRawBytes, GetJPEGBytes, GetPNGBytes, GetBMPBytes, CanSaveAsJPEG	메모리 안의 이미지 데이터입니다. JPEG 출력은 다시 압축하지 않고 저장되는 DCTDecode 이미지에서 사용할 수 있습니다.

TsgcPDFImageItem 은 이미지 하나를 나타냅니다. PageIndex, ObjectNumber, Name, Source(pisXObject 또는 pisInline), Width, Height, BitsPerComponent, ColorSpace, Filters, Matrix, Rect(그려진 영역, 포인트 단위), Placed, InAnnotation, HasSMask, HasMask, IsImageMask, UseCount, Stream입니다.

예제: 이미지 목록 확인과 저장

```
static void ExtractImages(string APDFFile, string AFolder)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    using (TsgcPDFImageExtractor oExtractor = new TsgcPDFImageExtractor())
    {
      oExtractor.Document = oPDF;
      oExtractor.UniqueOnly = true;
      for (int i = 0; i < oExtractor.Count; i++)
      {
        TsgcPDFImageItem oItem = oExtractor.Items(i);
        Console.WriteLine(string.Format("page {0}: {1}x{2} {3} [{4}] at {5:F0},{6:F0}",
          oItem.PageIndex + 1, oItem.Width, oItem.Height, oItem.ColorSpace, oItem.Filters,
          oItem.Rect.Left, oItem.Rect.Bottom));
      }
      // one file per image: page number and image number in the name
      oExtractor.FileNamePattern = "page%p_img%n";
      Console.WriteLine("files written: " +
        oExtractor.ExtractAll(AFolder, TsgcPDFImageExtractFormat.pifPNG));
      // or a single image in the format given by the extension
      if (oExtractor.Count > 0)
        oExtractor.SaveToFile(0, Path.Combine(AFolder, "first.bmp"));
    }
  }
}
```

예제: 픽셀

```
static void ReadImagePixels(string APDFFile)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    using (TsgcPDFImageExtractor oExtractor = new TsgcPDFImageExtractor())
    {
      oExtractor.Document = oPDF;
      oExtractor.ScanPage(0);
      if (oExtractor.Count > 0)
      {
        int vWidth, vHeight, vChannels;
        // decoded samples, 8 bits per channel, top-down rows
        byte[] vPixels = oExtractor.GetPixels(0, out vWidth, out vHeight, out vChannels);
        Console.WriteLine(string.Format("{0} x {1}, {2} channels, {3} bytes", vWidth, vHeight,
          vChannels, vPixels.Length));
      }
    }
  }
}
```

인코더는 함수로도 제공됩니다: `sgcPDFEncodePNG`와 `sgcPDFEncodeBMP(const APixels: TBytes; AWidth, AHeight, AChannels: Integer): TBytes`입니다.

참고 항목

- 이미지 디코더
- 이미지

TsgcPDFCanvas

TsgcPDFCanvas(유닛 sgcPDF_Canvas)는 VCL TCanvas를 본뜬 API로 PDF 페이지에 그립니다. Pen, Brush, Font 객체, MoveTo와 LineTo, Rectangle, Ellipse, Arc, Pie, TextOut, TextRect 등을 제공합니다. 모든 요소는 벡터 그래픽과 실제 텍스트로 페이지 콘텐츠에 기록됩니다. 기존 TCanvas 인쇄 코드를 PDF로 이식하는 가장 쉬운 방법이며, 크로스 플랫폼입니다(VCL과 GDI가 필요 없음).

좌표

캔버스 원점은 TCanvas와 마찬가지로 페이지의 **왼쪽 위** 모서리이며 Y 값은 아래로 갈수록 증가합니다. 단위는 생성자에서 선택합니다. `Create(APage: TsgcPDFPage; AUnit: TsgcPDFCanvasUnit = pcuPoints; ADPI: Double = 96)`이며, pcuPoints, pcuPixels(ADPI 기준), pcuMillimeters, pcuInches 중에서 선택할 수 있습니다. Translate, Rotate, Scale, MultiplyTransform, SetTransform, ResetTransform으로 변환을 변경할 수 있습니다.

주요 멤버

멤버	설명
Pen	Color, Width(0은 표시 가능한 가장 얇은 선을 그림), Style(ppsSolid, ppsDash, ppsDot, ppsDashDot, ppsDashDotDot, ppsClear), Cap, Join, MiterLimit, GapFill, GapColor입니다.
Brush	Color와 Style(pbsSolid, pbsClear)입니다.
Font	Name, Size(포인트), Style(pfsBold, pfsItalic, pfsUnderline, pfsStrikeOut), Color입니다.
Opacity: Double	채우기와 선의 알파값이며, 0부터 1까지입니다.
TrueTypeFonts: Boolean	Arial/Helvetica, Times, Courier 외의 폰트는 설치되어 있으면 TrueType으로 임베드됩니다. False로 설정하면 모든 폰트가 표준 14가지 폰트에 매핑됩니다.
MoveTo, LineTo, Rectangle, FillRect, FrameRect, RoundRect, Ellipse, Arc, ArcTo, Pie, Chord, Polygon, Polyline, PolylineTo, PolyBezier, PolyBezierTo	TCanvas의 매개변수 규칙을 따르는 선과 도형입니다.
BeginPath, EndPath, CloseFigure, FillPath, StrokePath, StrokeAndFillPath, ClipPath	경로입니다.
IntersectClipRect, ExcludeClipRect, ResetClip, SaveState, RestoreState	클리핑과 상태입니다.
TextOut, TextOutEx, TextRect, TextWidth, TextHeight, TextAscent, TextDescent	정렬, 줄바꿈, 문자 간격, 회전(TextOutEx)을 지원하는 텍스트입니다.
DrawImage(X, Y, W, H; AImageIndex | AData: TBytes | AStream | AImage: TsgcPDFImage)	이미지입니다.
EndDraw	열려 있는 클립 그룹을 닫습니다. 페이지 그리기를 마쳤을 때 호출하십시오.

색상은 TColor 형식($00BBGGRR)의 TsgcPDFCanvasColor 값입니다. `sgcPDFCanvasRGB(R, G, B)`로 만들 수 있습니다.

예제

```
static void DrawWithCanvas(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // millimeters, origin at the top left corner, Y grows downwards
    using (TsgcPDFCanvas oCanvas = new TsgcPDFCanvas(oPDF.AddPage(595, 842),
      TsgcPDFCanvasUnit.pcuMillimeters))
    {
      oCanvas.Pen.Color = sgcPDF_Canvas.sgcPDFCanvasRGB(0, 64, 128);
      oCanvas.Pen.Width = 0.5;
      oCanvas.Brush.Color = sgcPDF_Canvas.sgcPDFCanvasRGB(220, 235, 250);
      oCanvas.RoundRect(20, 20, 190, 60, 6, 6);
      oCanvas.Font.Name = "Arial";
      oCanvas.Font.Size = 18;
      oCanvas.Font.Style = TsgcPDFFontStyles.pfsBold;
      oCanvas.TextRect(20, 20, 190, 60, "Drawn with TsgcPDFCanvas", TsgcPDFTextAlign.ptaCenter,
        TsgcPDFTextVAlign.ptvCenter);
      oCanvas.Brush.Color = sgcPDF_Canvas.sgcPDFCanvasRGB(255, 200, 0);
      oCanvas.Pie(20, 80, 80, 140, 80, 110, 50, 80);
      oCanvas.Pen.Style = TsgcPDFPenStyle.ppsDash;
      oCanvas.MoveTo(100, 80);
      oCanvas.LineTo(190, 140);
      // semi transparent overlay
      oCanvas.Opacity = 0.4;
      oCanvas.Brush.Color = sgcPDF_Canvas.sgcPDFCanvasRGB(255, 0, 0);
      oCanvas.Ellipse(120, 90, 170, 130);
      oCanvas.EndDraw();
    }
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- EMF에서 PDF로
- TsgcPDFVCLCanvasHelper
- HTML에서 PDF로
- 그래픽

EMF에서 PDF로

TsgcPDFEMFPlayer(유닛 sgcPDF_EMF, Windows 전용)는 Windows 향상된 메타파일(EMF)을 TsgcPDFCanvas에 재생합니다. 메타파일의 선, 도형, 경로, 텍스트, 비트맵은 PDF에서 벡터 그래픽, 실제 텍스트, 이미지가 되므로, 다른 컴포넌트로 그린 차트와 보고서가 어떤 배율에서도 선명하게 유지되고 텍스트를 검색할 수 있습니다.

API

멤버	설명
Create(ACanvas: TsgcPDFCanvas)	그리기를 받을 캔버스입니다.
Play(AHandle: HENHMETAFILE; X, Y, W, H: Double)	메타파일의 프레임이 사각형(캔버스 단위)을 채우도록 재생합니다.
PlayFromFile, PlayFromStream, PlayFromBytes	동일한 기능이며, .emf 파일, 스트림, 바이트에서 재생합니다.
GetFrameSize(AHandle; out AWidth, AHeight: Double)	클래스 프로시저입니다. 메타파일 프레임 크기를 포인트 단위로 반환합니다.
RecordCount, SkippedRecords	재생된 레코드 수와 지원되지 않는 레코드 수입니다.
sgcPDFEMFAddPage(ADocument; AHandle | AFileName): TsgcPDFPage	메타파일 프레임 크기의 페이지를 추가하고 그 위에 메타파일을 그립니다.

예제

```
static void EMFToPDF(string AEMFFile, string APDFFile)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // a page with the size of the metafile frame
    sgcPDF_EMF.sgcPDFEMFAddPage(oPDF, AEMFFile);
    // or play a metafile into a rectangle of any page
    using (TsgcPDFCanvas oCanvas = new TsgcPDFCanvas(oPDF.AddPage(595, 842)))
    {
      TsgcPDFEMFPlayer oPlayer = new TsgcPDFEMFPlayer(oCanvas);
      oPlayer.PlayFromFile(AEMFFile, 50, 50, 300, 225);
      Console.WriteLine(string.Format("records {0}, skipped {1}", oPlayer.RecordCount,
        oPlayer.SkippedRecords));
      oCanvas.EndDraw();
    }
    oPDF.SaveToFile(APDFFile);
  }
}
```

참고 항목

- TsgcPDFCanvas
- TsgcPDFVCLCanvasHelper

TsgcPDFVCLCanvasHelper

TsgcPDFVCLCanvasHelper(유닛 sgcPDF_CanvasVCL)는 VCL 그래픽을 TsgcPDFCanvas에 연결합니다. Windows와 VCL 전용입니다.

멤버	설명
Create(ACanvas: TsgcPDFCanvas)	대상 캔버스입니다.
DrawMetafile(AMetafile: TMetafile; X, Y, W, H: Double)	TMetafile을 벡터 그래픽으로 재생합니다.
DrawGraphic(AGraphic: TGraphic; X, Y, W, H: Double)	임의의 TGraphic을 그립니다. 메타파일은 벡터 그래픽으로, 비트맵, PNG, JPEG 이미지는 이미지로 그립니다.
DrawControl(AControl: TWinControl; X, Y, W, H: Double)	향상된 메타파일을 통해 컨트롤과 그 자식(예: 차트나 그리드)을 벡터 그래픽으로 그립니다.
ControlToMetafile(AControl: TWinControl): TMetafile	클래스 함수입니다. 컨트롤을 새 메타파일에 기록합니다(직접 해제해야 함).

X, Y, W, H는 캔버스 단위입니다.

예제

```
static void GraphicToPDF(Image AGraphic, string APDFFile)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    using (TsgcPDFCanvas oCanvas = new TsgcPDFCanvas(oPDF.AddPage(595, 842)))
    {
      TsgcPDFVCLCanvasHelper oHelper = new TsgcPDFVCLCanvasHelper(oCanvas);
      // a Metafile is played as vector graphics, other images as bitmaps
      oHelper.DrawGraphic(AGraphic, 50, 50, 400, 300);
      oCanvas.EndDraw();
    }
    oPDF.SaveToFile(APDFFile);
  }
}
```

참고 항목

- TsgcPDFCanvas
- EMF에서 PDF로

HTML에서 PDF로

TsgcPDFHTMLRenderer(유닛 sgcPDF_HTML)는 CSS가 적용된 HTML을 페이지로 나뉜 PDF 페이지로 변환합니다. 순수 Pascal로 작성된 자체 레이아웃 엔진을 갖추고 있어 브라우저가 필요 없으며 모든 플랫폼에서 실행됩니다. HTML 템플릿으로 생성하는 인보이스, 보고서, 편지, 이메일 아카이브에 사용하십시오. sgcPDF 팔레트 페이지에 있습니다.

메서드

메서드	설명
HTMLToPDFFile(const AHTML, APDFFileName: string; const ABasePath: string = '')	HTML 문자열로부터 PDF 파일을 생성합니다.
HTMLFileToPDFFile(const AHTMLFileName, APDFFileName: string)	HTML 파일로부터 PDF 파일을 생성합니다.
RenderHTML(const AHTML: string; ADocument: TsgcPDFDocument; const ABasePath: string = ''): Integer	문서에 페이지를 추가하고 추가된 페이지 수를 반환합니다. RenderHTMLW는 WideString을, RenderHTMLBytes는 원시 바이트를(인코딩은 BOM, <meta charset> 또는 UTF-8 감지로 결정), RenderHTMLFile은 파일을 받습니다.

ABasePath는 상대 이미지 및 스타일시트 경로를 해석합니다. Delphi 2009 이전 버전에서는 AnsiString이 UTF-8 또는 Windows-1252를 담을 수 있으며, 자동으로 감지됩니다.

속성

속성	설명
PageWidth, PageHeight: Double	포인트 단위의 페이지 크기입니다. CSS @page size가 이 값을 재정의합니다.
Margins: TsgcPDFHTMLMargins	포인트 단위의 Left, Top, Right, Bottom입니다. CSS @page margin이 이 값을 재정의합니다.
DefaultFontName, DefaultFontSize	font-family나 font-size가 없는 텍스트에 사용할 폰트입니다.
TrueTypeFonts: Boolean	표준 14가지 폰트 외의 폰트 패밀리를 TrueType 폰트로 임베드합니다. Windows-1252 문자 집합을 벗어나는 텍스트에 필요합니다.
HeaderText, FooterText: string	위쪽과 아래쪽 여백 가운데에 표시되는 텍스트입니다. {page}와 {pages}는 각각 페이지 번호와 전체 페이지 수로 대체됩니다.
MaxPages: Integer	지정한 페이지 수에 도달하면 중단합니다.
OnPageHeader, OnPageFooter(Sender; ACanvas: TsgcPDFCanvas; APageNumber, APageCount: Integer)	TsgcPDFCanvas로 직접 머리글과 바닥글을 그립니다.
LastPageCount	마지막 렌더링에서 추가된 페이지 수입니다.

지원하는 HTML

- 블록 요소: html, body, div, p, h1부터 h6까지, blockquote, pre, hr, address, center, section, article, header, footer, nav, aside, main, figure, figcaption, form, fieldset, details, summary.
- 목록: ul, ol, li(disc, circle, square, decimal, decimal-leading-zero, 소문자와 대문자 alpha, latin, roman), dl, dt, dd.
- 인라인 요소: span, a, b, strong, i, em, u, ins, s, strike, del, small, big, sub, sup, code, kbd, samp, tt, cite, var, dfn, br.
- 표: table, caption, thead, tbody, tfoot, tr, th, td를 지원하며, colspan과 rowspan도 지원합니다. 머리글 행은 모든 페이지에 반복됩니다.
- 이미지: JPEG, PNG, BMP, TIFF 파일(ABasePath 기준의 로컬 경로)과 base64로 인코딩된 data: URI를 사용하는 img를 지원하며, width와 height도 지원합니다.
- 링크: a href는 링크 주석이 됩니다.
- 스타일시트: <style> 블록, style 속성, 로컬 파일을 가리키는 <link rel="stylesheet">를 지원합니다.

지원하는 CSS

- 선택자: 요소, 클래스, id, 하위 및 자식 결합자, :first-child를 지원하며, 우선순위, 상속, !important도 지원합니다.
- 박스 모델: margin, padding, border(width, style solid, dashed, dotted, 색상, 변 별 지정), width, max-width, height, background-color, border-collapse, border-spacing을 지원합니다.
- 텍스트: color, font, font-family, font-size(키워드, em, rem, %, px, pt, mm, cm, in 포함), font-weight, font-style, line-height, letter-spacing, text-align(justify 포함), text-indent, text-decoration, vertical-align, white-space를 지원합니다.
- 목록과 표시: list-style, list-style-type, display(block, inline, list-item, table과 그 구성 요소, none)를 지원합니다.
- 페이지 나누기: page-break-before, page-break-after, page-break-inside, break-before, break-after, break-inside, 그리고 size(A3, A4, A5, B5, letter, legal, landscape)와 margin이 포함된 @page를 지원합니다. 줄, 이미지, 표 행은 페이지 사이에서 나뉘지 않습니다.
- 색상 이름, #rgb, #rrggbb, rgb(), rgba()를 지원합니다.

지원하지 않는 기능: JavaScript, float, absolute 및 fixed 위치 지정, flexbox, grid, 웹 폰트(@font-face), HTTP로 로드하는 이미지, SVG, 폼 컨트롤입니다. 지원하지 않는 속성은 무시됩니다.

예제: HTML 문자열을 PDF로

```
static void HTMLToPDF(string APDFFile)
{
  using (TsgcPDFHTMLRenderer oHTML = new TsgcPDFHTMLRenderer())
  {
    oHTML.PageWidth = 595;
    oHTML.PageHeight = 842;
    oHTML.FooterText = "Page {page} of {pages}";
    oHTML.HTMLToPDFFile(
      "<html><head><style>" +
      "h1 { color: #003366; border-bottom: 1px solid #999 }" +
      "td, th { border: 1px solid #ccc; padding: 4px }" +
      ".total { font-weight: bold; text-align: right }" +
      "</style></head><body>" +
      "<h1>Invoice 2026-001</h1>" +
      "<p>Thank you for your order. See <a href=\"https://www.esegece.com\">" +
      "our website</a>.</p>" +
      "<table><tr><th>Item</th><th>Amount</th></tr>" +
      "<tr><td>sgcPDF license</td><td class=\"total\">299.00</td></tr></table>" +
      "<ul><li>Delivery by download</li><li>One year of updates</li></ul>" +
      "</body></html>", APDFFile);
  }
}
```

예제: 문서에 HTML 추가

```
static int AppendHTML(TsgcPDFDocument ADocument, string AHTMLFile)
{
  using (TsgcPDFHTMLRenderer oHTML = new TsgcPDFHTMLRenderer())
```

```
  {
    oHTML.Margins.Left = 50;
    oHTML.Margins.Right = 50;
    oHTML.DefaultFontName = "Arial";
    oHTML.DefaultFontSize = 11;
    // relative image and stylesheet paths resolve from the HTML file folder
    return oHTML.RenderHTMLFile(AHTMLFile, ADocument);
  }
}
```

참고 항목

- TsgcPDFCanvas
- 표

2D 바코드와 GS1-128

TsgcPDFBarcodeGenerator(유닛 sgcPDF_Barcodes)는 바코드에서 설명한 선형 바코드에 더해 2D 심볼과 GS1 바코드를 벡터 그래픽으로 그립니다. 모든 Draw 메서드의 APage 매개변수는 TsgcPDFPage나 그 콘텐츠 빌더(`Page.Content`)를 받습니다. 다른 객체를 전달하면 예외가 발생합니다.

심볼로지	메서드	옵션
QR Code	DrawQRCode(APage; const AData: AnsiString; AX, AY, AModuleSize: Double; AErrorLevel: TsgcPDFQRErrorLevel; AMode: TsgcPDFQRMode; AVersion: Integer)	버전 1부터 40까지 지원합니다(0을 지정하면 데이터가 맞는 가장 작은 버전을 선택하고, 고정된 버전에 데이터가 맞지 않으면 예외가 발생합니다). 오류 정정은 qrecL, qrecM, qrecQ, qrecH이고, 모드는 qrmAuto, qrmNumeric, qrmAlphanumeric, qrmByte입니다. 짧은 오버로드는 기본값을 사용합니다.
DataMatrix ECC200	DrawDataMatrix(APage; const AData: AnsiString; AX, AY, AModuleSize: Double; AShape: TsgcPDFDataMatrixShape = dmsSquare)	정사각형 24종과 직사각형 6종의 크기를 지원하며, dmsSquare, dmsRectangle 또는 dmsAuto(모양에 관계없이 가장 작은 심볼) 중에서 선택합니다. ASCII 및 Base256 인코딩을 지원합니다.
PDF417	DrawPDF417(APage; const AData: AnsiString; AX, AY, AModuleWidth: Double; ASecurityLevel: Integer = -1; AColumns: Integer = 0)	보안 수준은 0부터 8까지 지원하며(-1은 자동), 데이터 열은 1부터 30까지(0은 자동), 행은 3부터 90까지입니다. 텍스트, 숫자, 바이트 압축을 지원합니다.
GS1-128	DrawBarcode(APage; bcGS1_128; const AData: AnsiString; AX, AY: Double [; const AOptions])	응용 식별자는 괄호로 표기합니다. 예: (01)09501101530003(17)261231. FNC1이 삽입되고 GTIN 검사 숫자가 검증됩니다.

DataMatrix 와 PDF417 은 bcDataMatrix 와 bcPDF417 을 사용해 DrawBarcode 로도 사용할 수 있으며, TsgcPDFBarcodeOptions 의 QRVersion, QRErrorLevel, QRMode, DataMatrixShape, PDF417SecurityLevel, PDF417Columns, PDF417RowHeight 필드를 이용합니다. 심볼을 직접 그리려면 `EncodeQRCodeMatrix`, `EncodeDataMatrix`, `EncodePDF417`, `EncodeGS1_128`, 그리고 모든 선형 심볼로지용 `EncodeLinear` 인코더가 모듈 매트릭스나 바 너비를 반환합니다. TsgcPDFBarcodeSymbol은 모든 심볼로지에 대해 이 작업을 수행합니다. 여백 영역, 텍스트 줄, 회전을 추가하며, VCL과 FireMonkey 바코드 컨트롤이 표시하는 심볼이기도 합니다.

예제

```
static void Create2DBarcodes(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    TsgcPDFBarcodeGenerator oBarcodes = new TsgcPDFBarcodeGenerator();
    // QR Code: level H, alphanumeric mode, version 0 = smallest that fits
    oBarcodes.DrawQRCode(oPage, "HTTPS://WWW.ESEGECE.COM", 50, 650, 3,
      TsgcPDFQRErrorLevel.qrecH, TsgcPDFQRMode.qrmAlphanumeric, 0);
    // DataMatrix ECC200, square or rectangular symbols
    oBarcodes.DrawDataMatrix(oPage, "SGC-PDF-2026", 250, 700, 3);
    oBarcodes.DrawDataMatrix(oPage, "SGC-PDF-2026", 350, 700, 3,
      TsgcPDFDataMatrixShape.dmsRectangle);
    // PDF417: security level 5, 4 data columns
    oBarcodes.DrawPDF417(oPage, "sgcPDF PDF417 sample text", 50, 520, 1, 5, 4);
    // GS1-128: application identifiers in parentheses
    TsgcPDFBarcodeOptions oOptions = sgcPDF_Barcodes.sgcPDFDefaultBarcodeOptions();
    oOptions.Height = 40;
    oBarcodes.DrawBarcode(oPage, TsgcPDFBarcodeType.bcGS1_128, "(01)09501101530003(17)261231",
      50, 400, oOptions);
```

```
    oPDF.SaveToFile(AFileName);
  }
}
```

구현되지 않은 기능: DataMatrix의 C40 및 Text 인코딩, QR Code의 혼합 모드 세그먼트화입니다. 출력은 여전히 유효하지만 일부 데이터에서는 압축률이 낮을 수 있습니다.

참고 항목

- 바코드
- TsgcPDFCanvas
- TsgcPDFBarcodeSymbol
- TsgcPDFBarcodeControl

TsgcPDFBarcodeControl

TsgcPDFBarcodeControl(유닛 sgcPDF_Barcode_VCL)는 폼에 바코드 또는 2D 심볼을 표시하는 VCL 컨트롤입니다. TsgcPDFBarcodeGenerator의 인코더로 텍스트를 인코딩하는 TsgcPDFBarcodeSymbol을 호스팅하며, 어두운 모듈 연속 구간마다 채워진 사각형을 하나씩 그리므로 모든 모듈이 정수 픽셀에서 시작하고 끝나 스캐너가 읽기에 심볼이 선명하게 유지됩니다. sgcPDF의 11가지 심볼로지를 지원합니다: Code 128, Code 39, EAN-13, EAN-8, UPC-A, ITF, Codabar, GS1-128, QR Code, DataMatrix, PDF417.
이 컨트롤은 GDI+ 없이 VCL 캔버스만으로 그리므로 Delphi 7부터 동작합니다. VCL 패키지로 설치되며 **sgcPDF** 팔레트 페이지에 있습니다. 같은 심볼을 PNG나 SVG로 저장하거나, 화면과 같은 모듈로 PDF 페이지에 그릴 수 있습니다.
.NET에서 **TsgcPDFBarcodeControl**은 **esegece.sgcPDF.WinForms** 패키지의 Windows Forms 컨트롤입니다. System.Windows.Forms.Control에서 파생되고, 더블 버퍼링을 사용하며, 포커스를 받지 않고, 같은 속성, 메서드, OnError 이벤트를 가집니다. Color는 BackColor(기본값 White)이고, 텍스트 줄은 Font의 글꼴과 ForeColor의 색을 사용하며, Align은 Dock, OnClick은 Click 이벤트입니다. 색은 System.Drawing.Color 값이고, PaintTo는 Graphics와 Rectangle을 받으며, IsValid는 메서드이고, ModuleSize는 폼의 DPI 배율을 따릅니다. PDFBarcodeControl 데모는 모든 심볼로지를 보여 줍니다.

공개 속성

속성	설명
Symbology: TsgcPDFBarcodeType	심볼로지: bcCode128(기본값), bcCode39, bcEAN13, bcEAN8, bcUPCA, bcITF, bcCodabar, bcGS1_128 또는 2D 심볼로지인 bcQRCode, bcDataMatrix, bcPDF417.
Text: string	인코딩할 데이터. QR Code, DataMatrix, PDF417은 텍스트의 UTF-8 바이트를 인코딩합니다. 선형 심볼로지는 자체 문자 집합만 허용하며, GS1-128은 괄호로 묶은 애플리케이션 식별자를 받습니다.
QuietZone: Integer	모듈 단위의 밝은 여백. -1(기본값)은 심볼로지의 표준값을 사용합니다. 선형 심볼은 좌우로 10, QR Code는 4, DataMatrix는 1, PDF417은 2이며, 모두 네 변에 적용됩니다.
BarHeight: Integer	선형 심볼의 막대 높이(모듈 단위). 0(기본값)은 사용 가능한 높이를 모두 채웁니다.
ShowText: Boolean	선형 심볼의 막대 아래에 사람이 읽을 수 있는 줄을 그립니다. 기본값은 True입니다. 2D 심볼에는 텍스트 줄이 없습니다.
Rotation: TsgcPDFBarcodeRotation	br0(기본값), br90, br180, br270: 시계 방향 90도 단위 회전. 텍스트 줄은 심볼과 함께 회전합니다.
Fit: TsgcPDFBarcodeFit	bfFitInteger(기본값): 맞는 가장 큰 정수 픽셀 크기의 모듈로, 가운데에 배치합니다. bfModuleSize: 모듈당 정확히 ModuleSize 픽셀로, 가운데에 배치합니다. bfStretch: 심볼이 컨트롤을 가득 채우며 모듈이 1픽셀 차이 날 수 있습니다.
ModuleSize: Integer	bfModuleSize에 쓰는 모듈당 픽셀 수. 기본값은 2입니다. 고DPI 화면에서는 폼과 함께 배율이 조정됩니다.
Color: TColor	배경과 밝은 모듈. 기본값은 clWhite입니다.
BarColor: TColor	어두운 모듈. 기본값은 clBlack입니다.
Font, ParentFont	텍스트 줄의 글꼴, 스타일, 색. 높이는 모듈 크기를 따르므로(막대 아래 12모듈 띠 안의 8모듈) Font.Size는 무시됩니다. 항상 TrueType 글꼴이 사용됩니다.
Transparent: Boolean	배경을 칠하지 않아 막대 사이로 부모가 보입니다. 기본값은 False입니다.

속성	설명
QRErrorLevel: TsgcPDFQRErrorLevel	QR Code 오류 정정: qrecL, qrecM(기본값), qrecQ, qrecH.
QRMode: TsgcPDFQRMode	QR Code 인코딩 모드: qrmByte(기본값), qrmAuto, qrmNumeric, qrmAlphanumeric.
QRVersion: Integer	0(기본값)은 맞는 가장 작은 QR Code 버전을 고르고, 1에서 40은 버전을 고정합니다.
DataMatrixShape: TsgcPDFDataMatrixShape	dmsSquare(기본값), dmsRectangle, dmsAuto(모든 모양 중 가장 작은 심볼).
PDF417SecurityLevel: Integer	PDF417 보안 수준 0에서 8. -1(기본값)은 자동입니다.
PDF417Columns: Integer	PDF417 데이터 열 수 1에서 30. 0(기본값)은 자동입니다.
PDF417RowHeight: Integer	PDF417 행의 높이(모듈 단위). 기본값은 3입니다.
Align, Anchors, Visible, Hint	표준 VCL 속성.

이벤트

이벤트	설명
OnError: TsgcPDFBarcodeErrorEvent	텍스트를 인코딩할 수 없을 때 심볼이 변경될 때마다 한 번, Sender와 aError(Symbol.LastError)와 함께 호출됩니다. 그러면 컨트롤이 가는 빗금 테두리를 그립니다. 그리기에서는 예외가 발생하지 않습니다.
OnClick: TNotifyEvent	컨트롤을 클릭했습니다.

공개 메서드

메서드	설명
IsValid: Boolean	텍스트를 인코딩할 수 있으면 True. 그렇지 않으면 이유를 Symbol.LastError가 알려 줍니다.
PaintTo(aCanvas: TCanvas; const aRect: TRect)	컨트롤이 스스로 그리는 방식대로, 비트맵이나 프린터 캔버스 같은 임의의 캔버스의 aRect에 심볼을 그립니다. 배경(Transparent인 경우 제외), 모듈, 텍스트 줄을 그리며, 텍스트가 잘못되었으면 빗금 테두리를 그립니다.
SaveToPNG(const aFileName: string; aModuleSize: Integer), SaveToPNG(aStream: TStream; aModuleSize: Integer)	심볼을 모듈당 aModuleSize 픽셀의 PNG 이미지로 저장합니다. Color 위에 BarColor로 그립니다. 텍스트 줄은 Font의 글꼴과 BarColor를 사용합니다.
SaveToSVG(const aFileName: string; aModuleSize: Integer), SaveToSVG(aStream: TStream; aModuleSize: Integer)	심볼을 모듈당 aModuleSize 단위의 SVG로 저장합니다. 색과 글꼴은 SaveToPNG와 같습니다.

메서드	설명
Symbol: TsgcPDFBarcodeSymbol	호스팅된 심볼(읽기 전용). 컨트롤이 이를 소유하며 해당 OnChange를 사용합니다. 심볼을 PDF 페이지에 그리거나(DrawToPDF) Runs, ModuleWidth, ModuleHeight를 읽는 데 사용합니다.

SaveToPNG와 SaveToSVG는 텍스트가 잘못되었으면 파일을 만들기 전에 Symbol.LastError와 함께 예외를 발생시킵니다.

예제

```
private TsgcPDFBarcodeControl FBarcode;

public void CreateBarcode()
{
  FBarcode = new TsgcPDFBarcodeControl();
  FBarcode.Parent = this;
  FBarcode.SetBounds(16, 16, 320, 140);
  FBarcode.Symbology = TsgcPDFBarcodeType.bcEAN13;
  FBarcode.Text = "5901234123457";
  FBarcode.BarHeight = 60;
  FBarcode.BarColor = Color.Navy;
  // the text line takes the face of Font and the colour of ForeColor
  FBarcode.Font = new Font("Arial", 9F);
  FBarcode.ForeColor = Color.Navy;
  FBarcode.OnError = OnBarcodeError;
}

void OnBarcodeError(object Sender, string AError)
{
  // the control draws a hatched frame, the event tells why
  Text = AError;
}

public void SaveBarcode(string AFileName)
{
  if (FBarcode.IsValid())
  {
    // 4 pixels per module, BarColor on BackColor
    FBarcode.SaveToPNG(Path.ChangeExtension(AFileName, ".png"), 4);
    FBarcode.SaveToSVG(Path.ChangeExtension(AFileName, ".svg"), 4);
  }
}
```

컨트롤을 PDF에 추가하기

TsgcPDFBarcodeSymbol.DrawToPDF는 컨트롤의 심볼을 화면과 같은 모듈, 회전, 여백 영역(quiet zone), 텍스트 줄로 벡터 사각형으로 PDF 페이지에 그립니다. DrawToPDF는 색을 $RRGGBB로 받지만 VCL TColor는 $BBGGRR이므로, 먼저 빨강과 파랑 바이트를 서로 바꾸십시오.
.NET에서 `Color.ToArgb()`는 0xAARRGGBB를 반환하고 DrawToPDF는 최상위 바이트를 무시하므로, BarColor와 BackColor는 바이트를 바꾸지 않고 `(uint)Color.ToArgb()`로 전달합니다.

```
static void AddBarcodeToPDF(TsgcPDFBarcodeControl AControl, string AFileName)
{
  // Color.ToArgb() is 0xAARRGGBB and DrawToPDF ignores the top byte
  uint vFore = (uint)AControl.BarColor.ToArgb();
  uint vBack = (uint)AControl.BackColor.ToArgb();
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // the modules, rotation, quiet zone and text line of the control,
    // fitted in a 250 x 100 points box whose bottom left corner is 50, 680
    AControl.Symbol.DrawToPDF(oPage, 50, 680, 250, 100, vFore, vBack,
      TsgcPDFStandardFont.sfHelvetica);
```

```
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- TsgcPDFBarcodeFMX
- TsgcPDFBarcodeSymbol
- 바코드
- 2D 바코드와 GS1-128

TsgcPDFBarcodeFMX

TsgcPDFBarcodeFMX(유닛 sgcPDF_Barcode_FMX)는 TsgcPDFBarcodeControl의 FireMonkey 버전입니다. 같은 TsgcPDFBarcodeSymbol을 호스팅하며 어두운 모듈 연속 구간마다 사각형을 하나씩 그립니다. 레이아웃은 캔버스의 배율을 사용해 장치 픽셀 단위로 계산되므로, 모든 모듈이 정수 장치 픽셀에서 시작하고 끝나 고DPI 화면에서도 선명하게 유지됩니다. 컨트롤은 그리기를 자신의 경계로 클리핑합니다.
Delphi XE5 이상이 필요하며(이전 컴파일러에서는 유닛이 비어 있습니다) FMX 패키지로 설치되고 **sgcPDF** 팔레트 페이지에 있습니다.
sgcPDF .NET에는 FireMonkey 유닛이 없으므로 TsgcPDFBarcodeFMX에 해당하는 .NET 컨트롤은 없습니다. 폼에서는 Windows Forms TsgcPDFBarcodeControl을, 사용자 인터페이스가 필요 없는 TsgcPDFBarcodeSymbol을 사용하면 어떤 플랫폼에서도 PNG, SVG, PDF 출력을 작성할 수 있습니다.

공개 속성

속성	설명
Symbology: TsgcPDFBarcodeType	심볼로지: bcCode128(기본값), bcCode39, bcEAN13, bcEAN8, bcUPCA, bcITF, bcCodabar, bcGS1_128 또는 2D 심볼로지인 bcQRCode, bcDataMatrix, bcPDF417.
Text: string	인코딩할 데이터. QR Code, DataMatrix, PDF417은 텍스트의 UTF-8 바이트를 인코딩합니다. 선형 심볼로지는 자체 문자 집합만 허용하며, GS1-128은 괄호로 묶은 애플리케이션 식별자를 받습니다.
QuietZone: Integer	모듈 단위의 밝은 여백. -1(기본값)은 심볼로지의 표준값을 사용합니다. 선형 심볼은 좌우로 10, QR Code는 4, DataMatrix는 1, PDF417은 2이며, 모두 네 변에 적용됩니다.
BarHeight: Integer	선형 심볼의 막대 높이(모듈 단위). 0(기본값)은 사용 가능한 높이를 모두 채웁니다.
ShowText: Boolean	선형 심볼의 막대 아래에 사람이 읽을 수 있는 줄을 그립니다. 기본값은 True입니다. 2D 심볼에는 텍스트 줄이 없습니다.
Rotation: TsgcPDFBarcodeRotation	br0(기본값), br90, br180, br270: 시계 방향 90도 단위 회전. 텍스트 줄은 심볼과 함께 회전합니다.
Fit: TsgcPDFBarcodeFit	bfFitInteger(기본값): 맞는 가장 큰 정수 장치 픽셀 크기의 모듈로, 가운데에 배치합니다. bfModuleSize: 모듈당 ModuleSize 논리 단위를 정수 장치 픽셀로 반올림하여 가운데에 배치합니다. bfStretch: 심볼이 컨트롤을 가득 채우며 모듈이 1장치 픽셀 차이 날 수 있습니다.
ModuleSize: Integer	bfModuleSize에 쓰는 모듈당 논리 단위 수. 다른 FireMonkey 크기와 같습니다. 기본값은 2입니다.
Color: TAlphaColor	배경과 밝은 모듈. 기본값은 흰색입니다.
BarColor: TAlphaColor	어두운 모듈. 기본값은 검은색입니다.
TextSettings: TsgcPDFBarcodeTextSettingsFMX	텍스트 줄의 Font(패밀리와 스타일) 및 FontColor. 높이는 모듈 크기를 따르므로 Font.Size는 무시됩니다.
Transparent: Boolean	배경을 칠하지 않습니다. 기본값은 False입니다.
QRErrorLevel: TsgcPDFQRErrorLevel	QR Code 오류 정정: qrecL, qrecM(기본값), qrecQ, qrecH.

속성	설명
QRMode: TsgcPDFQRMode	QR Code 인코딩 모드: qrmByte(기본값), qrmAuto, qrmNumeric, qrmAlphanumeric.
QRVersion: Integer	0(기본값)은 맞는 가장 작은 QR Code 버전을 고르고, 1에서 40은 버전을 고정합니다.
DataMatrixShape: TsgcPDFDataMatrixShape	dmsSquare(기본값), dmsRectangle, dmsAuto(모든 모양 중 가장 작은 심볼).
PDF417SecurityLevel: Integer	PDF417 보안 수준 0에서 8. -1(기본값)은 자동입니다.
PDF417Columns: Integer	PDF417 데이터 열 수 1에서 30. 0(기본값)은 자동입니다.
PDF417RowHeight: Integer	PDF417 행의 높이(모듈 단위). 기본값은 3입니다.
Align, Anchors, Margins, Position, Width, Height, Size, Visible, Hint, ShowHint	표준 FireMonkey 속성. Size는 Delphi XE7부터, Hint와 ShowHint는 Delphi XE6부터 게시됩니다.

이벤트

이벤트	설명
OnError: TsgcPDFBarcodeErrorEvent	텍스트를 인코딩할 수 없을 때 심볼이 변경될 때마다 한 번, Sender와 aError(Symbol.LastError)와 함께 호출됩니다. 그러면 컨트롤이 가는 빗금 테두리를 그립니다. 그리기에서는 예외가 발생하지 않습니다.
OnClick: TNotifyEvent	컨트롤을 클릭하거나 탭했습니다.

공개 메서드

메서드	설명
IsValid: Boolean	텍스트를 인코딩할 수 있으면 True. 그렇지 않으면 이유를 Symbol.LastError가 알려 줍니다.
SaveToPNG(const aFileName: string; aModuleSize: Integer), SaveToPNG(aStream: TStream; aModuleSize: Integer)	심볼을 모듈당 aModuleSize 픽셀의 PNG 이미지로 저장합니다. Color 위에 BarColor로 그립니다. 텍스트 줄은 TextSettings.Font의 패밀리를 사용합니다.
SaveToSVG(const aFileName: string; aModuleSize: Integer), SaveToSVG(aStream: TStream; aModuleSize: Integer)	심볼을 모듈당 aModuleSize 단위의 SVG로 저장합니다. 색과 글꼴은 SaveToPNG와 같습니다.
Symbol: TsgcPDFBarcodeSymbol	호스팅된 심볼(읽기 전용). 컨트롤이 이를 소유하며 해당 OnChange를 사용합니다.

SaveToPNG와 SaveToSVG는 텍스트가 잘못되었으면 파일을 만들기 전에 Symbol.LastError와 함께 예외를 발생시킵니다.

색

TAlphaColor는 $AARRGGBB이고, TsgcPDFBarcodeSymbol의 메서드는 $RRGGBB를 받으며 최상위 바이트를 무시합니다. 따라서 Color와 BarColor는 바이트를 바꾸지 않고 그대로 심볼의 DrawToPDF, SaveToSVG, SaveToPNG에 전달됩니다.

예제

```
// .NET has no FireMonkey barcode control: the WinForms TsgcPDFBarcodeControl is used
private TsgcPDFBarcodeControl FQRCode;

public void CreateQRCode()
{
  FQRCode = new TsgcPDFBarcodeControl();
  FQRCode.Parent = this;
  FQRCode.Dock = DockStyle.Fill;
  FQRCode.Symbology = TsgcPDFBarcodeType.bcQRCode;
  FQRCode.QRErrorLevel = TsgcPDFQRErrorLevel.qrecH;
  FQRCode.Text = "https://www.esegece.com";
  FQRCode.BarColor = Color.Navy;
  FQRCode.OnError = OnBarcodeError;
}

void OnBarcodeError(object Sender, string AError)
{
  Text = AError;
}

public void SaveQRCode()
{
  if (FQRCode.IsValid())
    FQRCode.SaveToPNG(Path.Combine(Environment.GetFolderPath(
      Environment.SpecialFolder.MyDocuments), "qrcode.png"), 8);
}

public void AddQRCodeToPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // Color.ToArgb() is 0xAARRGGBB and DrawToPDF ignores the top byte
    FQRCode.Symbol.DrawToPDF(oPDF.AddPage(595, 842), 50, 600, 200, 200,
      (uint)FQRCode.BarColor.ToArgb(), (uint)FQRCode.BackColor.ToArgb());
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- TsgcPDFBarcodeControl
- TsgcPDFBarcodeSymbol
- macOS, iOS, Android

TsgcPDFBarcodeSymbol

TsgcPDFBarcodeSymbol(유닛 sgcPDF_Barcode_Symbol)는 TsgcPDFBarcodeControl과 TsgcPDFBarcodeFMX 뒤에 있는, 프레임워크에 의존하지 않는 바코드입니다. TsgcPDFBarcodeGenerator의 인코더로 바코드 하나를 인코딩하고, 모듈 격자에 배치한 다음 SVG 텍스트, PNG 이미지 또는 PDF 페이지의 벡터 사각형으로 기록합니다. VCL, FMX, GDI를 사용하지 않으므로 core 패키지의 일부이며 서비스, 콘솔 애플리케이션, Linux 64비트에서도 동작합니다.
.NET에서 TsgcPDFBarcodeSymbol은 **esegece.sgcPDF** 패키지의 클래스이며 Windows Forms를 사용하지 않으므로 서비스와 Linux, macOS에서도 동작합니다. 색은 `uint` 값 0xRRGGBB이고, Runs()는 TsgcPDFBarcodeRun 배열을 반환하며, OnChange는 TNotifyEvent 속성이고, TsgcPDFBarcodeErrorEvent는 델리게이트 `void (object Sender, string AError)`이며, sgcPDFBarcodeAxisPos와 CS_SGC_PDF_BARCODE 상수는 클래스 `sgcPDF_Barcode_Symbol`의 정적 멤버입니다.

모듈 격자

- 격자에는 회전 후의 심볼 전체가 들어 있으며, 여백 영역과 ShowText를 사용하는 선형 심볼로지의 경우 사람이 읽을 수 있는 줄의 띠도 포함됩니다.
- 텍스트 띠는 높이가 12모듈이며 회전 전에는 막대 아래에 놓입니다. 텍스트는 높이가 8모듈이고, 기준선은 막대 쪽 띠 가장자리에서 10모듈 떨어져 있으며, 띠를 따라 가운데에 배치됩니다. 제어 문자는 공백으로 표시됩니다.
- BarHeight가 0인 선형 심볼은 ModuleHeight와 SVG 및 PNG 출력에서 막대를 50모듈로 셉니다. 컨트롤과 DrawToPDF에서는 대신 막대가 사용 가능한 높이를 가득 채웁니다.
- Rotation은 심볼을 시계 방향으로 90도 단위 회전합니다.
- 색은 $RRGGBB이고 최상위 바이트는 무시되므로 FireMonkey의 TAlphaColor는 그대로 전달되지만, VCL TColor는 빨강과 파랑 바이트를 서로 바꿔야 합니다.

공개 속성

속성	설명
Symbology: TsgcPDFBarcodeType	심볼로지: bcCode128(기본값), bcCode39, bcEAN13, bcEAN8, bcUPCA, bcITF, bcCodabar, bcGS1_128 또는 2D 심볼로지인 bcQRCode, bcDataMatrix, bcPDF417.
Text: string	인코딩할 데이터. QR Code, DataMatrix, PDF417은 텍스트의 UTF-8 바이트를 인코딩합니다. 선형 심볼로지는 자체 문자 집합만 허용하며, GS1-128은 괄호로 묶은 애플리케이션 식별자를 받습니다.
QuietZone: Integer	모듈 단위의 밝은 여백. -1(기본값)은 심볼로지의 표준값을 사용합니다. 선형 심볼은 좌우로 10, QR Code는 4, DataMatrix는 1, PDF417은 2이며, 모두 네 변에 적용됩니다.
BarHeight: Integer	선형 심볼의 막대 높이(모듈 단위). 0(기본값)은 사용 가능한 높이를 모두 채웁니다.
ShowText: Boolean	선형 심볼의 막대 아래에 사람이 읽을 수 있는 줄을 그립니다. 기본값은 True입니다. 2D 심볼에는 텍스트 줄이 없습니다.
Rotation: TsgcPDFBarcodeRotation	br0(기본값), br90, br180, br270: 시계 방향 90도 단위 회전. 텍스트 줄은 심볼과 함께 회전합니다.
QRErrorLevel: TsgcPDFQRErrorLevel	QR Code 오류 정정: qrecL, qrecM(기본값), qrecQ, qrecH.
QRMode: TsgcPDFQRMode	QR Code 인코딩 모드: qrmByte(기본값), qrmAuto, qrmNumeric, qrmAlphanumeric.
QRVersion: Integer	0(기본값)은 맞는 가장 작은 QR Code 버전을 고르고, 1에서 40은 버전을 고정합니다.

속성	설명
DataMatrixShape: TsgcPDFDataMatrixShape	dmsSquare(기본값), dmsRectangle, dmsAuto(모든 모양 중 가장 작은 심볼).
PDF417SecurityLevel: Integer	PDF417 보안 수준 0에서 8. -1(기본값)은 자동입니다.
PDF417Columns: Integer	PDF417 데이터 열 수 1에서 30. 0(기본값)은 자동입니다.
PDF417RowHeight: Integer	PDF417 행의 높이(모듈 단위). 기본값은 3입니다.
OnChange: TNotifyEvent	속성이 변경되었습니다. 바코드 컨트롤은 다시 그리기 위해 Symbol의 OnChange를 사용하므로 그곳에서 교체하지 마십시오.

공개 메서드와 속성

멤버	설명
Encode: Boolean	마지막 호출 이후 속성이 변경되었으면 심볼을 인코딩하고, 텍스트가 올바르면 True를 반환합니다. 다른 메서드는 필요할 때 이를 호출합니다.
LastError: string	텍스트를 인코딩할 수 없는 이유.
ModuleWidth, ModuleHeight: Integer	회전된 격자의 크기(모듈 단위). 여백 영역과 텍스트 띠를 포함합니다. 텍스트가 잘못되었으면 0입니다.
ModuleAt(aCol, aRow: Integer): Boolean	회전된 격자의 aCol, aRow에 있는 모듈이 어두우면 True.
Runs: TsgcPDFBarcodeRuns	어두운 모듈을 TsgcPDFBarcodeRun 사각형(Col, Row, Cols, Rows)으로 반환합니다. 행별로, 그다음 행 사이에서 병합되며 행과 열 순서로 정렬됩니다. 텍스트가 잘못되었으면 비어 있습니다.
TextBounds(out aCol, aRow, aCols, aRows: Integer): Boolean	회전된 격자에서 텍스트 줄의 띠. 심볼이 텍스트를 그리지 않으면 False: 2D 심볼로지, ShowText 끔 또는 잘못된 텍스트.
FitModuleSize(aWidth, aHeight: Integer): Integer	심볼이 aWidth x aHeight 픽셀에 들어가는 가장 큰 정수 모듈 크기이며, 모듈당 1픽셀도 들어가지 않으면 0입니다. BarHeight가 0인 선형 심볼에서는 막대가 높이를 채우므로 막대를 가로지르는 방향으로는 텍스트 띠와 막대 1모듈만 계산됩니다.
LayoutAxes(aFit: TsgcPDFBarcodeFit; aModuleSize, aLeft, aTop, aWidth, aHeight: Integer; out aX, aY: TsgcPDFBarcodeAxis)	상자 안 격자의 전체 픽셀 레이아웃으로, 컨트롤이 그릴 때 사용하는 것입니다. 참조: 사용자 지정 그리기.
SaveToSVG(const aFileName: string; aModuleSize: Integer; aFore, aBack: Cardinal; const aFontName: string), SaveToSVG(aStream: TStream; ...)	심볼을 모듈당 aModuleSize 단위의 SVG로 기록합니다. aBack 위에 aFore로 그립니다. 텍스트 줄은 글꼴 패밀리 aFontName을 사용합니다.
SaveToPNG(const aFileName: string; aModuleSize: Integer; aFore, aBack: Cardinal; const aFontName: string), SaveToPNG(aStream: TStream; ...)	심볼을 모듈당 aModuleSize 픽셀의 PNG 이미지로, 순수 Pascal 래스터 캔버스를 통해 기록합니다. 텍스트 줄은 운영 체제의 TrueType 글꼴 aFontName을 사용하며, 설치되어 있지 않으면 산세리프 글꼴을 사용합니다. 사용 가능한 글꼴이 없으면 PNG에 텍스트 줄이 없습니다.
DrawToPDF(aPage: TObject; aX, aY, aWidth, aHeight: Double; aFore, aBack: Cardinal; aFont: TsgcPDFStandardFont = sfHelvetica)	심볼을 PDF 페이지에 그립니다. 참조: PDF 페이지에 그리기.

멤버	설명
Assign(Source: TPersistent)	다른 심볼의 게시된 속성을 복사합니다.

SaveToSVG와 SaveToPNG는 aModuleSize가 1보다 작으면 예외를 발생시키고, 텍스트가 잘못되었으면 LastError와 함께 예외를 발생시키며, 두 경우 모두 파일을 만들기 전에 발생합니다.

PDF 페이지에 그리기

DrawToPDF는 심볼을 벡터 그래픽으로 그리므로 PDF 페이지에는 바코드 컨트롤이 보여 주는 것과 똑같은 내용이 표시됩니다. 컨트롤의 속성을 TsgcPDFBarcodeOptions에 직접 대응시키는 작업을 대신합니다. 회전, 여백 영역, 막대 높이, 텍스트 줄은 심볼에서 가져옵니다. 데이터에서 바로 바코드를 그리는 가장 짧은 방법은 여전히 TsgcPDFBarcodeGenerator.DrawBarcode입니다.

매개변수	설명
aPage: TObject	TsgcPDFPage 또는 그 콘텐츠 빌더인 `TsgcPDFPage.Content`. DrawBarcode와 같습니다. 그 밖의 객체는 예외를 발생시킵니다.
aX, aY: Double	상자의 왼쪽 아래 모서리(포인트 단위). PDF의 y축은 위쪽을 향합니다.
aWidth, aHeight: Double	상자의 크기(포인트 단위). 빈 상자는 예외를 발생시킵니다.
aFore: Cardinal	막대와 텍스트 줄의 색($RRGGBB).
aBack: Cardinal	여백 영역을 포함한 심볼 전체 아래에 칠하는 색($RRGGBB).
aFont: TsgcPDFStandardFont	텍스트 줄에 쓰는 표준 PDF 글꼴. 기본값은 sfHelvetica입니다.

- 크기: 심볼은 정수 포인트에 한정되지 않고 상자에 들어가는 가장 큰 모듈 크기를 취하며 상자 안에서 가운데에 배치됩니다. BarHeight가 0인 선형 심볼은 FitModuleSize와 LayoutAxes처럼 막대를 가로지르는 방향으로 상자를 가득 채웁니다.
- 출력: aBack 사각형, 그다음 Runs의 각 항목마다 aFore로 채운 사각형 하나, 따라서 모든 모듈 가장자리가 모듈 격자에 정확히 놓이고, 그다음 텍스트 줄이 이어지며, 모두 저장된 그래픽 상태 안에서 이루어집니다.
- 텍스트: 텍스트 줄은 aFont로 8모듈 크기로 그려지고, 기준선은 막대 쪽 띠 가장자리에서 10모듈 떨어지며, 띠의 가운데에 배치되고, 심볼과 함께 시계 방향으로 회전합니다.
- 오류: 잘못된 텍스트는 LastError와 함께 예외를 발생시키고 아무것도 그리지 않습니다.

예제

GS1-128 라벨을 PDF 페이지에 두 번 그리고(두 번째는 90도 회전), SVG와 PNG로 저장하는 예제입니다. 폼도 컨트롤도 사용하지 않습니다.

```
static void CreateShippingLabel(string AFileName)
{
  TsgcPDFBarcodeSymbol oSymbol = new TsgcPDFBarcodeSymbol();
  oSymbol.Symbology = TsgcPDFBarcodeType.bcGS1_128;
  oSymbol.Text = "(01)09501101530003(17)261231";
  oSymbol.BarHeight = 30;
  if (!oSymbol.Encode())
    throw new Exception(oSymbol.LastError);
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    // black on white in a 300 x 90 points box, bottom left corner 50, 700
    oSymbol.DrawToPDF(oPage, 50, 700, 300, 90, 0x000000, 0xFFFFFF);
    // a quarter turn clockwise, navy, the text line in Courier
    oSymbol.Rotation = TsgcPDFBarcodeRotation.br90;
```

```
    oSymbol.DrawToPDF(oPage.Content, 450, 350, 90, 300, 0x000080, 0xFFFFFF,
      TsgcPDFStandardFont.sfCourier);
    oPDF.SaveToFile(AFileName);
  }
  // the same symbol as SVG (2 units per module) and PNG (3 pixels)
  oSymbol.Rotation = TsgcPDFBarcodeRotation.br0;
  oSymbol.SaveToSVG(Path.ChangeExtension(AFileName, ".svg"), 2, 0x000000, 0xFFFFFF,
    "Arial");
  oSymbol.SaveToPNG(Path.ChangeExtension(AFileName, ".png"), 3, 0x000000, 0xFFFFFF,
    "Arial");
}
```

사용자 지정 그리기

LayoutAxes는 왼쪽 위 모서리가 aLeft, aTop이고 크기가 aWidth x aHeight 픽셀인 상자에 대해, 컨트롤이 그릴 때 사용하는 전체 픽셀 레이아웃을 계산합니다. aFit이 모듈 크기를 정하고(bfModuleSize에서는 aModuleSize 픽셀) 심볼은 가운데에 배치되며, BarHeight가 0인 선형 심볼은 막대를 가로지르는 방향으로 상자를 가득 채웁니다. 축마다 TsgcPDFBarcodeAxis 하나를 반환하고, `sgcPDFBarcodeAxisPos(aAxis, aIndex)`는 격자선 aIndex(0에서 Count)의 픽셀을 반환합니다. 이웃한 연속 구간은 가장자리를 공유하므로 모듈 사이에 틈이 생기지 않습니다.

```
static void PaintSymbol(TsgcPDFBarcodeSymbol ASymbol, Graphics AGraphics, Rectangle ARect)
{
  if (!ASymbol.Encode())
    return;
  // the largest whole module size that fits, centred in ARect
  TsgcPDFBarcodeAxis vX, vY;
  ASymbol.LayoutAxes(TsgcPDFBarcodeFit.bfFitInteger, 0, ARect.Left, ARect.Top,
    ARect.Width, ARect.Height, out vX, out vY);
  // pixel i covers [i, i + 1), so whole pixel rectangles fill exactly their pixels
  AGraphics.PixelOffsetMode = PixelOffsetMode.Half;
  TsgcPDFBarcodeRun[] vRuns = ASymbol.Runs();
  for (int i = 0; i < vRuns.Length; i++)
  {
    int vLeft = sgcPDF_Barcode_Symbol.sgcPDFBarcodeAxisPos(vX, vRuns[i].Col);
    int vTop = sgcPDF_Barcode_Symbol.sgcPDFBarcodeAxisPos(vY, vRuns[i].Row);
    int vRight = sgcPDF_Barcode_Symbol.sgcPDFBarcodeAxisPos(vX, vRuns[i].Col + vRuns[i].Cols);
    int vBottom = sgcPDF_Barcode_Symbol.sgcPDFBarcodeAxisPos(vY, vRuns[i].Row + vRuns[i].Rows);
    AGraphics.FillRectangle(Brushes.Black, vLeft, vTop, vRight - vLeft, vBottom - vTop);
  }
}
```

형식과 상수

이름	설명
TsgcPDFBarcodeRotation	br0, br90, br180, br270: 시계 방향 90도 단위 회전.
TsgcPDFBarcodeFit	bfFitInteger, bfModuleSize, bfStretch: 컨트롤과 LayoutAxes가 모듈 크기를 정하는 방식.
TsgcPDFBarcodeRun, TsgcPDFBarcodeRuns	회전된 격자에서 어두운 모듈로 이루어진 사각형(Col, Row, Cols, Rows)과 Runs가 반환하는 배열.
TsgcPDFBarcodeAxis	한 축의 격자선이 정수 픽셀에 대응하는 방식: Origin, Module(모듈당 픽셀 수), Count(모듈 수). Module이 0(bfStretch)이면 Span 픽셀이 Count 모듈에 분배됩니다. FillStart, FillCount, FillPixels는 상자를 채우는 띠, 즉 BarHeight가 0인 선형 심볼의 막대를 설명합니다.
sgcPDFBarcodeAxisPos(const aAxis: TsgcPDFBarcodeAxis; aIndex: Integer): Integer	LayoutAxes 축에서 격자선 aIndex의 픽셀.

이름	설명
TsgcPDFBarcodeErrorEvent	`procedure(Sender: TObject; const aError: string) of object` . 두 바코드 컨트롤의 OnError 이벤트입니다.
CS_SGC_PDF_BARCODE_TEXT_BAND, CS_SGC_PDF_BARCODE_TEXT_SIZE, CS_SGC_PDF_BARCODE_TEXT_BASELINE	모듈 단위의 텍스트 줄: 띠 높이 12, 글꼴 크기 8, 기준선 10.
CS_SGC_PDF_BARCODE_BAR_HEIGHT	50. BarHeight가 0인 선형 심볼의 ModuleHeight, SVG, PNG에서 막대 높이(모듈 단위).

참고 항목

- TsgcPDFBarcodeControl
- TsgcPDFBarcodeFMX
- 바코드
- 2D 바코드와 GS1-128

Linux

sgcPDF 코어는 Linux 64비트에서 컴파일되고 실행됩니다(Delphi 10.2 Tokyo 이상, Linux64 플랫폼). 일반적인 서버 작업, 즉 PDF 문서 생성, 편집, 병합, 채우기, 암호화, 읽기, 그리고 웹 서비스나 배치 작업에서의 페이지 이미지 렌더링을 포함합니다.
.NET에서는 esegece.sgcPDF 패키지가 .NET 6 이상에서 Linux에서 실행되며, Delphi 코어 유닛과 동일한 기능을 제공합니다. esegece.sgcPDF.WinForms 패키지는 Windows 전용입니다. .NET API 차이점을 참조하십시오.

Linux에서 사용 가능한 기능

- TsgcPDFDocument, TsgcPDFEditor, 그리고 파서와 라이터입니다. 증분 저장, 압축된 객체 스트림, 2GB가 넘는 파일도 포함합니다.
- 텍스트, TrueType과 OpenType 폰트(AddTrueTypeFontFromFile로 파일에서, 또는 시스템 폰트 폴더에서 로드), 셰이핑, 표, 바코드, 주석, 북마크, 레이어, 폼, 첨부 파일, 태그된 PDF, PDF/A 출력과 검증입니다.
- 암호 암호화(RC4와 AES), 마스킹, 옵티마이저입니다.
- 문자 위치를 포함한 텍스트 추출, TsgcPDFImageExtractor, 모든 이미지 디코더입니다.
- TsgcPDFRendererRaster: 페이지를 PNG나 BMP로, 페이지 픽셀도 지원합니다.
- TsgcPDFCanvas와 TsgcPDFHTMLRenderer입니다.
- OnExternalSign 이벤트를 통한 전자 서명입니다(예를 들어 sgcSign이나 자체 HSM 사용). 전자 서명을 참조하십시오.

Windows 전용

- TsgcPDFRendererGDI, TsgcPDFViewer, TsgcPDFExporter, 그리고 VCL 컨트롤인 TsgcPDFViewerControl, TsgcPDFThumbnailView, TsgcPDFOutlineView입니다.
- TsgcPDFEMFPlayer와 TsgcPDFVCLCanvasHelper입니다.
- PFX 파일이나 Windows 인증서 저장소의 인증서를 이용한 서명과 검증, 그리고 Windows CryptoAPI를 사용하는 인증서(공개 키) 암호화입니다. Linux에서는 이러한 호출이 예외를 발생시킵니다.
- sgcPDF_Fonts_CJK의 저수준 CJK 클래스입니다(CJK 텍스트 자체는 AddTrueTypeFontFromFile로 동작합니다).

Linux에서의 폰트

`AddTrueTypeFont`는 /usr/share/fonts와 /usr/local/share/fonts에서 설치된 폰트를 찾습니다. 래스터 렌더러는 ~/.fonts와 ~/.local/share/fonts, 그리고 `sgcPDFRasterAddFontDirectory`로 추가한 폴더도 검사합니다. 최소 구성의 서버나 컨테이너는 폰트가 전혀 없는 경우가 많습니다. fonts-dejavu-core, fonts-liberation, fonts-noto 같은 패키지를 설치하거나, 애플리케이션과 함께 폰트를 배포하고 `AddTrueTypeFontFromFile`을 사용하십시오. 폰트가 임베드된 문서는 설치된 폰트가 없어도 올바르게 렌더링됩니다.

예제: 서버 측 썸네일

Windows와 동일한 코드입니다.

```
static void RenderPageToPNG(string APDFFile, string APNGFile, int ADPI)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    TsgcPDFPage oPage = oPDF.GetPage(0);
    TsgcPDFRendererRaster oRenderer = new TsgcPDFRendererRaster();
    // page size in points (1/72 inch) to pixels
    oRenderer.RenderPageToPNG(oPDF.Parser, oPage.PageDict,
      (int)Math.Round(oPage.Width * ADPI / 72), (int)Math.Round(oPage.Height * ADPI / 72),
      APNGFile);
  }
}
```

참고 항목

- macOS, iOS, Android
- TsgcPDFRendererRaster
- 설치

macOS, iOS, Android

Delphi XE5부터 sgcPDF는 macOS, iOS, Android용 FireMonkey 애플리케이션과 Windows, Linux(FMXLinux)에서 사용할 수 있습니다. Linux와 동일한 비Windows 유닛에 더해 FireMonkey 뷰어도 사용할 수 있습니다.
sgcPDF .NET에는 FireMonkey 유닛이 없습니다. esegece.sgcPDF 패키지는 Linux에서와 마찬가지로 .NET 6 이상에서 macOS에서 실행됩니다. .NET API 차이점을 참조하십시오.

기능	유닛	참고 사항
생성, 편집, 읽기, 추출	sgcPDF 코어 유닛	Linux와 동일합니다. 모바일 플랫폼은 시스템 폰트를 파일로 이식성 있게 제공하지 않으므로, 애플리케이션과 함께 배포한 파일에서 폰트를 로드하십시오(AddTrueTypeFontFromFile).
뷰어 컨트롤	sgcPDF_ViewerFMX	TsgcPDFViewerFMX: 스크롤, 핀치 확대/축소, 너비나 페이지에 맞추기, 링크입니다.
페이지 이미지	sgcPDF_Renderer_FMX, sgcPDF_Renderer_Raster	TsgcPDFRendererFMX는 FMX TBitmap으로 렌더링하고, TsgcPDFRendererRaster는 PNG, BMP, 픽셀로 렌더링합니다.
서명	sgcPDF_Signatures	OnExternalSign을 통해서만 가능합니다(CryptoAPI 없음).

설치

팔레트에서 TsgcPDFViewerFMX를 사용하려면 코어 패키지에 더해 FMX 패키지(sgcPDFFMXDxx와 dclsgcPDFFMXDxx)를 설치하십시오. 모바일 대상에서는 IDE가 Source 폴더의 유닛을 애플리케이션에 컴파일하므로, 사용하는 각 플랫폼(Android 32비트 및 64비트, iOS Device 64비트, macOS 64비트 및 ARM 64비트, Linux 64비트)의 라이브러리 경로에 sgcPDF Source 폴더를 추가하십시오.

파일 배포하기

Deployment Manager를 사용하여 PDF 파일과 폰트를 애플리케이션에 추가하고, 예를 들어 `TPath.Combine(TPath.GetDocumentsPath, 'sample.pdf')`(유닛 System.IOUtils)로 문서 폴더에서 여십시오.

참고 항목

- TsgcPDFViewerFMX
- Linux
- 렌더러 아키텍처

C++Builder

sgcPDF는 런타임 및 디자인 타임 패키지로 C++Builder 2007부터 C++Builder 13까지를 지원합니다. Win32 플랫폼은 모든 버전에서, Win64는 C++Builder XE6부터 지원합니다(C++Builder 13의 Win64 Modern 플랫폼 포함).

패키지

Packages 폴더에는 각 C++Builder 버전마다 코어 런타임 패키지 **sgcPDFCxx.cbproj**, VCL 런타임 패키지 **sgcPDFVCLCxx.cbproj**, 디자인 타임 패키지 **dclsgcPDFCxx.cbproj**가 있습니다. 예를 들어 sgcPDFC13.cbproj, sgcPDFVCLC13.cbproj, dclsgcPDFC13.cbproj입니다. 두 런타임 패키지를 빌드한 다음 디자인 타임 패키지를 빌드하고 설치하십시오. 빌드하면 모든 유닛의 .hpp 헤더와 .lib 또는 .a 임포트 파일이 생성됩니다.

C++에서 sgcPDF 사용하기

- 사용하는 각 유닛의 헤더를 포함하십시오. 예를 들어 sgcPDF_Document.hpp, sgcPDF_Fonts.hpp, sgcPDF_Renderer_Raster.hpp입니다.
- 클래스와 함수는 Delphi와 동일한 이름을 가지며, 해당 유닛의 네임스페이스 안에 있습니다(예: `Sgcpdf_document::TsgcPDFDocument`). 대부분의 헤더는 생성된 using 지시문을 통해 전역 네임스페이스에도 노출됩니다.
- `new`로 컴포넌트를 생성하고 `delete`로 해제하십시오. TComponent에서 파생된 컴포넌트는 생성자에서 소유자를 받습니다(없으면 NULL을 전달).
- WideString과 AnsiString 매개변수는 같은 이름의 C++Builder 문자열 클래스를 받으며, TBytes는 `System::DynamicArray<System::Byte>`입니다.

헤더는 메서드 이름과 충돌하는 Windows 매크로(DrawText, FindText, GetObject, GetCharWidth, CreateWindow, CertNameToStr)를 정의 해제하므로, `TsgcPDFTextExtractor::FindText` 같은 메서드를 정상적으로 호출할 수 있습니다.

참고 항목

- 설치

첨부 파일

PDF는 보고서의 원본 데이터, 스프레드시트, 인보이스의 XML 같은 다른 파일을 내부에 담을 수 있습니다. **TsgcPDFDocument.Attachments**(클래스 TsgcPDFAttachments, 유닛 sgcPDF_Attachments)는 이렇게 임베드된 파일을 추가, 나열, 추출, 삭제합니다. 이 파일들은 카탈로그의 `/EmbeddedFiles` 이름 트리에 기록되며, 관계 정보와 함께 PDF/A-3에서 요구하는 카탈로그 `/AF` 배열에도 기록됩니다.

파일 추가

```
static void CreateWithAttachments(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.AddPage(595, 842).DrawText("Report with attachments", 50, 780,
      TsgcPDFStandardFont.sfHelveticaBold, 18);
    // from bytes: name, data, MIME type, description, relationship
    oPDF.Attachments.AddFromBytes("data.csv", Encoding.ASCII.GetBytes("id;qty\r\n1;10"),
      "text/csv", "Source data of the report", TsgcPDFAFRelationship.afrSource);
    // from a file on disk, the file name becomes the attachment name
    TsgcPDFAttachment oFile = oPDF.Attachments.AddFromFile(Dir + "notes.txt", "text/plain",
      "Notes");
    oFile.Relationship = TsgcPDFAFRelationship.afrSupplement;
    // the viewer opens the attachments panel
    oPDF.Attachments.ShowAttachmentsPanel = true;
    oPDF.SaveToFile(AFileName);
  }
}
```

메서드	설명
AddFromBytes(AName; AData; AMimeType; ADescription; ARelationship)	AName이라는 이름으로 AData를 임베드합니다. AName과 AData만 필수입니다.
AddFromStream(AName; AStream; ...)	AStream의 나머지 부분을 임베드합니다.
AddFromFile(AFileName; AMimeType; ADescription; ARelationship)	디스크의 파일을 임베드합니다. 첨부 파일 이름은 원본 파일 이름을 따릅니다.
Delete(AIndex), Clear, IndexOf(AName)	첨부 파일을 제거하거나 이름으로 찾습니다.

이름은 고유해야 합니다. 같은 이름의 파일을 두 번째로 추가하면 예외가 발생합니다. 모든 첨부 파일은 사양이 권장하는 대로 크기, 생성 및 수정 날짜, MD5 체크섬을 저장합니다.

관계(AFRelationship)

ARelationship 또는 TsgcPDFAttachment의 `Relationship` 속성은 파일이 문서와 어떤 관계인지를 리더에 알려줍니다(PDF 2.0과 PDF/A-3).

값	의미
afrSource	문서 내용의 원본 소스입니다. 예를 들어 보고서를 만든 원본 스프레드시트입니다.
afrData	문서를 작성하는 데 사용된 데이터입니다. 예를 들어 차트의 값입니다.
afrAlternative	콘텐츠의 대체 표현입니다. 예를 들어 전자 인보이스의 XML입니다.

값	의미
afrSupplement	보충 표현입니다. 예를 들어 기계가 읽을 수 있는 사본입니다.
afrEncryptedPayload	암호화된 페이로드 문서입니다.
afrFormData	문서의 폼 필드와 연관된 데이터입니다.
afrSchema	연관된 객체의 스키마 정의입니다.
afrUnspecified	관계를 지정하지 않음(기본값).

읽기와 추출

문서를 로드하면 Attachments 목록이 자동으로 채워집니다. 각 TsgcPDFAttachment는 Name, Description, MimeType, Relationship, CreationDate, ModDate, Size를 제공하며, `GetBytes`, `SaveToStream` 또는 `SaveToFile`로 데이터를 반환합니다.

```
static void ListAttachments(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    for (int i = 0; i < oPDF.Attachments.Count; i++)
      Console.WriteLine(oPDF.Attachments[i].Name + " " + oPDF.Attachments[i].Size +
        " bytes " + oPDF.Attachments[i].MimeType);
    int vIndex = oPDF.Attachments.IndexOf("data.csv");
    if (vIndex >= 0)
      oPDF.Attachments[vIndex].SaveToFile(Dir + "extracted.csv");
    // remove one attachment and save a copy without it
    oPDF.Attachments.Delete(oPDF.Attachments.IndexOf("notes.txt"));
    oPDF.SaveToFile(Dir + "attachments_deleted.pdf");
  }
}
```

첨부 파일과 PDF/A

레벨	규칙
PDF/A-1b	임베드된 파일이 허용되지 않습니다. 파일을 추가하면 예외가 발생합니다.
PDF/A-2b	PDF/A 문서 자체인 파일만 허용됩니다. 추가하기 전에 `Attachments.PDFACompliantFiles` 를 True로 설정하여 확인하십시오. 그렇지 않으면 추가 시 예외가 발생합니다.
PDF/A-3b	모든 파일 형식을 허용하며, 각 파일은 카탈로그 /AF 배열에 자신의 AFRelationship을 가집니다.

`ShowAttachmentsPanel`을 True로 설정하면 `/PageMode /UseAttachments`가 기록되어 뷰어가 첨부 파일 패널을 열도록 합니다. 첨부 파일은 암호화된 문서와 SaveIncremental에서도 유지됩니다.

참고 항목

- Factur-X 및 ZUGFeRD 인보이스
- PDF/A
- TsgcPDFDocument

Factur-X 및 ZUGFeRD 인보이스

Factur-X(프랑스)와 ZUGFeRD 2(독일)는 동일한 하이브리드 전자 인보이스 형식입니다. 사람이 읽을 수 있는 PDF/A-3 문서에, 회계 소프트웨어를 위한 구조화된 인보이스(UN/CEFACT Cross Industry Invoice XML)가 임베드되어 있습니다. XRechnung은 동일한 컨테이너를 사용하되 파일 이름만 다릅니다.
sgcPDF는 PDF 컨테이너를 생성합니다. 인보이스 XML 자체는 애플리케이션(또는 ERP)에서 생성하여 바이트로 **Attachments.AddFacturX**에 전달합니다.

```
static void CreateFacturX(string AInvoiceXML, string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // the invoice XML (CII) is created by your application
    byte[] vXML = File.ReadAllBytes(AInvoiceXML);
    oPDF.Info.Title = "Invoice 2026-001";
    // PDF/A requires embedded fonts
    int vFont = oPDF.AddTrueTypeFont("Arial", false, false);
    oPDF.AddPage(595, 842).DrawTextTT("Invoice 2026-001", 50, 780, vFont, 18);
    // embeds factur-x.xml with the relationship of the profile, adds the
    // Factur-X XMP extension schema and saves the document as PDF/A-3b
    oPDF.Attachments.AddFacturX(vXML, TsgcPDFFacturXProfile.fxpEN16931);
    oPDF.SaveToFile(AFileName);
  }
}
```

AddFacturX의 동작

- XML을 MIME 형식 text/xml로 **factur-x.xml**(fxpXRechnung의 경우 **xrechnung.xml**, 또는 전달한 AFileName)이라는 이름으로 임베드합니다.
- 프로파일이 요구하는 관계를 설정합니다. 완전한 인보이스가 아닌 MINIMUM과 BASIC WL은 **Data**, BASIC, EN 16931, EXTENDED, XRECHNUNG은 **Alternative**입니다.
- 메타데이터에 Factur-X XMP 확장 스키마(fx:DocumentType, fx:DocumentFileName, fx:Version, fx:ConformanceLevel)를 추가합니다.
- PDF/A 레벨이 없는 문서를 **PDF/A-3b**로 전환합니다. Factur-X는 PDF/A-3을 요구하므로, 이미 PDF/A-1b나 PDF/A-2b로 설정된 문서에서는 예외가 발생합니다.
- 이전 인보이스 XML을 대체합니다. 문서에는 하나의 인보이스만 담을 수 있습니다.

프로파일(TsgcPDFFacturXProfile)	ConformanceLevel	관계
fxpMinimum	MINIMUM	Data
fxpBasicWL	BASIC WL	Data
fxpBasic	BASIC	Alternative
fxpEN16931	EN 16931	Alternative
fxpExtended	EXTENDED	Alternative
fxpXRechnung	XRECHNUNG	Alternative

선택적인 AVersion 매개변수는 fx:Version을 설정합니다(기본값 1.0). PDF/A 규칙은 문서 전체에 적용되므로, 텍스트는 임베드된 TrueType 폰트(AddTrueTypeFont와 DrawTextTT)로 그리십시오.
문서를 로드하면 `Attachments.FacturX`, `FacturXProfile`, `FacturXFileName`에서 인보이스 정보를 확인할 수 있습니다.
위 예제는 veraPDF로 검사했으며(PDF/A-3b: PASS), factur-x Python 라이브러리로 인보이스를 추출한 결과 원본과 동일했습니다.

참고 항목

- 첨부 파일
- PDF/A

증분 저장과 압축 출력

SaveIncremental

TsgcPDFDocument.SaveIncremental은 새 리비전을 추가하여 로드된 PDF를 업데이트합니다. 원본 바이트는 변경되지 않은 채로 기록되고, 그 뒤에 새로 추가되거나 변경된 객체, 원본과 같은 종류(테이블 또는 스트림)의 상호 참조 섹션, `/Prev`가 포함된 트레일러가 이어집니다. `/ID`와 `/Encrypt` 항목은 그대로 유지됩니다.
서명을 깨뜨리지 않고 서명된 문서를 편집하거나, 큰 파일에 주석이나 폼 값을 빠르게 추가하거나, 문서의 이력을 유지할 때 사용합니다.

```
static void AppendChanges(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(AFileName);
    oPDF.Info.Title = "Reviewed";
    oPDF.Annotations.AddText(0, 300, 700, 320, 720, "Checked by QA");
    // the original bytes are kept, only the changes are appended
    oPDF.SaveIncremental(AFileName);
  }
}
```

두 가지 오버로드가 있습니다. `SaveIncremental(AStream: TStream)`은 업데이트된 전체 파일을 스트림에 쓰고, `SaveIncremental(AFileName: string)`은 파일을 쓰며, 이 파일은 문서를 로드한 파일과 같을 수 있습니다. 변경 사항이 없는 문서는 원본 바이트를 그대로 생성합니다.
주석 편집도 같은 방식으로 저장됩니다. 파일의 변경된 주석은 자신의 객체 번호로 다시 기록되고, 새 주석과 답글은 새 리비전에 추가되므로 기존 모든 서명의 바이트 범위는 그대로입니다. 검증 도구는 서명이 코멘트를 허용하는 경우에 그 리비전을 허용합니다. DocMDP 권한 P=3인 인증 서명 또는 승인 서명만 있는 경우입니다. P=1 또는 P=2이면 허용되지 않는 변경으로 보고합니다. 주석을 참고하십시오.

제한 사항

다음의 경우 SaveIncremental은 예외를 발생시키며, 전체 저장(SaveToFile 또는 SaveToStream)이 필요합니다.

- 문서가 PDF에서 로드되지 않은 경우(새 문서)
- AppendDocument로 다른 문서를 추가한 경우
- 새 레이어나 새 폼 필드를 추가한 경우(기존 필드를 채우는 것은 문제없음)
- 암호화를 변경한 경우
- 암호로 연 문서를 마스킹(redaction)한 경우(TsgcPDFRedactor 참조)

CompressObjects

CompressObjects를 True로 설정하면 전체 저장 시 스트림이 아닌 객체를 압축된 객체 스트림에 기록하고, 상호 참조도 압축된 상호 참조 스트림(PDF 1.5)으로 기록합니다. 작은 객체(페이지, 주석, 폼 필드, 구조 트리)가 많은 문서는 테스트에서 39%에서 57%까지 작아졌습니다. 암호화도 지원됩니다. 객체 스트림을 허용하지 않는 PDF/A-1에서는 이 속성이 무시됩니다.

```
static void SaveCompact(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    for (int i = 1; i <= 50; i++)
      oPDF.AddPage(595, 842).DrawText("Page " + i, 50, 780,
        TsgcPDFStandardFont.sfHelvetica, 12);
    // object streams and a compressed cross-reference stream (PDF 1.5)
    oPDF.CompressObjects = true;
    oPDF.SaveToFile(AFileName);
  }
}
```

참고 항목

- 대용량 파일
- 타임스탬프와 장기 검증
- PDF 파일 최적화
- 주석

대용량 파일

sgcPDF는 모든 곳에서 64비트 파일 오프셋을 사용하므로 2GB보다 큰 파일도 읽고 쓸 수 있으며, 큰 파일을 처리하기 위해 메모리에 전체를 로드할 필요가 없습니다.

MemoryThreshold

TsgcPDFDocument.LoadFromFile은 **MemoryThreshold** 바이트(기본값 64MB)까지의 파일을 메모리로 로드합니다. 더 큰 파일은 공유 읽기로 열리며, 읽기 캐시를 통해 객체를 필요할 때 읽으므로 3GB 파일을 열어도 상호 참조와 실제로 사용하는 객체만 읽습니다. 모든 파일을 온디맨드로 읽으려면 MemoryThreshold를 0으로 설정하십시오.

```
static void OpenLargeFile(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // files larger than MemoryThreshold (default 64 MB) are read on demand
    // from the file instead of being loaded into memory
    oPDF.MemoryThreshold = 16 * 1024 * 1024;
    oPDF.LoadFromFile(AFileName);
    Console.WriteLine("Pages: " + oPDF.PageCount() + ", read on demand: " +
      oPDF.Parser.StreamBacked);
    Console.WriteLine(oPDF.ExtractTextW(0));
  }
}
```

동작

- 문서를 온디맨드로 읽을 때 `Parser.StreamBacked`는 True이며, `Parser.Size`는 파일 크기를 Int64로 나타냅니다.
- 파일은 다른 문서를 로드하거나 문서를 해제할 때까지 열려 있습니다. 그동안 다른 프로세스는 파일을 읽을 수는 있지만 삭제하거나 덮어쓸 수는 없습니다.
- SaveIncremental은 온디맨드로 읽은 문서에서도 동작하며, 동일한 파일에 쓸 수 있습니다. 원본 바이트는 블록 단위로 복사되고, 파일이 교체된 다음 파서가 새 파일을 다시 엽니다.
- LoadFromStream과 LoadFromBytes는 항상 메모리로 로드합니다. 스트림을 온디맨드로 읽으려면 파서를 직접 사용하십시오. `TsgcPDFParser.LoadFromStream(AStream, AOwnsStream)`. 파서를 사용하는 동안 스트림은 유효하고 변경되지 않은 상태로 유지되어야 합니다.
- 32비트 프로세스는 하나의 배열에 2GB를 담을 수 없습니다. 이런 소스를 메모리로 로드하면 온디맨드 읽기를 요구하는 예외가 발생합니다.

참고 항목

- 증분 저장과 압축 출력
- 오류 처리 및 악의적인 입력

FDF and XFDF

FDF(ISO 32000, 12.7.8)와 XFDF(ISO 19444-1, XML) 파일은 PDF 폼 필드의 값만 담습니다. 채워진 데이터를 서버로 보내거나, 데이터베이스에서 폼을 미리 채우거나, 두 버전의 폼 사이에 값을 옮길 때 사용합니다. TsgcPDFAcroForm(`TsgcPDFDocument.AcroForm`)은 두 형식을 모두 내보내고 가져옵니다.

내보내기

```
static void ExportFormData(string APDFFile, string AFDFFile, string AXFDFFile)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    // the optional second parameter names the PDF the data belongs to
    oPDF.AcroForm.ExportFDF(AFDFFile, Path.GetFileName(APDFFile));
    oPDF.AcroForm.ExportXFDF(AXFDFFile, Path.GetFileName(APDFFile));
  }
}
```

메서드	설명
ExportFDF(AStream or AFileName; ASourceFile)	모든 필드의 값을 정규화된 이름으로 담은 FDF 파일을 씁니다. ASourceFile(선택 사항)은 /F로 기록됩니다.
ExportXFDF(AStream or AFileName; ASourceFile)	XFDF 파일(UTF-8)을 씁니다. ASourceFile(선택 사항)은 <f href>로 기록됩니다.

가져오기

```
static void ImportFormData(string APDFFile, string AXFDFFile, string AOutput)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.LoadFromFile(APDFFile);
    int vCount = oPDF.AcroForm.ImportXFDF(AXFDFFile);
    Console.WriteLine(vCount + " values set, " + oPDF.AcroForm.ImportUnknownCount +
      " unknown fields");
    // the appearances are regenerated from the imported values
    oPDF.SaveToFile(AOutput);
  }
}
```

`ImportFDF`와 `ImportXFDF`는 정규화된 이름(예: `address.city`)으로 일치하는 모든 필드의 값을 설정하고, 설정된 값의 개수를 반환합니다. 일치하는 필드가 없는 이름은 `ImportUnknownCount`에 집계됩니다. 변경된 필드의 모양은 문서를 저장할 때 다시 생성됩니다.
두 형식 모두 텍스트 값은 유니코드이며, 확인란과 라디오 버튼은 내보내기 값(예: Off 또는 Yes)을 사용하고, 다중 선택 목록 상자는 선택한 옵션마다 값을 하나씩 씁니다.

참고 항목

- 폼 값과 모양
- 대화형 폼

폼 값과 모양

```
static void CreateForm(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.AddPage(595, 842);
    oPDF.AcroForm.AddTextField("name", 0, 50, 740, 300, 760);
    oPDF.AcroForm.AddCheckBox("agree", 0, 50, 700, 65, 715);
    TsgcPDFFormField oList = oPDF.AcroForm.AddListBox("languages", 0, 50, 600, 200, 680,
      new string[] { "Delphi", "C++", "Pascal", "Python" });
    // multi-select list box: flag bit 22 (MultiSelect)
    oList.Flags = oList.Flags | 0x200000;
    oList.SetValues(new string[] { "Delphi", "Pascal" });
    // Cyrillic value: the appearance embeds a font that covers it
    oPDF.AcroForm.AppearanceFontName = "Arial";
    oPDF.AcroForm.FillField("name", "Иван Петров");
    oPDF.SaveToFile(AFileName);
  }
}
```

다중 선택 목록 상자

MultiSelect 플래그(`Flags`의 22번째 비트, 값 $200000)가 설정된 목록 상자는 여러 값을 담을 수 있습니다. `SetValues`로 값을 선택하고, `ValueCount`와 `GetValues(AIndex)`로 값을 읽으며, `Value`는 첫 번째 값을 반환합니다. 값은 /V에 배열로, 선택한 옵션의 /I 인덱스로 기록되며, FDF와 XFDF에도 유지됩니다.

유니코드 모양

표준 폰트 인코딩을 벗어난 문자(키릴 문자, 그리스 문자, CJK, 아랍어, 히브리어 등)가 포함된 필드 값의 모양은 임베드된 TrueType 또는 OpenType 폰트로 그려지며, 모양 리소스와 폼 /DR 딕셔너리에 `/SgcU`로 등록됩니다. 그 덕분에 모양을 다시 생성하지 않는 뷰어에서도 값이 올바르게 표시되고 인쇄됩니다.

속성	설명
AppearanceFontFile: string	이 값에 사용할 폰트 파일(.ttf, .otf)입니다. AppearanceFontName보다 우선합니다.
AppearanceFontName: WideString	이 값에 사용할 설치된 폰트입니다.

두 속성이 모두 비어 있으면 Arial Unicode MS, Segoe UI, Arial, Yu Gothic, MS Gothic, Microsoft YaHei, Malgun Gothic, Noto Sans 중 해당 문자를 지원하는 첫 번째 설치된 폰트가 사용됩니다. 이러한 폰트가 없는 Linux나 서버에서는 AppearanceFontFile을 설정하십시오.

NeedAppearances

`AcroForm.NeedAppearances`는 `/NeedAppearances true`를 기록하여, 파일을 열 때 뷰어가 필드 모양을 다시 생성하도록 요청합니다. 새 폼에서는 기본값이 True입니다. 로드된 문서는 파일의 값을 그대로 유지합니다. sgcPDF는 항상 자체 모양을 기록하므로 이 플래그를 무시하는 뷰어에서도 필드가 올바르게 보입니다. PDF/A 표준은 이 값을 true로 허용하지 않으므로 PDF/A 파일에서는 False로 설정하십시오.

참고 항목

- FDF와 XFDF

- 대화형 폼

PAdES 서명

PAdES(ETSI EN 319 142-1)는 eIDAS에 따른 고급 및 적격 전자 서명에 사용되는 유럽식 PDF 서명 프로파일입니다. TsgcPDFSignatureHandler는 Windows CryptoAPI로 PAdES 베이스라인 B-B 서명을 생성하고, 임의의 외부 서명자가 만든 CMS를 받아들이며, B-T, B-LT, B-LTA 레벨을 위한 문서 타임스탬프와 검증 데이터를 추가합니다.

SubFilter

SubFilter (TsgcPDFSignatureSubFilter)	기록되는 값	용도
pssPKCS7Detached (default)	/adbe.pkcs7.detached	모든 리더와 호환되는 고전적인 PDF 서명입니다.
pssCAdESDetached	/ ETSI.CAdES.detached	PAdES 베이스라인 서명입니다. CMS에는 ESS signing-certificate-v2 속성이 포함됩니다.

```
static void SignPAdES(string AInput, string AOutput, string APFXFile)
{
  using (TsgcPDFSignatureHandler oSigner = new TsgcPDFSignatureHandler())
  {
    oSigner.LoadCertificateFromPFX(APFXFile, "test123");
    // PAdES baseline: /SubFilter /ETSI.CAdES.detached
    oSigner.SubFilter = TsgcPDFSignatureSubFilter.pssCAdESDetached;
    oSigner.HashAlgorithm = TsgcPDFSignatureHashAlgorithm.shaSHA256;
    oSigner.SignerName = "John Smith";
    oSigner.Reason = "Approved";
    oSigner.Location = "Barcelona";
    oSigner.SetField("Signature1", 0, 350, 50, 550, 110);
    oSigner.SignDocument(AInput, AOutput);
  }
}
```

외부 서명자(OnExternalSign)

개인 키를 CryptoAPI에서 사용할 수 없는 경우(원격 서명 서비스, HSM, 스마트카드 미들웨어, 클라우드 키 보관소), **OnExternalSign**을 할당하십시오. 인증서를 로드할 필요가 없습니다. sgcPDF가 서명 딕셔너리와 바이트 범위를 준비하고, 이벤트가 CMS를 반환합니다.

```
static void DoExternalSign(object Sender, byte[] AData, byte[] ADigest,
  ref byte[] ASignature)
{
  // AData: the bytes covered by /ByteRange, ADigest: their SHA-256 digest.
  // Return a complete DER encoded detached CMS SignedData
  ASignature = SignWithRemoteService(AData);
}

static void SignExternal(string AInput, string AOutput)
{
  using (TsgcPDFSignatureHandler oSigner = new TsgcPDFSignatureHandler())
  {
    // no certificate is loaded: the CMS comes from the event
    oSigner.OnExternalSign = DoExternalSign;
    oSigner.SignerName = "Remote signer";
    oSigner.SetInvisible();
    oSigner.SignDocument(AInput, AOutput);
  }
```

```
}
```

매개변수	설명
AData	/ByteRange가 포함하는 바이트입니다(/Contents 플레이스홀더를 제외한 파일).
ADigest	해시를 서명하는 서비스를 위한, HashAlgorithm으로 계산한 AData의 다이제스트입니다.
ASignature	완전한 DER 인코딩된 분리형 CMS SignedData를 반환합니다. PlaceholderSize 바이트(기본값 8192, 긴 인증서 체인이나 임베드된 타임스탬프에는 값을 높여야 함) 안에 들어가야 합니다.

sgcSign으로 만든 완전한 PAdES 서명(임의의 키 공급자, 서명 타임스탬프, OCSP)에는 TsgcPDFPAdESSigner 브리지를 사용하십시오.

참고 항목

- 타임스탬프와 장기 검증
- 서명 검증
- 전자 서명
- TsgcPDFSignatureHandler

타임스탬프와 장기 검증

서명은 인증서가 유효하고 폐기 상태를 확인할 수 있는 동안에만 검증할 수 있습니다. 장기 검증(LTV)은 검증 데이터를 문서에 임베드하고 타임스탬프로 모든 것을 보호하여, 그 이후에도 서명을 검증할 수 있게 유지합니다. TsgcPDFSignatureHandler는 둘 다 증분 업데이트로 기록하므로 이전 서명은 그대로 유효합니다.

문서 타임스탬프

TimestampDocument(및 TimestampDocumentStream)는 SubFilter `/ETSI.RFC3161`을 사용하는 `/DocTimeStamp` 유형의 보이지 않는 서명을 추가합니다. 타임스탬프 토큰은 **OnTimestamp** 이벤트에서 옵니다. sgcPDF가 타임스탬프할 다이제스트를 전달하면, 사용자의 코드가 타임스탬프 기관(TSA)에 RFC 3161 요청을 보내고 응답의 TimeStampToken을 반환합니다.

```
static void DoTimestamp(object Sender, byte[] ADigest,
  TsgcPDFSignatureHashAlgorithm AHashAlgorithm, ref byte[] AToken)
{
  // send an RFC 3161 request with ADigest to your time-stamp authority and
  // return the TimeStampToken of the response (DER)
  AToken = RequestTimeStampToken(ADigest, AHashAlgorithm);
}

static void AddDocumentTimestamp(string AInput, string AOutput)
{
  using (TsgcPDFSignatureHandler oSigner = new TsgcPDFSignatureHandler())
  {
    oSigner.OnTimestamp = DoTimestamp;
    // appends a /DocTimeStamp (ETSI.RFC3161) as an incremental update
    oSigner.TimestampDocument(AInput, AOutput);
  }
}
```

OnTimestamp는 문서 타임스탬프에만 사용됩니다. CMS 안의 서명 타임스탬프(PAdES B-T)는 TsgcPDFPAdESSigner가 추가합니다.

Document Security Store(DSS)

DSS(카탈로그의 `/DSS`)는 검사기에 필요한 인증서, OCSP 응답, CRL을 담고 있으며, 문서의 모든 서명에 대한 `/VRI` 항목도 포함합니다.

```
static void AddValidationData(string AInput, string AOutput)
{
  using (TsgcPDFSignatureHandler oSigner = new TsgcPDFSignatureHandler())
  {
    // certificates of the chain, CRLs and OCSP responses (DER)
    oSigner.AddDSSCertificate(File.ReadAllBytes(Pki + "ica.der"));
    oSigner.AddDSSCertificate(File.ReadAllBytes(Pki + "root.der"));
    oSigner.AddDSSCRL(File.ReadAllBytes(Pki + "ica.crl"));
    oSigner.AddDSSCRL(File.ReadAllBytes(Pki + "root.crl"));
    // writes /DSS with a /VRI entry per signature (incremental update)
    oSigner.AddDSSDocument(AInput, AOutput);
  }
}
```

메서드	설명
AddDSSCertificate(AData)	DER 인증서(중간 CA, 루트 CA, TSA 인증서)를 추가합니다.
AddDSSOCSPResponse(AData)	DER OCSP 응답을 추가합니다.

메서드	설명
AddDSSCRL(AData)	DER 인증서 폐기 목록을 추가합니다.
ClearDSS	수집된 데이터를 지웁니다.
AddDSSDocument(AInputFile, AOutputFile), AddDSSStream(AInput, AOutput)	서명된 문서의 증분 업데이트로 DSS를 씁니다.

레벨

레벨	방법
B-B	SubFilter pssCAdESDetached로 서명합니다.
B-T	서명 타임스탬프가 있는 B-B이거나, 서명 직후 추가한 문서 타임스탬프입니다.
B-LT	체인과 폐기 데이터가 포함된 DSS를 추가합니다.
B-LTA	DSS 다음에 문서 타임스탬프를 추가합니다. 타임스탬프 인증서가 만료되기 전에 DSS와 타임스탬프를 반복하십시오.

이 페이지의 예제 출력은 pyHanko로 검증했습니다. CAdES 서명은 손상되지 않았고 유효하며 신뢰할 수 있고, 그 이후의 변경 사항은 LTA 업데이트로만 인식되며, 문서 타임스탬프도 손상되지 않았고 신뢰할 수 있습니다.

참고 항목

- PAdES 서명
- TsgcPDFPAdESSigner
- 서명 검증

서명 검증

TsgcPDFSignatureVerifier는 PDF의 모든 서명과 문서 타임스탬프를 검사합니다. `GetSignature(AIndex)`는 각각의 세부 정보를 담은 TsgcPDFSignatureInfo를 반환합니다.

```
static void VerifyAll(string AFileName)
{
  using (TsgcPDFSignatureVerifier oVerifier = new TsgcPDFSignatureVerifier())
  {
    oVerifier.LoadDocument(AFileName);
    for (int i = 0; i < oVerifier.GetSignatureCount(); i++)
    {
      oVerifier.VerifySignature(i);
      TsgcPDFSignatureInfo oSig = oVerifier.GetSignature(i);
      Console.WriteLine(oSig.FieldName + " " + oSig.SubFilter);
      Console.WriteLine("  doc time-stamp: " + oSig.IsDocTimeStamp);
      Console.WriteLine("  signer: " + oSig.CertInfo.SubjectName);
      Console.WriteLine("  signed: " + oSig.SignDate);
      Console.WriteLine("  valid: " + oSig.IsValid + ", trusted: " + oSig.IsTrusted +
        ", modified: " + oSig.IsModified);
    }
  }
}
```

속성	설명
FieldName	서명 필드의 이름입니다.
SubFilter	예를 들어 adbe.pkcs7.detached, ETSI.CAdES.detached 또는 ETSI.RFC3161입니다.
IsDocTimeStamp	문서 타임스탬프(/Type /DocTimeStamp)이면 True입니다. 해당 토큰은 포함된 바이트에 대해 검증됩니다.
SignDate	서명의 /M 항목이거나, 문서 타임스탬프인 경우 토큰의 genTime(UTC)입니다.
CertInfo	서명자(또는 TSA) 인증서의 SubjectName, IssuerName, SerialNumber, NotBefore, NotAfter, Thumbprint입니다.
IsValid	CMS 서명이 서명된 바이트 범위와 일치하는지 나타냅니다.
IsTrusted	서명자 인증서가 Windows에서 신뢰하는 루트로 체인되는지 나타냅니다. 별도로 검사되며, 알 수 없는 서명자의 유효한 서명은 유효하지만 신뢰할 수는 없습니다.
IsModified	서명이 파일 전체를 포함하지 않음을 나타냅니다. 서명 후의 증분 업데이트(폼 채우기, 주석, DSS, 타임스탬프)가 이 값을 설정합니다.
Name, Reason, Location	서명 딕셔너리의 값입니다.

VerifySignature(AIndex)는 IsValid를 반환합니다. IsSignatureTrusted(AIndex)와 IsDocumentModified(AIndex)는 나머지 두 결과를 반환합니다. 신뢰는 Windows 인증서 저장소로 평가되므로 테스트 인증서는 신뢰할 수 없는 것으로 보고됩니다. 특정 신뢰 목록에 대한 PAdES 정책 검증에는 sgcSign 같은 전용 검사기를 사용하십시오.

참고 항목

- PAdES 서명
- TsgcPDFSignatureVerifier

TsgcPDFPAdESSigner(sgcSign 브리지)

TsgcPDFPAdESSigner(유닛 **sgcPDF_Sign_sgcSign**)는 eSeGeCe의 전자 서명 라이브러리인 **sgcSign**으로 PAdES 베이스라인 서명 B-B, B-T, B-LT, B-LTA를 생성합니다. sgcPDF는 PDF 측면(필드, 모양, 증분 업데이트, DSS, 문서 타임스탬프)을 처리하고, sgcSign은 암호화 측면(PAdES가 요구하는 속성을 가진 CMS, 키 공급자, TSA와 OCSP 클라이언트)을 처리합니다.
개인 키는 sgcSign 키 공급자를 통해 접근하므로, sgcSign이 지원하는 모든 키 저장소를 사용할 수 있습니다. PFX와 PEM 파일, PKCS#11 토큰과 스마트카드, Windows 인증서 저장소, 클라우드 HSM입니다.

요구 사항

- sgcSign 라이선스, 그리고 sgcPDF 옆에 있는 sgcSign의 Source 폴더가 프로젝트의 검색 경로에 있어야 합니다.
- sgcPDF_Sign_sgcSign.pas는 선택적 유닛입니다. sgcPDF 패키지에 **포함되어 있지 않습니다**. 프로젝트에 직접 추가하십시오.

예제

```
static void SignPAdESLT(string AInput, string AOutput)
{
  TsgcPEMKeyProvider oKey = new TsgcPEMKeyProvider();
  using (TsgcPDFPAdESSigner oSigner = new TsgcPDFPAdESSigner())
  {
    // any sgcSign key provider: PEM, PFX, PKCS#11, Windows store, cloud HSM
    oKey.LoadFromFile(Pki + "signer.pem", Pki + "signer.key");
    oSigner.KeyProvider = oKey;
    oSigner.Level = TsgcPDFPAdESLevel.paBLT;
    oSigner.TSAURL = "http://timestamp.digicert.com";
    // validation material: CA certificates and CRLs (or an OCSPClient)
    oSigner.AddCertificate(File.ReadAllBytes(Pki + "ica.der"));
    oSigner.AddCertificate(File.ReadAllBytes(Pki + "root.der"));
    oSigner.AddCRL(File.ReadAllBytes(Pki + "ica.crl"));
    oSigner.AddCRL(File.ReadAllBytes(Pki + "root.crl"));
    // signature field and appearance through the sgcPDF handler
    oSigner.Handler.SetField("Signature1", 0, 350, 50, 550, 110);
    oSigner.Handler.SignerName = "John Smith";
    oSigner.Handler.Reason = "Approved";
    oSigner.SignPDFFile(AInput, AOutput);
    oSigner.KeyProvider = null;
  }
}
```

이 예제는 sgcPDF와 sgcSign 소스에 대해 Delphi 13으로 컴파일하고, 실제 공개 TSA로 실행했습니다. pyHanko는 결과를 유효한 서명 타임스탬프와 /DSS를 가진 손상되지 않고 신뢰할 수 있는 ETSI.CAdES.detached 서명으로 검증합니다.

멤버	설명
KeyProvider: IsgcKeyProvider	서명자 인증서와 개인 키를 가진 sgcSign 키 공급자입니다.
Level: TsgcPDFPAdESLevel	paBB, paBT, paBLT 또는 paBLTA입니다. 기본값은 paBB입니다.
TSAURL: string / TSAClient: IsgcTSAClient	B-T 이상에 사용할 타임스탬프 기관입니다. URL(내부 sgcSign TSA 클라이언트 사용)이나 자체 클라이언트를 지정합니다.
OCSPClient: IsgcOCSPClient	선택 사항입니다. 할당하면 B-LT를 위해 서명자 인증서의 OCSP 응답을 가져와 저장합니다.
HashAlgorithm	shaSHA256(기본값), shaSHA384 또는 shaSHA512입니다.

멤버	설명
AddCertificate, AddCRL, AddOCSPResponse, ClearValidationData	검증 자료입니다. 인증서는 CMS에도 임베드되며, CRL과 OCSP 응답은 /DSS로 들어갑니다.
Handler: TsgcPDFSignatureHandler	서명 필드, 모양, 이유, 장소, 플레이스홀더 크기입니다.
SignPDF(AInput, AOutput), SignPDFFile(AInputFile, AOutputFile)	선택한 레벨로 서명합니다.
ExtendToLTA(AInput, AOutput)	이미 서명된 PDF에 DSS와 문서 타임스탬프를 추가합니다.

브리지는 CMS signing-time 속성을 기록하지 않습니다. PAdES는 서명 딕셔너리의 /M 항목과 함께 이 속성을 사용하는 것을 금지합니다.

참고 항목

- PAdES 서명
- 타임스탬프와 장기 검증

인증서 암호화

암호 암호화는 모든 독자와 하나의 비밀을 공유합니다. 인증서(공개 키) 암호화인 `/Adobe.PubSec` 보안 핸들러는 대신 각 수신자의 X.509 인증서로 문서 키를 암호화합니다. 일치하는 개인 키를 가진 소유자만 파일을 열 수 있고, 암호가 오가지 않으며, 수신자마다 다른 권한을 받을 수 있습니다.

암호화

```
static void EncryptForRecipients(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.AddPage(595, 842).DrawText("For the finance team only", 50, 780,
      TsgcPDFStandardFont.sfHelvetica, 14);
    // every recipient gets its own permissions
    oPDF.AddEncryptionRecipient(Pki + "signer.der", TsgcPDFPermissions.ppPrint |
      TsgcPDFPermissions.ppCopy | TsgcPDFPermissions.ppExtract);
    oPDF.AddEncryptionRecipient(Pki + "tsa.pem", TsgcPDFPermissions.ppPrint);
    // emAES_256 writes /Adobe.PubSec with /SubFilter /adbe.pkcs7.s5
    oPDF.SetCertificateEncryption(TsgcPDFEncryptionMethod.emAES_256);
    oPDF.SaveToFile(AFileName);
  }
}
```

메서드	설명
AddEncryptionRecipient(ACertificate: TBytes; APermissions)	DER나 PEM 인증서로 수신자를 추가합니다.
AddEncryptionRecipient(ACertificateFile: string; APermissions)	동일하지만 .cer, .der 또는 .pem 파일에서 추가합니다.
AddEncryptionRecipientFromStore(ACertificateId; APermissions; AStoreName)	Windows 저장소의 인증서를 SHA-1 지문(16진수)이나 주체의 일부로 찾습니다. AStoreName의 기본값은 MY입니다.
SetCertificateEncryption(AMethod)	추가된 수신자에 대해 암호화를 활성화합니다. emAES_256과 emAES_128은 /SubFilter /adbe.pkcs7.s5(crypt 필터)를 기록하고, emRC4_128과 emRC4_40은 예전 방식인 /adbe.pkcs7.s4를 기록합니다.

APermissions는 TsgcPDFPermission의 집합입니다: ppPrint, ppModify, ppCopy, ppAnnotate, ppFillForms, ppExtract, ppAssemble, ppPrintHigh.

열기

```
static void OpenWithCertificate(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    // the private key of one recipient opens the document
    oPDF.SetDecryptionCertificate(Pki + "signer.pfx", "test123");
    oPDF.LoadFromFile(AFileName);
    Console.WriteLine(oPDF.ExtractTextW(0));
    Console.WriteLine("Can print: " +
      ((oPDF.Encryption.Permissions & TsgcPDFPermissions.ppPrint) != 0));
    Console.WriteLine("Can modify: " +
```

```
      ((oPDF.Encryption.Permissions & TsgcPDFPermissions.ppModify) != 0));
  }
}
```

LoadFromFile 나 LoadFromStream 전 에 `SetDecryptionCertificate(APFXFileName, APassword)`, `SetDecryptionCertificate(APFX: TBytes, APassword)` 또는 현재 사용자 저장소의 개인 키를 가진 인증서를 시도하는 `SetDecryptionCertificateFromStore(AStoreName)`으로 개인 키를 설정하십시오. 그러면 `Encryption.Permissions`에 파일을 연 수신자의 권한이 담깁니다.

플랫폼

인증서 암호화와 복호화는 Windows CryptoAPI(Windows에 포함된 crypt32.dll)를 사용하므로 Windows에서만 사용할 수 있습니다. 예제에서 생성한 파일은 첫 번째 수신자의 PFX로 pyHanko를 이용해 독립적으로 복호화했습니다.

보안 참고 사항

- 파일 키, AES 초기화 벡터, 솔트는 운영체제의 CSPRNG에서 가져옵니다(Windows는 CryptGenRandom, Linux/macOS/iOS/Android는 /dev/urandom).
- emAES_256은 PDF 2.0(ISO 32000-2)의 보안 핸들러 리비전 6을 기록합니다. 더 이상 사용되지 않는 리비전 5(Acrobat 9)의 파일도 계속 열 수 있습니다.
- 암호화는 전체 저장으로만 변경할 수 있습니다. 암호화가 변경되면 SaveIncremental은 예외를 발생시킵니다.

참고 항목

- 암호화
- TsgcPDFRedactor

TsgcPDFRedactor

TsgcPDFRedactor(유닛 sgcPDF_Redaction)는 진정한 의미의 마스킹을 수행합니다. 표시된 영역 안의 텍스트, 이미지 픽셀, 벡터 그래픽, 주석이 검은 상자로 덮이는 것이 아니라 파일에서 실제로 제거됩니다. 마스킹된 콘텐츠는 출력물에서 선택, 복사, 추출, 복구할 수 없습니다.

```
static void RedactFile(string AInput, string AOutput)
{
  using (TsgcPDFRedactor oRedactor = new TsgcPDFRedactor())
  {
    // every occurrence of a text, on all pages
    oRedactor.AddText("John Smith");
    // an area of page 0: left, bottom, right, top in points
    oRedactor.AddArea(0, 45, 735, 300, 752);
    oRedactor.AddArea(0, 50, 500, 250, 650);
    TsgcPDFRedactionOptions vOptions = oRedactor.Options;
    vOptions.OverlayText = "REDACTED";
    vOptions.FillColorR = 0;
    vOptions.FillColorG = 0;
    vOptions.FillColorB = 0;
    vOptions.RemoveMetadata = true;
    oRedactor.Options = vOptions;
    oRedactor.Apply(AInput, AOutput);
    Console.WriteLine("Text matches: " + oRedactor.TextMatches);
    Console.WriteLine("Removed glyphs: " + oRedactor.RemovedGlyphs);
    Console.WriteLine("Removed paths: " + oRedactor.RemovedPaths);
  }
}
```

표시

메서드	설명
AddArea(APageIndex; ARect: TsgcPDFRect)	페이지의 기본 사용자 공간(왼쪽 아래가 원점, /Rotate 적용 전)에서 포인트 단위의 사각형입니다.
AddArea(APageIndex; ALeft, ABottom, ARight, ATop)	좌표로 지정하는 동일한 기능입니다.
AddQuad(APageIndex; AQuad: TsgcPDFRedactionQuad)	회전된 텍스트를 위한 임의의 볼록 사각형입니다. sgcPDFRedactionQuadFromRect는 사각형을 변환합니다.
AddText(AText; ACaseSensitive; AWholeWord; APageIndex)	APageIndex가 -1이면 모든 페이지에서, 그렇지 않으면 해당 페이지에서 AText가 나타나는 모든 위치입니다. 일치 항목은 텍스트 줄 안에서 검색되며, 연속된 공백은 임의의 공백과 일치합니다.
Clear	모든 표시를 제거합니다.

적용

메서드	설명
Apply(ADocument: TsgcPDFDocument)	로드된 문서를 마스킹합니다. 문서는 저장되고 마스킹된 다음, 마스킹된 바이트에서 다시 로드됩니다.
Apply(AInputFile, AOutputFile)	파일을 마스킹합니다.

메서드	설명
ApplyToStream(AInput, AOutput), ApplyToBytes(AData)	스트림이나 바이트 배열을 마스킹합니다.

출력은 항상 완전히 새로운 파일입니다(전체 저장). 증분 업데이트가 유지했을 원본 바이트는 절대 기록되지 않습니다. Apply 이후에는 TextMatches, RemovedGlyphs, RedactedImages, RemovedImages, RemovedPaths, RemovedAnnotations 통계를 확인할 수 있으며, `GetLog`는 작업 로그를 반환합니다.

옵션

TsgcPDFRedactionOptions의 필드	설명
FillColorR, FillColorG, FillColorB	영역 위에 칠하는 사각형의 색상(0에서 1까지)입니다. 기본값은 검은색입니다.
OverlayText, OverlayFontSize	모든 영역 가운데에 그려지는 텍스트입니다(Helvetica). 0으로 설정하면 영역에 맞춥니다.
RemoveAnnotationsInArea	사각형이 영역과 교차하는 주석을 제거합니다. 링크와 필드를 가진 폼 필드 위젯도 포함됩니다. 기본값은 True입니다.
RemoveMetadata	/Info 딕셔너리와 XMP 메타데이터를 제거합니다.
RemoveAttachments	임베드된 파일과 파일 첨부 주석을 제거합니다.
RemoveJavaScript	JavaScript, /OpenAction, 추가 액션, 그리고 JavaScript, Launch, SubmitForm, ImportData 액션을 제거합니다.

암호화된 문서

암호로 문서를 연 다음 `Apply(ADocument)`를 호출하십시오. 콘텐츠는 복호화된 상태로 마스킹되며, 문서는 암호화 설정을 유지하므로 다음 SaveToFile이나 SaveToStream에서 다시 암호화되어 기록됩니다. 마스킹 후에는 SaveIncremental이 예외를 발생시킵니다. 파일, 스트림, 바이트 오버로드는 암호를 받지 않으므로 암호화된 입력을 거부합니다.

```
static void RedactEncrypted(string AFileName, string APassword, string AOutput)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  using (TsgcPDFRedactor oRedactor = new TsgcPDFRedactor())
  {
    oPDF.SetPassword(APassword);
    oPDF.LoadFromFile(AFileName);
    oRedactor.AddText("Salary", false, true);
    oRedactor.Apply(oPDF);
    // full save: written encrypted again with the same settings
    oPDF.SaveToFile(AOutput);
  }
}
```

콘텐츠가 제거되는 방식

- **텍스트**: 상자가 영역과 교차하는 모든 글리프가 콘텐츠 스트림에서 제거됩니다. 나머지 글리프는 위치를 그대로 유지합니다.
- **이미지**: sgcPDF가 디코딩할 수 있는, 구성 요소당 1비트부터 8비트까지의 이미지는 마스킹된 픽셀을 채우기 색상으로 칠하여 다시 작성됩니다(RedactedImages). 지원되지 않는 필터를 사용하거나 너무 크거나 구성 요소당 16비트인 등 디코딩할 수 없는 이미지는 완전히 제거됩니다(RemovedImages).
- **벡터 그래픽**: 영역 안에 완전히 들어가는 하위 경로는 제거됩니다(RemovedPaths). 영역을 가로지르기만 하는 하위 경로는 채우기 사각형 아래에 그대로 남습니다. 클리핑 경로는 변경되지 않습니다.

- **폼 XObject**는 재귀적으로 마스킹됩니다. 너무 깊게 중첩되었거나 지원되지 않는 필터를 사용하는 폼 XObject는 전체가 제거됩니다.
- 콘텐츠 스트림이 지원되지 않는 필터를 사용하는 페이지는 마스킹되지 않은 채로 남는 대신 EsgcPDFRedactionError를 발생시킵니다.

예제에서 pypdf는 출력물에서 마스킹된 단어의 흔적을 전혀 찾지 못했습니다. 추출된 텍스트에서도, 어떤 콘텐츠 스트림에서도 찾지 못했으며, 파일은 단일 리비전으로 구성됩니다.

참고 항목

- 인증서 암호화
- 텍스트 추출

태그된 PDF

태그된 PDF(ISO 32000-1, 14.8)는 페이지 콘텐츠와 함께 논리적 구조 트리를 포함합니다. 제목, 단락, 목록, 표, 대체 텍스트가 있는 그림, 읽기 순서대로 나열된 링크가 여기에 해당합니다. 스크린 리더, 모바일 기기의 리플로우, 텍스트 내보내기가 이를 사용하며, PDF/UA와 PDF/A 레벨 A 규격 준수를 위해 필수입니다.

구조 만들기

TsgcPDFDocument.Tagged를 True로 설정하고, 각 페이지의 그리기 호출을 `Page.Tagging.BeginElement`와 `EndElement` 사이에 포함하십시오. 페이지 태깅은 마크된 콘텐츠 연산자(MCID가 있는 BDC/EMC)를 기록하여 요소와 연결하며, `StructureTree`는 요소, ParentTree, RoleMap을 수집하여 문서를 저장할 때 /MarkInfo, /Lang, 페이지 /StructParents를 기록합니다.

```
static void CreateTaggedPDF(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.Info.Title = "Quarterly report";
    oPDF.Tagged = true;
    oPDF.StructureTree.Lang = "en-US";
    int vFont = oPDF.AddTrueTypeFont("Arial", false, false);
    TsgcPDFPage oPage = oPDF.AddPage(595, 842);
    TsgcPDFPageTagging oTag = oPage.Tagging;
    // running header: not part of the logical content
    oTag.BeginArtifact(TsgcPDFArtifactKind.akPagination, TsgcPDFArtifactSubtype.asHeader);
    oPage.DrawTextTT("ACME Corp.", 50, 810, vFont, 8);
    oTag.EndArtifact();
    oTag.BeginElement(TsgcPDFStructType.stH1);
    oPage.DrawTextTT("Quarterly report", 50, 770, vFont, 20);
    oTag.EndElement();
    oTag.BeginElement(TsgcPDFStructType.stP);
    oPage.DrawTextTT("Sales grew in every region.", 50, 740, vFont, 11);
    oTag.EndElement();
    // figures need an alternate text
    TsgcPDFStructElement oFigure = oTag.BeginElement(TsgcPDFStructType.stFigure);
    oFigure.Alt = "Blue bar chart of the sales per region";
    oFigure.SetBBox(50, 600, 250, 700);
    oPage.DrawRectangleFill(50, 600, 200, 100, 0.2, 0.4, 0.8);
    oTag.EndElement();
    // grouping elements (no content) are opened on the tree
    oPDF.StructureTree.BeginElement(TsgcPDFStructType.stTable);
    oPDF.StructureTree.BeginElement(TsgcPDFStructType.stTR);
    TsgcPDFStructElement oCell = oTag.BeginElement(TsgcPDFStructType.stTH);
    oCell.Scope = TsgcPDFTableScope.tsColumn;
    oPage.DrawTextTT("Region", 50, 570, vFont, 11);
    oTag.EndElement();
    oCell = oTag.BeginElement(TsgcPDFStructType.stTH);
    oCell.Scope = TsgcPDFTableScope.tsColumn;
    oPage.DrawTextTT("Sales", 200, 570, vFont, 11);
    oTag.EndElement();
    oPDF.StructureTree.EndElement();
    oPDF.StructureTree.BeginElement(TsgcPDFStructType.stTR);
    oTag.BeginElement(TsgcPDFStructType.stTD);
    oPage.DrawTextTT("North", 50, 550, vFont, 11);
    oTag.EndElement();
    oTag.BeginElement(TsgcPDFStructType.stTD);
    oPage.DrawTextTT("1200", 200, 550, vFont, 11);
    oTag.EndElement();
    oPDF.StructureTree.EndElement();
    oPDF.StructureTree.EndElement();
    // a link: the annotation added inside the Link element is attached to it
    oTag.BeginElement(TsgcPDFStructType.stLink).Alt = "eSeGeCe web site";
```

```
    oPage.DrawTextTT("www.esegece.com", 50, 520, vFont, 11);
    oPDF.Annotations.AddLinkURI(0, 50, 515, 150, 532,
      "https://www.esegece.com");
    oTag.EndElement();
    // PDF/UA-1 and PDF/A-2a in the same file
    oPDF.SetPDFALevel(TsgcPDFALevel.paA2a);
    oPDF.PDFUA = true;
    oPDF.SaveToFile(AFileName);
  }
}
```

멤버	설명
Page.Tagging.BeginElement(AType; ATitle)	현재 요소의 자식으로, 이 페이지에 콘텐츠가 그려지는 요소를 엽니다. TsgcPDFStructElement를 반환합니다.
StructureTree.BeginElement(AType; ATitle)	페이지 콘텐츠가 없는 그룹 요소(Table, TR, L, Sect, Part...)를 엽니다.
EndElement	현재 요소를 닫습니다(Begin 호출에 맞추어 페이지 태깅이나 트리에서).
BeginCustomElement(ATypeName...)와 StructureTree.AddRole	/RoleMap을 통해 표준 유형에 매핑되는 사용자 지정 구조 유형입니다.
Page.Tagging.BeginArtifact(AKind; ASubtype), EndArtifact	문서의 일부가 아닌 콘텐츠입니다. 반복되는 머리글과 바닥글, 페이지 번호, 워터마크, 장식이 해당합니다. AKind: akPagination, akLayout, akPage, akBackground. ASubtype: asHeader, asFooter, asWatermark.
StructureTree.Lang	문서의 자연어(카탈로그 /Lang)입니다. 예: en-US.

TsgcPDFStructType에는 모든 표준 유형이 있습니다: stDocument, stPart, stArt, stSect, stDiv, stH, stH1부터 stH6까지, stP, stL, stLI, stLbl, stLBody, stTable, stTHead, stTBody, stTFoot, stTR, stTH, stTD, stFigure, stCaption, stLink, stSpan, stQuote, stNote, stReference, stCode, stBlockQuote, stFormula, stTOC, stTOCI, stAnnot.

요소 속성

속성	설명
Alt	대체 텍스트입니다. Figure와 Formula에 필수이며, Alt 없이 Figure를 저장하면 EsgcPDFTaggedError가 발생합니다.
ActualText, Expansion, Lang, Title, ID	대체 텍스트, 약어 확장, 요소의 언어, 제목, 식별자입니다.
SetBBox(ALeft, ABottom, ARight, ATop)	Figure, Formula 또는 Table의 경계 상자입니다.
Scope, RowSpan, ColSpan, Headers	표 셀: TH의 범위(tsRow, tsColumn, tsBoth), 스팬, 헤더 셀의 ID입니다.

링크와 주석

Link 요소가 열려 있는 동안 `Annotations.AddLinkURI`(또는 다른 Add 메서드)로 추가한 주석은 객체 참조로서 해당 Link 요소에 /StructParent와 함께 연결됩니다. `Page.Tagging.AddAnnotation`은 명시적으로 하나를 연결합니다.

표

태그된 문서에 그려진 TsgcPDFTable은 Table, THead, TBody, TR, TH, TD로 자동 태그됩니다. DrawMultiPage를 사용하면 다음 페이지에서 반복되는 헤더 행이 페이지 매김 아티팩트로 태그되어, 스크린 리더가 한 번만 읽습니다.

```
static void CreateTaggedTable(string AFileName)
{
  using (TsgcPDFDocument oPDF = new TsgcPDFDocument())
  {
    oPDF.Tagged = true;
    oPDF.StructureTree.Lang = "en-US";
    TsgcPDFTable oTable = new TsgcPDFTable(2);
    oTable.SetHeaders(new string[] { "Product", "Quantity" });
    for (int i = 1; i <= 60; i++)
      oTable.AddRowData(new string[] { "Item " + i, i.ToString() });
    // tagged as Table / TR / TH / TD; the header repeated on the next
    // pages is tagged as a pagination artifact
    oTable.DrawMultiPage(oPDF, 50, 790, 495, 595, 842);
    oPDF.SaveToFile(AFileName);
  }
}
```

로드된 태그 문서는 저장할 때 구조 트리를 그대로 유지합니다.

참고 항목

- PDF/UA와 PDF/A 레벨 A
- 접근성 검사기
- 표

PDF/UA와 PDF/A 레벨 A

PDF/UA-1

PDF/UA-1(ISO 14289-1)은 보편적으로 접근 가능한 PDF를 위한 표준으로, 여러 국가에서 공공 문서에 대해 접근성 법률로 요구합니다. **TsgcPDFDocument.PDFUA**를 True로 설정하면 문서가 XMP 메타데이터에 pdfuaid:part 1과 dc:title을, 뷰어 환경 설정에 /DisplayDocTitle을 포함하여 저장됩니다.
PDF/UA 요구 사항을 충족하지 못하면 저장 시 예외가 발생합니다.

- 문서가 태그되지 않은 경우(`Tagged`가 False)
- `StructureTree.Lang`이 비어 있는 경우
- `Info.Title`이 비어 있는 경우
- 텍스트가 임베드되지 않은 14가지 표준 폰트 중 하나로 그려진 경우: AddTrueTypeFont(또는 AddTrueTypeFontFromFile)와 DrawTextTT를 사용하십시오.
- 보조 기술에 필요한 ppExtract 권한 없이 문서가 암호화된 경우

PDF/A 레벨 A

PDF/A의 레벨 A(접근 가능)는 레벨 B에 태그된 PDF 요구 사항을 추가한 것입니다. SetPDFALevel에 **paA1a**, **paA2a** 또는 **paA3a**를 전달하고, PDF/UA와 마찬가지로 콘텐츠를 태그하십시오. PDF/UA와 PDF/A는 같은 파일에서 함께 사용할 수 있습니다.

```
oPDF.SetPDFALevel(TsgcPDFALevel.paA2a);
oPDF.PDFUA = true;
```

태그된 PDF에서 만든 파일은 바로 이 조합을 사용하며, PDF/UA-1과 PDF/A-2a 모두에 대해 veraPDF를 통과합니다.

TsgcPDFALevel	표준
paA1b, paA1a	PDF/A-1(ISO 19005-1), 레벨 B 또는 A
paA2b, paA2a	PDF/A-2(ISO 19005-2), 레벨 B 또는 A
paA3b, paA3a	PDF/A-3(ISO 19005-3), 레벨 B 또는 A. 임의의 첨부 파일을 허용합니다.

체크리스트

- Info.Title과 StructureTree.Lang을 설정하십시오.
- 모든 폰트(TrueType 또는 OpenType)를 임베드하십시오.
- 모든 콘텐츠를 태그하십시오. 모든 텍스트와 그래픽은 요소나 아티팩트에 속해야 합니다.
- 모든 Figure와 Formula에 대체 텍스트를 지정하십시오.
- 제목을 순서대로(H1, 그다음 H2...) 사용하고, 표에서는 Scope가 있는 TH 셀을, 링크에는 Link 요소를 사용하십시오.
- 저장한 파일을 검사기로 검증하십시오.

참고 항목

- 태그된 PDF
- 접근성 검사기
- PDF/A

접근성 검사기

TsgcPDFAValidator(유닛 sgcPDF_PDFA)는 저장된 파일을 PDF/A 레벨에 대해 검사합니다. 레벨 A에서는 구조도 검사하며, **PDFUA**를 True로 설정하면 PDF/UA-1 검사도 함께 수행합니다.

```
static void ValidateAccessible(string AFileName)
{
  TsgcPDFAValidator oValidator = new TsgcPDFAValidator(TsgcPDFALevel.paA2a);
  using (TsgcPDFParser oParser = new TsgcPDFParser())
  {
    oParser.LoadFromFile(AFileName);
    oParser.Parse();
    // PDF/UA-1 checks in addition to the PDF/A-2a ones
    oValidator.PDFUA = true;
    if (oValidator.Validate(oParser))
      Console.WriteLine("PDF/A-2a and PDF/UA-1: no errors");
    else
      for (int i = 0; i < oValidator.Errors.Count; i++)
        Console.WriteLine("Error: " + oValidator.Errors[i]);
    for (int i = 0; i < oValidator.Warnings.Count; i++)
      Console.WriteLine("Warning: " + oValidator.Warnings[i]);
  }
}
```

검사 대상

메서드	검사 항목
CheckStructure	구조 트리, /MarkInfo, /Marked, /Lang.
CheckContentTagged	모든 페이지의 모든 텍스트 및 그리기 연산자가 구조 요소나 아티팩트에 속하는지 검사합니다.
CheckStructureElements	Figure와 Formula의 대체 텍스트, 제목 순서, 표 구조와 헤더 셀을 검사합니다.
CheckFontUnicode	모든 폰트에 ToUnicode 맵이나 표준 인코딩이 있어 텍스트를 읽을 수 있는지 검사합니다.
CheckPDFUA	pdfuaid:part, dc:title, /DisplayDocTitle, 구조 안의 주석, 페이지 /Tabs /S.

Validate는 해당 레벨의 모든 검사(PDF/UA가 활성화된 경우 이것도 포함)를 수행하고, 오류가 없으면 True를 반환합니다. 개별 메서드도 단독으로 호출할 수 있습니다.

오류 및 경고

`Errors`는 준수를 위반하는 문제를 나열합니다. 각 메시지는 참조하는 표준 조항으로 시작하므로 다음과 같이 조회할 수 있습니다.

```
[PDF/A-2a 6.7.3.3] [PDF/UA-1 7.1] Page 1: 1 content operator(s) neither tagged nor marked as artifact (first: TJ)
[PDF/UA-1 5] XMP metadata without pdfuaid:part 1
```

`Warnings`는 표준의 오류는 아니지만 접근성을 해치는 문제를 나열합니다. 예를 들면 다음과 같습니다.

```
[PDF/UA-1 7.4.2] Heading level skipped: H1 followed by H3
```

이 검사기는 자체 출력물을 빠르게 확인하기 위한 내장 도구입니다. 다른 소스의 문서에 대한 veraPDF나 PAC 같은 완전한 규격 검사기를 대체하지는 않습니다.

참고 항목

- PDF/UA와 PDF/A 레벨 A
- 태그된 PDF
- PDF/A

FAQ

DLL을 배포해야 합니까?

아닙니다. sgcPDF는 실행 파일에 컴파일되어 포함됩니다. Windows에 포함된 라이브러리만 사용하며, 렌더링과 폰트 로딩에는 GDI를, 전자 서명에는 CryptoAPI를 사용합니다.
.NET에서 sgcPDF는 네이티브 DLL 없이 NuGet 패키지로 배포되는 매니지드 어셈블리입니다. .NET 설치를 참조하십시오.

어떤 플랫폼을 지원합니까?

Delphi 7부터 Delphi 13까지, C++Builder 2007부터 13까지에서 Windows 32비트 및 64비트를 지원합니다. 코어, 래스터 렌더러, FireMonkey 뷰어는 Linux 64비트, macOS, iOS, Android에서도 실행됩니다. 설치를 참조하십시오.
sgcPDF .NET은 .NET Standard 2.0, .NET Framework 4.7.2, .NET 6, .NET 8, .NET 9를 대상으로 하며 Windows, Linux, macOS에서 실행됩니다. Windows Forms 패키지에는 Windows가 필요합니다. .NET 설치를 참조하십시오.

페이지 좌표의 원점은 어디입니까?

페이지의 왼쪽 아래 모서리이며, 단위는 포인트(1/72인치)입니다. Y 값은 위로 갈수록 증가합니다. 아키텍처를 참조하십시오.

악센트 문자나 라틴 문자가 아닌 문자가 표시되지 않는 이유는 무엇입니까?

표준 폰트는 WinAnsi 문자 집합만 지원합니다. AddTrueTypeFont와 DrawTextTT를 사용하여 TrueType 폰트를 사용하십시오. 텍스트와 폰트를 참조하십시오.

중국어, 일본어, 한국어 텍스트는 어떻게 씁니까?

설치된 CJK TrueType 폰트를 추가하십시오. 예를 들어 sgcPDFCJKFontNameFromEnum(cjkMSGothic)이 반환하는 이름을 사용하고, DrawTextTT로 그립니다.

암호로 보호된 PDF는 어떻게 엽니까?

암호화를 참조하십시오. LoadFromFile을 호출하기 전에 SetPassword를 호출하십시오.

바코드가 표시되지 않습니다.

페이지가 아니라 페이지 콘텐츠 빌더(`Page.Content`)를 TsgcPDFBarcodeGenerator에 전달하십시오. 바코드를 참조하십시오.

sgcPDFExtractAllText가 빈 문자열을 반환합니다.

TsgcPDFParser는 로드한 후 파싱해야 합니다. LoadFromFile 다음에 Parse를 호출하십시오. TsgcPDFDocument.LoadFromFile은 이 작업을 대신 수행합니다.

서명은 유효한데 신뢰할 수 없다고 나오는 이유는 무엇입니까?

서명자 인증서가 Windows에서 신뢰하는 루트로 체인되지 않기 때문이며, 자체 서명된 테스트 인증서에서는 정상적인 현상입니다. 프로덕션 환경에서는 신뢰할 수 있는 인증 기관이 발급한 인증서를 사용하십시오.

옵티마이저가 파일 크기를 줄이지 못했습니다.

이미 압축되어 있고 중복 객체가 없는 파일은 더 이상 작아지지 않습니다. 추가로 줄이려면 DownsampleImages나 Remove 옵션을 사용하십시오. PDF 파일 최적화를 참조하십시오.

업로드된 파일을 열 때 사용하는 메모리를 제한할 수 있습니까?

가능합니다. sgcPDFMaxDecodedSize 값을 낮추십시오. 오류 처리 및 악의적인 입력을 참조하십시오.

sgcPDF는 Linux에서 실행됩니까?

실행됩니다. 코어 유닛은 Delphi Linux 64비트 컴파일러로 컴파일됩니다. 서버나 콘솔 애플리케이션에서 문서를 생성, 편집, 암호로 암호화하고, 텍스트를 추출하고, TsgcPDFRendererRaster로 페이지를 PNG로 렌더링할 수 있습니다. AddTrueTypeFontFromFile로 파일에서 폰트를 로드하십시오. Linux를 참조하십시오.
.NET에서는 .NET 6 이상에서 esegece.sgcPDF 패키지를 사용하십시오. .NET API 차이점을 참조하십시오.

C++Builder에서 sgcPDF를 사용할 수 있습니까?

가능합니다. C++Builder 2007부터 13까지의 패키지가 포함되어 있으며, 생성된 헤더 파일을 바로 사용할 수 있습니다. C++Builder를 참조하십시오.

인보이스(Factur-X, ZUGFeRD)를 임베드한 PDF/A-3 파일은 어떻게 만듭니까?

애플리케이션에서 인보이스 XML을 생성한 다음 Attachments.AddFacturX에 전달하십시오. 문서는 XML과 Factur-X 메타데이터가 포함된 PDF/A-3b로 저장됩니다. Factur-X 및 ZUGFeRD 인보이스를 참조하십시오.

sgcPDF는 PAdES 서명을 생성할 수 있습니까?

가능합니다. PAdES B-B는 SubFilter를 pssCAdESDetached로 설정하고, B-LT와 B-LTA는 AddDSSDocument로 검증 데이터를 추가한 다음 TimestampDocument로 문서 타임스탬프를 추가하십시오. sgcSign을 사용하면 TsgcPDFPAdESSigner가 어떤 sgcSign 키 공급자로도 모든 레벨을 한 번의 호출로 생성합니다. PAdES 서명을 참조하십시오.

검은색 사각형으로 가리면 아래의 텍스트가 숨겨집니까?

아닙니다. 텍스트는 파일에 그대로 남아 있습니다. 마스킹된 영역의 텍스트, 이미지 픽셀, 그래픽을 실제로 제거하는 TsgcPDFRedactor를 사용하십시오. TsgcPDFRedactor를 참조하십시오.

접근성을 갖춘 PDF는 어떻게 만듭니까?

Page.Tagging으로 콘텐츠를 태그하고, Info.Title과 StructureTree.Lang을 설정하고, 폰트를 임베드한 다음 PDFUA를 True로 설정하십시오. 태그된 PDF를 참조하십시오.

sgcPDF .NET에서 사용할 수 없는 기능은 무엇입니까?

FireMonkey 유닛(TsgcPDFViewerFMX, TsgcPDFBarcodeFMX와 TsgcPDFRendererFMX)과 디자인 타임 패키지입니다. 그 외에는 동일한 클래스 이름으로 모두 사용할 수 있습니다. .NET API 차이점을 참조하십시오.

소개

sgcReports는 sgcPDF 위에 구축된 Delphi용 헤드리스, 플랫폼 중립 보고서 엔진입니다. 서버 및 웹 보고서 생성을 최우선으로 목표로 합니다: Windows 서비스, Linux64 데몬 및 HTTP 엔드포인트이며, VCL 및 FMX 데스크톱과 모바일 애플리케이션에서도 변경 없이 실행됩니다.
엔진은 기본적으로 접근 가능한 PDF(PDF/UA-1)와 아카이브용 PDF(PDF/A-3) 출력을 생성합니다. 보고서는 동일한 준비된 페이지로부터 XLSX, CSV 또는 PNG로도 내보낼 수 있습니다.

설계상 헤드리스

코어(보고서 모델, 표현식, 데이터 소스, 글꼴, 레이아웃, 엔진 및 내보내기)는 VCL이나 FMX에 의존하지 않으며 외부 DLL도 없습니다. Windows, Linux64, macOS, iOS 및 Android에서 컴파일되고 실행됩니다. 미리보기 컴포넌트(`TsgcReportPreviewVCL`, `TsgcReportPreviewFMX`)와 브라우저 뷰어는 선택적 추가 패키지입니다.

기본적으로 접근 가능하고 아카이브 가능함

PDF 내보내기는 제목, 단락, 머리글 행이 있는 표, 대체 텍스트가 있는 그림 및 아티팩트 등 구조 트리를 자동으로 태그화하므로, `PDFUA` 속성은 기본적으로 `True`입니다. 아카이브용 PDF/A-3 출력은 `PDFA` 속성으로 제어되며 PDF/UA와 결합할 수 있습니다. 접근성과 PDF/A를 참조하십시오.

플랫폼

Delphi 7, 2007, 2009, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12 및 13. Windows, Linux64, macOS, iOS 및 Android(FireMonkey 사용)입니다. 코어는 제네릭, 익명 메서드, 인라인 변수 또는 레코드 메서드 없이 Delphi 7에서 컴파일됩니다.

에디션 및 요구 사항

패키지	요구 사항	필요한 이유
`sgcReports` (코어)	sgcPDF	보고서 모델, 표현식, 데이터 소스, 글꼴, 레이아웃, 엔진, PDF/XLSX/CSV/PNG 내보내기, 플루언트 API, JSON 정의입니다. sgcPDF는 모든 구성에서 필요합니다.
`sgcReportsVCL` / `sgcReportsFMX`	코어, VCL 또는 FMX	`TsgcReportPreviewVCL` 또는 `TsgcReportPreviewFMX` 입니다.
`sgcReportsHTML`	코어, sgcWebSockets(Professional 에디션 이상), sgcHTML	`TsgcReportHTTPEndpoint`, `TsgcReportHTMLViewer` 및 `TsgcReportHTMLDesigner` 입니다. sgcWebSockets는 이 패키지에만 필요하며, 코어에서는 전혀 사용되지 않습니다.

다음: 설치, 그다음 시작하기.

설치

sgcReports는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다. sgcReports에는 sgcPDF가 필요하며, 설치 프로그램이 이를 처리합니다.

요구 사항

- Delphi 7, 2007, 2009, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12 또는 13(또는 이에 대응하는 C++Builder 버전, C2007부터 C13까지).
- sgcPDF, 동일한 Delphi 버전입니다. sgcUI 설치 프로그램이 sgcReports와 함께 설치합니다.
- `sgcReportsHTML` 패키지가 설치된 경우에만 sgcWebSockets Professional 에디션 이상과 sgcHTML이 필요합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcReports (installs sgcPDF)**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcReports에 포함된 항목

- **sgcPDF**는 sgcReports가 기반으로 삼는 엔진입니다. sgcReports를 선택하면 sgcPDF도 함께 선택되며, sgcReports가 설치될 때마다 sgcPDF도 설치됩니다.
- **sgcReports HTML for sgcWebSockets**는 **sgcWebSockets bridges** 아래에 표시됩니다. Delphi 10 Seattle 이후 버전용 `sgcReportsHTML` 패키지(HTTP 엔드포인트, 브라우저 뷰어, 웹 디자이너)를 설치합니다. 설치 프로그램은 sgcWebSockets가 이미 컴퓨터에 설치되어 있고 라이선스에 sgcReports가 포함된 경우에만 이 항목을 제공합니다. 그렇지 않으면 회색으로 표시됩니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드이며, sgcReports와 sgcPDF도 포함됩니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Reports`	sgcReports 데모입니다.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcPDF는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcReports와 sgcPDF도 포함됩니다. 체험판 빌드는 시작 알림(`sgcReports_License.pas`)을 표시하며, 라이선스 빌드는 표시하지 않습니다.
- **Community**: sgcPDF에는 Community 에디션이 없으므로, 무료 sgcUI Community 설치 프로그램은 sgcPDF를 설치하지 않습니다. 컴퓨터에 라이선스가 있는 sgcPDF가 이미 설치되어 있을 때만 sgcReports를 제공하며, sgcPDF를 포함한 Registered sgcUI 설치본이든 독립 실행형 sgcPDF 설치본이든 상관없습니다. 이때 그 sgcPDF를 기준으로 sgcReports를 컴파일합니다. 라이선스가 있는 sgcPDF가 없으면 sgcReports는 회색으로 표시되고, 설치 프로그램은 먼저 sgcPDF를 설치하라고 안내합니다. sgcWebSockets 브리지는 Community 설치 프로그램에 포함되지 않습니다.

독립 실행형 sgcPDF와 sgcReports 설치에서 마이그레이션하기

sgcPDF 또는 sgcReports가 이전에 자체 독립 실행형 설치 프로그램으로 설치된 경우, sgcUI 설치 프로그램은 그 설치를 감지하고 그 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다. 이 페이지는 이전 설치가 발견된 경우에만 표시됩니다. 마이그레이션 후에는 두 제품 모두 하나의 `sgcUI\Source` 폴더에서 컴파일됩니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcReportsD13.dpk`입니다.

패키지	내용
`sgcPDF<V>` , `sgcPDFVCL<V>` , `sgcPDFFMX<V>` / `dclsgcPDF<V>` , `dclsgcPDFFMX<V>`	sgcPDF, sgcReports가 필요로 하는 엔진(런타임/디자인 타임).
`sgcReports<V>`	코어 엔진과 내보내기(런타임).
`sgcReportsVCL<V>` / `dclsgcReports<V>`	VCL 미리 보기(런타임/디자인 타임).
`sgcReportsFMX<V>` / `dclsgcReportsFMX<V>`	FMX 미리 보기(런타임/디자인 타임).
`sgcReportsHTML<V>` / `dclsgcReportsHTML<V>`	HTTP 엔드포인트, 브라우저 뷰어, 웹 디자이너(런타임/디자인 타임).

의존 순서대로 빌드합니다. 먼저 sgcPDF 런타임 패키지, 그다음 sgcReports 코어, 그다음 VCL 또는 FMX, 그다음 디자인 타임 패키지, 마지막으로 `sgcReportsHTML`입니다. 그런 다음 `dcl*` 디자인 타임 패키지를 설치합니다. 디자인 타임 패키지를 설치하면 `TsgcReport`가 **sgcReports** 팔레트 페이지에 나타납니다.

검색 경로

사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가하거나, 프로젝트의 유닛 검색 경로에 추가합니다. sgcReports와 sgcPDF는 이 폴더를 공유하므로, 항목 하나로 둘 다 해결됩니다.

C++Builder

사용 중인 C++Builder 버전에 맞는 `.cbproj` 프로젝트를 `sgcUI\Packages` 아래에서 열고(예: `sgcReportsC13.cbproj`) 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

시작하기

보고서를 작성하는 방법은 세 가지입니다: 모델 API(페이지, 밴드 및 항목으로 TsgcReport 컴포넌트를 채움), 플루언트 API(TsgcReportBuilder에서 호출을 체이닝함), 또는 런타임에 불러오는 JSON 보고서 정의입니다. 세 가지 모두 동일한 TsgcReportEngine과 동일한 내보내기로 귀결됩니다.

1. 모델 API

`TsgcReport`, `TsgcReportEngine` 및 `TsgcReportPDFExport`를 폼에 놓거나(또는 코드에서 생성) 페이지 하나, 밴드 하나, 텍스트 항목 하나로 보고서를 채우십시오:

2. 플루언트 API

`TsgcReportBuilder`를 사용한 동일한 보고서이며, 폼에 컴포넌트가 필요하지 않습니다:
모든 스타일, 페이지 및 밴드 호출은 범위가 지정된 빌더(`TsgcReportStyleBuilder`, `TsgcReportPageBuilder`, `TsgcReportBandBuilder`)를 반환합니다. `EndStyle`, `EndBand` 및 `EndPage`는 감싸는 범위로 돌아가므로 체인을 계속할 수 있습니다.
전체 체인(스타일, 페이지, 밴드, 항목, 데이터, 내보내기 헬퍼)은 TsgcReportBuilder를 참조하십시오.

3. JSON 보고서 정의

어느 방식으로 작성된 보고서든 JSON으로 저장한 다음 재컴파일 없이 다시 불러올 수 있습니다:
엔벨로프 형식, 명명 규칙 및 생성된 스키마는 JSON 보고서 정의를 참조하십시오.

다음 단계

- 보고서 모델: 페이지, 밴드, 항목, 스타일 및 단위입니다.
- 데이터 소스: `TDataSet`, JSON 또는 이벤트로부터 레코드를 읽습니다.
- 사용법: 마스터-디테일 데이터를 사용한 인보이스.

보고서 모델

보고서(TsgcReport)는 페이지, 밴드, 항목 및 스타일의 트리이며, `TCollection` 위에 구축되어 다른 컴포넌트와 마찬가지로 DFM으로 스트리밍됩니다. 동일한 트리는 JSON으로 불러오거나 저장할 수도 있습니다. JSON 보고서 정의를 참조하십시오.

Units

`TsgcReport.Units`(`TsgcReportUnits`: `ruMillimeters`, `ruPoints`, `ruInches`)는 모든 `Left`, `Top`, `Width`, `Height` 및 여백을 입력하는 단위입니다. 내부적으로 레이아웃과 PDF 내보내기는 항상 포인트(1/72인치) 단위로 동작하며, 보고서는 이 속성을 읽거나 쓸 때 변환을 수행합니다.

Pages

`TsgcReportPage`(`TsgcReport.Pages` 내)는 `PaperSize`(`rpsA3`, `rpsA4`, `rpsA5`, `rpsLetter`, `rpsLegal`, `rpsCustom`), `Orientation`, `Width` 및 `Height`(`PaperSize`가 `rpsCustom`일 때만 사용됨), `Margins`, `Columns` 및 `ColumnSpacing`을 갖습니다. 보고서는 둘 이상의 페이지 정의를 가질 수 있습니다. `rbtDetailData` 하위 보고서는 이름(`TsgcReportItem.SubreportPageName`)으로 페이지를 참조합니다.

밴드

`TsgcReportBand`(`TsgcReportPage.Bands` 내)는 언제 얼마나 자주 인쇄될지 결정하는 `BandType`을 갖습니다:

BandType	인쇄 시점
rbtReportTitle	첫 페이지 머리글 앞에 한 번 인쇄됩니다.
rbtReportSummary	마지막 그룹과 디테일 데이터 이후에 한 번 인쇄됩니다.
rbtPageHeader	모든 페이지 상단에 인쇄됩니다.
rbtPageFooter	모든 페이지 하단에 인쇄됩니다.
rbtColumnHeader	다단 페이지에서 모든 열 상단에 인쇄됩니다.
rbtColumnFooter	모든 열 하단에 인쇄됩니다.
rbtGroupHeader	`GroupExpression` 의 서로 다른 값마다 한 번씩 인쇄되며, 그룹이 페이지 나누기에 걸치면 `RepeatOnEveryPage` 로 선택적으로 다시 인쇄할 수 있습니다.
rbtGroupFooter	집계가 완료된 후 각 그룹의 끝에서 한 번 인쇄됩니다.
rbtMasterData	`DataSourceName` 의 레코드를 순회하며, 그 자체는 아무것도 인쇄하지 않습니다.
rbtDetailData	현재 마스터 레코드 아래에서 `DataSourceName` 의 레코드를 순회합니다.
rbtChild	`ChildBandName` 을 통해 부모 밴드 바로 다음에 인쇄되며, 동일한 레코드를 공유합니다.

그 외 밴드 속성: `Height`, `CanGrow` 및 `CanShrink`(가장 큰 항목에 맞춰 늘어나거나 줄어듦), `KeepTogether`(페이지 나누기로 밴드를 분할하지 않음), `StartNewPage`, `PrintOnFirstPage` / `PrintOnLastPage`, `VisibleExpression` 및 `StyleName`(항목의 기본 스타일)입니다.

항목과 종류

`TsgcReportItem`(`TsgcReportBand.Items` 내)은 `Kind`(`TsgcReportItemKind`)를 갖습니다:

Kind	하위 옵션	참고
rikText	`Text` (`TsgcReportTextOptions`)	`Text.Text` 는 텍스트 템플릿입니다. 표현식과 텍스트 템플릿을 참조하십시오.
rikImage	`Image` (`TsgcReportImageOptions`)	`PictureData` 로 스트리밍되는 임베드된 그림 바이트입니다.
rikLine	—	수평 또는 수직 선입니다.
rikShape	`Shape` (`TsgcReportShapeOptions`)	`ShapeKind` : 사각형, 둥근 사각형, 타원, 선입니다.
rikBarcode	`Barcode` (`TsgcReportBarcodeOptions`)	11가지 심볼로지가 있습니다. `TsgcReportBarcodeSymbology` 를 참조하십시오. `Barcode.Value` 는 텍스트 템플릿입니다.
rikSubreport	`SubreportPageName`	그 자리에서 같은 보고서의 다른 페이지를 실행합니다.
rikPageBreak	—	새 페이지를 강제로 시작합니다.

공통 항목 속성: `Left`, `Top`, `Width`, `Height`, `StyleName`, `Visible` / `VisibleExpression`, `CanGrow` / `CanShrink`, `AltText`(접근성) 및 `TagRole`(접근성 참조).

스타일

`TsgcReportStyle`(`TsgcReport.Styles` 내)은 밴드나 항목이 `StyleName`으로 참조하는 `Name` 아래에 `Font`, `BackColor`, `Frame`, `Padding`, `HAlign` 및 `VAlign`을 그룹화합니다. 색상은 (`Graphics.TColor`가 아니라) 플랫폼 중립적인 `$00RRGGBB` cardinal인 `TsgcReportColor`입니다. `sgcReportColorNone`, `sgcReportColorBlack` 및 `sgcReportColorWhite`가 미리 정의되어 있습니다.

표현식과 텍스트 템플릿

sgcReports는 모든 표현식을 한 번만 컴파일한 다음, 레코드마다 한 번씩 평가합니다. 값은 Variant입니다.

필드, 변수 및 범위

필드는 `Source.Field`로 표기되며, 여기서 `Source`는 엔진에 추가된 데이터 소스의 `Name`(`TsgcReportEngine.AddDataSource`)이고 `Field`는 그 필드 이름 중 하나입니다. 단순 식별자는 엔진 변수(`TsgcReportEngine.Variables`, `Name=Value` 쌍)로 해석되며, `OnGetVariable`에서도 접근할 수 있습니다. 항상 사용할 수 있는 두 식별자는 `Page`(현재 페이지 번호)와 `TotalPages`이며, 후자는 마지막 페이지 이후에만 해결됩니다(지연 평가, 2단계 처리, "Page X of Y").
그룹 바닥글이나 페이지 바닥글은 다음 레코드가 아니라, 자신이 속한 그룹 또는 페이지의 **마지막** 레코드의 필드 값을 봅니다.

연산자

산술 연산자 `+ - * /`, 비교 연산자 `= <> < <= > >=`, 논리 연산자 `and or not`, `+`를 이용한 문자열 연결, 그리고 그룹화를 위한 괄호입니다.

함수

함수	설명
IIF(cond, a, b)	cond가 참이면 a, 아니면 b입니다.
IsNull(v)	v가 Null이거나 비어 있으면 True입니다.
Coalesce(v1, v2, ...)	Null이 아닌 첫 번째 인수입니다.
Upper(s)	대문자로 변환합니다.
Lower(s)	소문자로 변환합니다.
Trim(s)	앞뒤 공백을 제거합니다.
Length(s)	문자열 길이입니다.
Copy(s, from, count)	부분 문자열입니다.
Pos(sub, s)	1부터 시작하는 위치이며, 찾지 못하면 0입니다.
Replace(s, from, to)	모든 발생을 치환합니다.
FormatFloat(fmt, v)	숫자를 형식화합니다.
FormatDateTime(fmt, v)	날짜 또는 시간을 형식화합니다.
Str(v)	문자열로 변환합니다.
Round(v)	가장 가까운 정수로 반올림합니다.
Trunc(v)	0 방향으로 잘라냅니다.
Abs(v)	절대값입니다.
Int(v)	정수 부분입니다.

MinOf(a, b)	둘 중 더 작은 값입니다.
MaxOf(a, b)	둘 중 더 큰 값입니다.
Now()	현재 날짜와 시간입니다.
Date()	현재 날짜입니다.
Year(v)	날짜의 연도입니다.
Month(v)	날짜의 월입니다.
Day(v)	날짜의 일입니다.

집계 함수

`Sum`, `Count`, `Avg`, `Min` 및 `Max`는 인수를 하나 받습니다. 예를 들어 `Sum(Lines.Qty * Lines.Price)`입니다. 집계 함수는 인쇄되는 범위 전체에 걸쳐 누적됩니다: 그룹 바닥글 안에서는 그룹마다 재설정되고, 페이지 바닥글 안에서는 페이지마다 재설정되며, 보고서 요약 안에서는 보고서 전체에 대해 실행됩니다.

Null 처리 규칙

누락된 필드나 설정되지 않은 변수는 Null로 평가되며, 이는 산술 연산을 통해 전파되고(`Null + 1`은 Null입니다), Delphi Variant 규칙과 마찬가지로 Null이 아닌 값과 비교할 때 같지도 다르지도 않은 것으로 처리됩니다. 형식화하기 전에 기본값으로 대체하려면 `IsNull` 또는 `Coalesce`를 사용하십시오.

텍스트 템플릿

`TsgcReportTextTemplate`(`TsgcReportItem.Text.Text`와 `TsgcReportItem.Barcode.Value`가 사용함)은 대괄호 안에 표현식이 포함된 리터럴 텍스트입니다. 예를 들어 `'Invoice [Invoices.Number], total [Sum(Lines.Qty * Lines.Price)]'`입니다. `[ ]` 밖의 모든 내용은 그대로 복사되므로, 대괄호가 없는 `STOCK-2026-09`와 같은 바코드 값은 앞자리 0을 포함하여 문자 그대로 인쇄됩니다. `TsgcReportTextOptions.DisplayFormat`(해당하는 경우 `Barcode.DisplayFormat`)은 `FormatFloat` 나 `FormatDateTime`과 같은 방식으로 단일 표현식 템플릿의 평가 결과를 형식화합니다.

데이터 소스

데이터 소스는 TsgcReportEngine에 `AddDataSource`로 추가되는 `TsgcReportDataSource` 하위 클래스입니다. 밴드 표현식이 `Source.Field`로 참조하는 것은 컴포넌트의 `Name`이 아니라 해당 `SourceName` 속성이며, `rbtMasterData` 또는 `rbtDetailData` 밴드는 `DataSourceName`을 통해 동일한 이름으로 이를 참조합니다.

종류

클래스	읽어오는 대상
TsgcReportDataSetSource	임의의 `TDataSet` (VCL `TClientDataSet`, FireDAC, dbExpress 등).
TsgcReportJSONSource	문자열, 파일 또는 스트림으로부터의 객체 JSON 배열입니다.
TsgcReportEventSource	`OnFirst`, `OnNext`, `OnEof` 및 `OnGetValue` 이벤트 핸들러이며, 페이지 단위 REST 호출과 같이 데이터셋이나 JSON 형태가 없는 데이터를 위한 것입니다.

마스터-디테일

`TsgcReportDataSource`(공통 기반 클래스)에는 `MasterSource`, `MasterFields` 및 `DetailFields`(세미콜론으로 구분되며 양쪽 개수가 같음)가 있습니다. `DataSourceName`이 디테일 소스를 가리키는 디테일 밴드가 마스터 밴드 아래에 중첩되면, 엔진은 각 디테일 반복 전에 디테일 소스를 다시 시작하고 이 필드들을 기준으로 현재 마스터 레코드와 매칭시킵니다. 이것이 인보이스 데모에서 인보이스 라인을 해당 인보이스에 연결하는 방식입니다. 사용법: 마스터-디테일 데이터를 사용한 인보이스를 참조하십시오.

JSON 레코드

`TsgcReportJSONSource.RecordsPath`는 레코드 배열까지의 점(dot) 표기 경로입니다. JSON 루트 자체가 배열인 경우 비워 둡니다. 필드 이름은 첫 번째 레코드의 멤버로부터 발견됩니다.

스레드 안전성

데이터 소스는 동시에 준비되는 보고서들 사이에서 공유되지 않습니다. 각 스레드는 자신만의 소스 인스턴스(및 자신만의 엔진)를 사용해야 합니다. 스레드 안전성과 서버 사용을 참조하십시오.

그룹, 머리글, 바닥글 및 열

그룹

`GroupExpression`(예: `'Sales.Region'`)이 있는 `rbtGroupHeader` 밴드는 해당 표현식의 값이 바뀔 때마다 새 그룹을 시작하며, 짝을 이루는 `rbtGroupFooter` 밴드는 집계가 확정된 후 그룹을 닫습니다. 그룹이 페이지 나누기에 걸치는 경우 머리글을 다시 인쇄하려면 `RepeatOnEveryPage`를 설정하고, 모든 그룹을 각자의 페이지에 강제로 배치하려면(인보이스당 한 페이지에 사용) `StartNewPage`를 설정하십시오. 그룹은 중첩될 수 있습니다: 한 페이지에 둘 이상의 `rbtGroupHeader` / `rbtGroupFooter` 쌍이 있을 수 있으며, 밴드가 나타나는 순서대로 평가됩니다.

페이지 및 열 머리글과 바닥글

`rbtPageHeader` 와 `rbtPageFooter` 는 모든 페이지에 인쇄됩니다. `rbtColumnHeader` 와 `rbtColumnFooter` 는 `TsgcReportPage.Columns`가 1보다 클 때 모든 열에 인쇄됩니다. `PrintOnFirstPage`와 `PrintOnLastPage`(기본값 `True`)는 첫 페이지나 마지막 페이지에서 페이지 머리글 또는 바닥글을 표시하지 않도록 합니다.

함께 유지

`TsgcReportBand.KeepTogether := True`는 밴드 전체를 한 페이지에 강제로 배치합니다: 현재 페이지에 남은 공간에 맞지 않으면, 엔진은 밴드를 분할하는 대신 인쇄하기 전에 새 페이지를 시작합니다. 합계 블록 등 페이지 나누기로 잘려서는 안 되는 밴드에 사용됩니다.

열

`TsgcReportPage.Columns`(기본값 1)와 `ColumnSpacing`은 마스터 및 디테일 밴드를 신문 스타일의 열로 배치합니다. 엔진은 한 열을 채운 다음 다음 열로 넘어가며, 각 열 경계에서 `rbtColumnHeader` / `rbtColumnFooter`를 인쇄합니다.

하위 보고서

`rikSubreport` 항목(`TsgcReportItem.SubreportPageName`)은 항목의 위치에서 같은 보고서의 다른 `TsgcReportPage`를 그 자리에서 실행합니다. 하위 보고서 페이지는 자체 밴드와 자체 데이터 소스 참조를 가질 수 있으며, 동일한 실행의 일부로 준비됩니다.

Page X of Y

`Page`와 `TotalPages`는 페이지 번호 매기기 텍스트를 위해 예약된 두 식별자이며, 전형적으로 `rbtPageFooter` 밴드에서 `'Page [Page] of [TotalPages]'`로 사용됩니다. `TotalPages`는 지연 평가됩니다: 엔진은 페이지 수를 세기 위해 전체 보고서를 한 번 실행한 다음, 페이지가 내보내기로 전달되기 전에 2차 처리에서 모든 지연된 표현식을 해결합니다.

글꼴

TsgcReportFontManager(유닛 sgcReports_Fonts)는 레이아웃 엔진(줄바꿈, CanGrow)과 PDF 내보내기 모두를 위해 글꼴을 검색, 측정 및 대체합니다. 전역 상태를 가지지 않으며, 모든 엔진과 내보내기는 자신의 FontManager 속성을 통해 관리자를 참조합니다.

기본 관리자

FontManager가 지정되지 않으면 엔진과 내보내기는 프로세스 전체에서 공유되는 sgcReportDefaultFontManager를 사용합니다. 이는 시스템에 설치된 글꼴로부터 처음 사용될 때 한 번 생성되는 공유 인스턴스입니다. 기본 관리자에 없는 글꼴이 보고서에 필요하거나, 동시에 실행되는 보고서 간에 글꼴 로딩을 분리하고 싶을 때는 자체 TsgcReportFontManager를 지정하십시오.

검색

관리자는 플랫폼의 글꼴 폴더를 검색합니다: Windows 글꼴 폴더, 표준 Linux 글꼴 디렉터리, macOS 및 iOS 시스템 글꼴 위치, Android 글꼴 및 자산 경로입니다. 검색된 TrueType 서체는 패밀리 이름, 두께 및 스타일(굵게, 기울임)로 색인화됩니다.

표준 글꼴과 대체

14개의 표준 PDF 글꼴(Helvetica, Times, Courier와 그 굵게 및 기울임 변형, 그리고 Symbol과 ZapfDingbats)이 기본적으로 매핑되어 있습니다. Font.Name := 'Helvetica'로 지정된 스타일은 내보내기에 임베드 가능한 서체가 필요하지 않는 한(PDF/A이거나 기본 글꼴이 표현할 수 없는 문자인 경우) 임베드 없이 렌더링됩니다. 관리자는 요청된 패밀리에 대해 TrueType 서체를 대체하며, 현재 서체에 해당 문자에 대한 글리프가 없으면 글리프 단위로 다른 설치된 서체로 추가 대체합니다. 이 덕분에 라틴 기본 글꼴로도 데이터에 포함된 CJK나 키릴 문자 텍스트를 렌더링할 수 있습니다.

임베딩

PDF 내보내기는 실제로 사용된 TrueType 서체를 문서에서 참조하는 글리프로 서브셋하여 임베드합니다. 단, PDF/A가 이미 부분적으로 임베드된 서체의 완전한 임베딩을 요구하는 경우는 예외입니다. TsgcReportPDFExport.FontManager는 명시적인 서체가 없는 항목의 서체를 결정하며, PDF/A 출력의 경우 TsgcReportFontManager.ResolveEmbeddable을 통해 표준 14개 글꼴의 TrueType 대체 서체를 결정합니다.

모바일 및 헤드리스 플랫폼

iOS는 샌드박스 앱에서 읽을 수 있는 시스템 글꼴 폴더가 없으며, Linux64 데몬은 설치된 글꼴이 거의 없는 최소 이미지에서 실행될 수 있습니다. 두 경우 모두 실행 파일 옆이나 앱 번들에 TrueType 글꼴(예: 저작권 제한이 없는 DejaVu Sans)을 함께 배포하여 글꼴 검색이 이를 대체로 사용하도록 하십시오. 사용법: 모바일 FMX 보고서를 참조하십시오.

스레드 안전성과 서버 사용

sgcReports는 서버 및 웹 보고서 생성을 위해 설계되었으며, 여러 보고서가 서로 다른 스레드에서 동시에 준비되고 내보내집니다.

규칙

엔진, 그 보고서, 데이터 소스 및 내보내기는 스레드 간에 공유되지 않습니다. 각 워커 스레드는 자신만의 `TsgcReportEngine`, `TsgcReport`(또는 읽기 전용 보고서 정의를 공유하고 엔진만 복제, 엔진이 실행 상태를 갖고 있으므로), 자신만의 데이터 소스 인스턴스, 자신만의 내보내기 컴포넌트를 생성하고 사용해야 합니다. 모델, 레이아웃 또는 내보내기 클래스에는 보고서 자체의 상태를 둘러싼 전역 잠금이 없으므로, 두 스레드가 하나의 엔진이나 하나의 내보내기를 공유하면 두 실행 모두 손상됩니다.

공유할 수 있는 것

`TsgcReportFontManager`는 폰트가 검색된 후에는 프로세스 전역 `sgcReportDefaultFontManager`를 포함하여 스레드 간에 읽기 전용으로 공유될 수 있습니다. 폰트 검색과 측정은 다른 스레드가 의존하는 호출별 상태를 변경하지 않기 때문입니다. `TsgcReportFontManager.ResolveEmbeddable` 역시 관리자의 `EmbedStandardFonts` 설정을 변경하지 않으므로, 스레드 간에 공유되는 관리자에서 호출해도 마찬가지로 안전합니다. `TsgcReportDefinition`(또는 단순 함수 `sgcReportLoadFromJSON` / `sgcReportSaveToJSON`)은 각 호출이 자체적인 `TsgcReport` 인스턴스를 사용하는 한 어떤 스레드에서도 호출할 수 있습니다.

패턴: 워커 풀

시작 시 공유된 읽기 전용 JSON 보고서 정의를 로드합니다. 이후 각 워커 스레드는 작업마다 다음을 수행합니다: `TsgcReport`를 생성하고, `sgcReportLoadFromJSON`을 호출하여 이를 채우고, 해당 작업을 위한 `TsgcReportEngine`과 데이터 소스를 생성하여 추가하고, `Prepare`를 호출하고, 내보낸 다음, 해당 스레드가 생성한 모든 것을 해제합니다. 이것이 Linux 데몬 데모가 사용하는 설계이며, Win32, Win64 및 Linux64에서 8스레드 스트레스 테스트로 검증되었습니다.

Windows에서의 서버 애플리케이션

`sgcReportsConfigureForServer`(유닛 `sgcReports_Types`)는 많은 스레드에서 동시에 보고서를 준비하고 내보내는 서버를 위한 선택적(옵트인) 설정입니다. Windows에서 Delphi 2006 이상인 경우, 이 함수는 `System.NeverSleepOnMMThreadContention`을 `True`로 설정하여, 두 스레드가 동일한 블록 크기를 두고 경합할 때 RTL 메모리 관리자가 잠들지 않고 스핀하도록 합니다. 다른 플랫폼과 이전 컴파일러에서는 아무 동작도 하지 않습니다. sgcReports는 이 함수를 스스로 호출하지 않으므로, 애플리케이션이 직접 호출하기 전까지는 아무 효과가 없으며, 호출하면 이 설정은 프로세스 전체에 적용되어 보고서 내보내기뿐 아니라 애플리케이션이 수행하는 모든 메모리 할당에 영향을 줍니다.
멀티스레드 서버에서는 워커 스레드가 생성되기 전, 애플리케이션 시작 시 한 번만 호출하십시오. Win64에서 8개의 스레드가 각각 25개의 보고서를 내보내도록 측정한 결과, 두 번의 실행에서 처리량이 단일 스레드 대비 약 x1.8배에서 약 x5배로 증가했습니다(13.6초에서 4.4초로, 그리고 7.8초에서 2.8초로). 이 설정은 스레드가 대기하는 동안 CPU를 더 많이 사용하게도 하므로, 단일 스레드 애플리케이션에서는 비활성 상태로 두십시오.

HTTP 엔드포인트

TsgcReportHTTPEndpoint는 보고서 이름당 하나의 장기 실행 엔진을 등록하고 이에 대한 동시 요청을 처리합니다. 공유된 엔진과 준비된 페이지 캐시에 대한 접근을 어떻게 동기화하는지는 해당 컴포넌트의 주제를 참조하십시오.

접근성(PDF/UA)

TsgcReportPDFExport.PDFUA는 기본적으로 `True`입니다. 명시적으로 끄지 않는 한 내보낸 모든 PDF는 PDF/UA-1(ISO 14289-1)로 태그가 지정됩니다. 태깅은 구조 트리(제목, 단락, 머리글 행이 있는 표, 대체 텍스트가 있는 그림, 아티팩트), 문서 `/Lang`, 제목 및 `pdfuaid` 식별자를 기록하며, 사용된 모든 글꼴을 임베드합니다(태그가 지정된 PDF는 임베드되지 않은 기본 글꼴에 의존할 수 없으므로, 표준 글꼴은 글꼴 관리자를 통해 TrueType 대체 글꼴로 교체됩니다).

자동 태깅 규칙

각 항목의 구조 트리 내 역할은 재정의되지 않는 한 밴드와 종류에서 파생됩니다.

소스	기본 역할
`rbtReportTitle` 밴드 내의 텍스트 항목	제목(레벨은 설정된 경우 `TagRole` 에 따르며, 그렇지 않으면 최상위 제목).
`rbtGroupHeader` 밴드 내의 텍스트 항목	제목입니다.
그 외 텍스트 항목	단락입니다.
열 머리글 밴드 아래에서 반복되는 상세 행으로 구성된 표	표이며, 열 머리글 밴드의 셀은 표 머리글 셀로 표시됩니다.
이미지 및 바코드 항목	그림이며, `AltText` 를 대체 텍스트로 사용합니다.
선 및 장식용 도형	아티팩트(보조 기술에 제공되는 읽기 순서에서 제외됨).

TagRole

`TsgcReportItem.TagRole`을 설정하여 자동 역할을 재정의합니다: `rtrAuto`(기본값), `rtrHeading1`, `rtrHeading2`, `rtrHeading3`, `rtrParagraph`, `rtrFigure`, `rtrArtifact`, `rtrTableHeader` 또는 `rtrTableCell`입니다. `rtrFigure`로 태그가 지정된 항목에는 항상 `AltText`를 설정하십시오. 대체 텍스트는 그림이 PDF/UA를 통과하기 위해 요구되는 항목이기 때문입니다.

읽기 순서

구조 트리는 항목이 인쇄되는 순서, 즉 위에서 아래로, 밴드 내에서는 왼쪽에서 오른쪽으로의 순서를 따르며, 이는 일반적인 보고서 레이아웃에서 이미 의도된 읽기 순서와 일치하므로 별도의 수동 순서 지정 단계가 없습니다.

veraPDF를 이용한 검증

PDF/UA-1 파일을 ISO 프로필에 대해 검사하는 명령은 사용법: veraPDF로 검증된 접근성 PDF를 참조하십시오.

PDF/A

TsgcReportPDFExport.PDFA(`TsgcReportPDFAMode`)는 아카이브 출력을 선택합니다: `rpaNone`(기본값, 일반 PDF), `rpaA3b`(PDF/A-3b, 시각적 재현성) 또는 `rpaA3a`(PDF/A-3a, A-3b에 태깅 및 언어 요구 사항을 추가함)입니다. PDF/A-3는 문서에 임의의 파일을 임베드할 수 있게 하며, 이 때문에 전자 인보이스 등 기계가 읽을 수 있는 원본(예: 인보이스의 XML)이 동일한 PDF 안에 함께 전달되는 워크플로에서 아카이브용으로 선택됩니다.

PDF/UA와 결합하기

`PDFA = rpaA3a`와 `PDFUA = True`는 함께 설정할 수 있습니다: 둘 다 태그가 지정된 구조 트리, 선언된 언어, 임베드된 글꼴을 요구하므로, 내보내기는 두 적합성 요구를 모두 충족하는 하나의 구조 트리를 생성합니다. 이는 접근 가능하면서도 아카이브 가능해야 하는 인보이스 등 문서에 사용되는 조합입니다. 인보이스 사용법과 veraPDF 사용법을 참조하십시오.

글꼴

PDF/A는 모든 글꼴이 임베드되어야 합니다. 스타일이 이름으로 표준 14 글꼴을 사용하는 경우, 내보내기는 PDF/A가 유일한 텍스트 표현 수단으로 금지하는 기본 글꼴에 의존하는 대신 글꼴 관리자를 통해 TrueType 서체를 대체하고 임베드합니다. 글꼴을 참조하십시오.

재현 가능한 출력

`TsgcReportPDFExport`의 `FixedCreationDate`와 `FixedDocumentID`는 생성 및 수정 날짜와 파일 식별자를 현재 시간 대신 호출자가 지정한 값으로 고정하므로, 동일한 보고서를 반복해서 내보내도 바이트 단위로 동일한 파일이 생성됩니다. 골든 파일 테스트와 아카이브 중복 제거에 유용합니다. 둘 중 하나를 설정하면 값을 패치 가능하게 유지하기 위해 PDF/A 객체 스트림이 비활성화됩니다.

TsgcReport

Unit: sgcReports_Model
보고서 정의: 페이지, 밴드, 항목 및 스타일. `TComponent` 기반이며 `TCollection` 위에 구축되어 다른 컴포넌트와 마찬가지로 DFM으로 스트리밍되며, JSON으로 불러오거나 저장할 수도 있습니다. JSON 보고서 정의를 참조하십시오. 페이지, 밴드, 항목 및 스타일에 대한 전체 설명은 보고서 모델을 참조하십시오.

속성

속성	형식	설명
Units	TsgcReportUnits	모든 기하학적 속성(Left, Top, Width, Height, 여백)을 입력하는 단위입니다: ruMillimeters, ruPoints 또는 ruInches.
Pages	TsgcReportPages	TsgcReportPage의 컬렉션입니다.
스타일	TsgcReportStyles	TsgcReportStyle의 컬렉션이며, 밴드와 항목에서 StyleName으로 참조됩니다.
Title	string	보고서 제목이며, 기본 PDF 문서 제목이기도 합니다.
Author	string	보고서 작성자입니다.

예제

완전한 모델 API 예제(스타일, 페이지 1개, 밴드 1개, 텍스트 항목 1개)는 시작하기를, 전체 보고서는 사용법: 마스터-디테일 데이터를 사용한 인보이스를 참조하십시오.

TsgcReportEngine

Unit: sgcReports_Engine
TsgcReport를 해당 데이터 소스에 대해 실행합니다: 마스터-디테일과 그룹을 순회하고, 표현식과 집계를 평가하며, 내보내기와 미리보기 컴포넌트가 읽는 준비된 페이지 표시 목록을 생성합니다. `TComponent` 기반입니다.

속성

속성	형식	설명
Report	TsgcReport	실행할 보고서 정의입니다.
Variables	TStrings	Name=Value 형태의 사용자 매개변수이며, 표현식에서 단순 식별자로 노출됩니다.
FontManager	TsgcReportFontManager	게시되지 않음(public). Nil이면 공유된 sgcReportDefaultFontManager를 사용합니다. 글꼴을 참조하십시오.
OnBeforePrintBand	TsgcReportBeforePrintBandEvent	밴드가 인쇄되기 전에 발생하며, 핸들러가 표시 여부나 값을 변경할 수 있습니다.
OnGetVariable	TsgcReportGetVariableEvent	Variables에서 찾을 수 없는 식별자를 해결합니다.

메서드

메서드	반환값	설명
AddDataSource(aSource: TsgcReportDataSource)	—	데이터 소스를 SourceName으로 등록합니다. 이후 밴드 표현식은 이를 SourceName.Field로 참조합니다.
RemoveDataSource(aSource: TsgcReportDataSource)	—	데이터 소스 등록을 해제합니다.
ClearDataSources	—	등록된 모든 데이터 소스를 제거합니다.
Prepare	TsgcReportPreparedPages	보고서를 실행합니다. 호출자가 결과를 소유하고 해제합니다.
Prepare(aPages: TsgcReportPreparedPages)	—	기존의 먼저 지워진 TsgcReportPreparedPages로 준비하는 오버로드입니다.

예제

TsgcReportDataSetSource

Unit: sgcReports_Data
`TDataSet`(`TClientDataSet`, FireDAC, dbExpress 등)에서 레코드를 읽는 데이터 소스입니다. `TsgcReportDataSource`를 상속하며, 공통 마스터-디테일 속성인 `MasterSource`, `MasterFields` 및 `DetailFields`를 제공합니다. 데이터 소스를 참조하십시오.

속성

속성	형식	설명
DataSet	TDataSet	읽어들일 데이터셋입니다. 열려 있지 않으면 실행 중에 소스가 열고 닫으며, 이미 열려 있으면 실행 후 상태와 위치가 복원됩니다.
SourceName	string	상속됨. 밴드 표현식이 Source.Field로 사용하는 이름이며, 컴포넌트의 Name과 무관합니다.
MasterSource	TsgcReportDataSource	상속됨. 이 소스가 디테일로 속한 마스터 소스입니다.
MasterFields	string	상속됨. 세미콜론으로 구분된 마스터 필드 이름입니다.
DetailFields	string	상속됨. 세미콜론으로 구분된 디테일 필드 이름이며, MasterFields와 개수가 같습니다.

예제

이 소스 쌍을 사용하는 전체 보고서는 사용법: 마스터-디테일 데이터를 사용한 인보이스를 참조하십시오.

TsgcReportJSONSource

Unit: sgcReports_Data
문자열, 파일 이름 또는 스트림으로 주어진 객체의 JSON 배열에서 레코드를 읽는 데이터 소스입니다. `TsgcReportDataSource`를 상속하므로 `MasterSource`, `MasterFields` 및 `DetailFields`도 지원합니다.

속성

속성	형식	설명
JSON	string	직접 설정하는 경우의 JSON 텍스트입니다.
FileName	string	JSON을 읽어올 파일입니다.
RecordsPath	string	레코드 배열까지의 점(dot) 표기 경로이며, JSON 루트 자체가 배열인 경우 비워 둡니다.

메서드

메서드	설명
LoadFromStream(aStream: TStream)	스트림에서 즉시 UTF-8 JSON을 구문 분석합니다. 이후 JSON과 FileName 중 하나가 변경될 때까지 이를 대체합니다.

예제

"sales" 아래의 배열인 sales.json:
사용법: 합계가 있는 그룹 보고서를 참조하십시오.

TsgcReportEventSource

Unit: sgcReports_Data
`TDataSet`이나 JSON 형태가 없는 레코드, 예를 들어 요청 시 가져오는 페이지 단위 REST 응답을 위해 전적으로 이벤트로 구동되는 데이터 소스입니다. `OnEof` 핸들러가 없으면 소스에 레코드가 없습니다.

속성

속성	형식	설명
FieldNames	TStrings	이 소스가 노출하는 필드 이름입니다. 이를 알아낼 데이터셋이나 JSON 객체가 없기 때문입니다.
OnFirst	TNotifyEvent	반복을 첫 레코드로 재설정하기 위해 발생합니다.
OnNext	TNotifyEvent	다음 레코드로 이동하기 위해 발생합니다.
OnEof	TsgcReportEofEvent	`procedure(Sender: TObject; var aEof: Boolean) of object`. 더 이상 레코드가 없으면 aEof를 True로 설정합니다.
OnGetValue	TsgcReportGetValueEvent	`procedure(Sender: TObject; const aFieldName: string; var aValue: Variant) of object`. 현재 레코드의 한 필드 값을 반환합니다.

예제

`OnFirst`, `OnNext`, `OnEof` 및 `OnGetValue`는 메서드 포인터 이벤트로 선언되어 있으므로(`of object`로 선언), 익명 프로시저가 아니라 폼이나 클래스의 명명된 메서드를 받습니다. 이 예제는 `FIndex`와 `FRows` 두 필드를 사용하며, 이들은 아래 네 개의 핸들러 메서드와 함께 상위 폼에 선언되어 있습니다.
`FRows`는 `array [0..1] of TArray<string>`로 선언됩니다. 이 패턴은 익명 메서드가 아니라 일반 메서드 포인터를 할당하므로 지원되는 모든 Delphi 버전, Delphi 7부터 Delphi 13까지 동작합니다.

TsgcReportPDFExport

Unit: sgcReports_Export_PDF
sgcPDF를 통해 준비된 페이지를 렌더링합니다: 임베드 및 서브셋 TrueType 글꼴, 기본 14 글꼴, 오른쪽에서 왼쪽으로 쓰는 텍스트, 회전, 클리핑, 양쪽 정렬 텍스트, 도형, 이미지 및 벡터 바코드. 기본적으로 PDF/UA-1 태그가 지정된 출력을 생성하며, 선택적으로 PDF/A-3 아카이브 출력을 생성합니다. `TComponent` 기반입니다.

속성

속성	형식	기본값	설명
Compress	Boolean	True	객체 스트림과 압축된 상호 참조 테이블입니다. 콘텐츠 스트림은 항상 압축됩니다.
PDFA	TsgcReportPDFAMode	rpaNone	rpaNone, rpaA3b 또는 rpaA3a입니다. PDF/A를 참조하십시오.
PDFUA	Boolean	True	PDF/UA-1 구조 트리를 기록합니다. PDFA와 함께 사용할 수 있습니다. 접근성을 참조하십시오.
Title / Author / Subject / Keywords	string	—	PDF 문서 메타데이터입니다.
Language	string	—	태그가 지정된 내보내기가 기록하는 BCP 47 자연어입니다.
Creator / Producer	string	—	PDF 문서 메타데이터입니다.
OnAfterRender	TsgcReportPDFDocumentEvent	—	모든 페이지가 렌더링된 후, 문서가 저장되기 전에 한 번 발생합니다.
FontManager	TsgcReportFontManager	—	게시되지 않음(public). 글꼴을 참조하십시오.
FixedCreationDate	TDateTime	0	게시되지 않음(public). 바이트 단위로 동일한 출력을 위해 PDF/A 생성 및 수정 날짜를 고정합니다.
FixedDocumentID	string	—	게시되지 않음(public). /ID를 이 값의 MD5로 고정합니다.

메서드

메서드	설명
SaveToStream(aPages: TsgcReportPreparedPages; aStream: TStream)	준비된 페이지를 스트림으로 렌더링합니다.
SaveToFile(aPages: TsgcReportPreparedPages; const aFileName: string)	준비된 페이지를 파일로 렌더링합니다.

예제: 접근성 및 아카이브용 PDF

TsgcReportXLSXExport

Unit: sgcReports_Export_XLSX
준비된 페이지를 .xlsx 워크북의 워크시트 하나로 내보냅니다. `TsgcReportGridExport`를 상속하며, 이는 준비된 페이지를 그리드(밴드 인스턴스당 한 행, 허용 오차 내에서 항목 위치가 병합된 열)로 변환하는 기능으로 TsgcReportCSVExport와 공유됩니다. 패키지는 외부 zip 라이브러리 없이 작성된, 인라인 문자열을 사용하는 STORED 항목(Content_Types, 관계, 워크북, 시트 하나, 스타일)으로 구성된 ZIP입니다.

속성(고유)

속성	형식	설명
SheetName	string	워크시트 이름입니다.
FixedDateTime	TDateTime	ZIP 항목에 저장되는 수정 시간입니다. 0(기본값)은 현재 시간을 사용합니다. 바이트 단위로 동일한 출력을 위해 설정하십시오.

속성(TsgcReportGridExport에서 상속)

속성	형식	기본값	설명
ExcludePageBands	Boolean	True	페이지 머리글, 페이지 바닥글, 열 머리글 및 열 바닥글 밴드를 그리드에서 제외합니다.
IncludeBandTypes	TsgcReportBandTypes	rbtReportTitle..rbtChild	어떤 밴드 유형이 행을 생성하는지 지정합니다.
PageFrom / PageTo	Integer	0	1부터 시작하는 페이지 범위입니다. 0은 모든 페이지를 의미합니다.
DetectNumbers	Boolean	True	숫자로 해석되는 텍스트 셀은 숫자 셀로 기록됩니다.
ColumnTolerance	Double	—	이 값(포인트 단위)보다 가까운 왼쪽 위치는 같은 열로 취급됩니다.
DecimalSeparator / ThousandSeparator	Char	locale	게시되지 않음(public). DetectNumbers에서 사용됩니다.

메서드(상속)

메서드	설명
SaveToStream(aPages, aStream)	워크북을 스트림에 기록합니다.
SaveToFile(aPages, const aFileName)	워크북을 파일에 기록합니다.
BuildGrid(aPages): TsgcReportGrid	중간 그리드를 생성합니다. 호출자가 결과를 소유합니다.

예제

TsgcReportCSVExport

Unit: sgcReports_Export_CSV
TsgcReportXLSXExport와 동일한 레이아웃 기반 그리드를 사용하여 준비된 페이지를 CSV 텍스트로 내보냅니다(해당 페이지의 `TsgcReportGridExport` 속성인 ExcludePageBands, IncludeBandTypes, PageFrom, PageTo, DetectNumbers, ColumnTolerance 참조).

속성(고유)

속성	형식	기본값	설명
Delimiter	Char	','	필드 구분 기호입니다.
QuoteChar	Char	'"'	인용 문자입니다. #0은 인용을 비활성화합니다.
LineBreak	string	—	행 구분자입니다.
WriteBOM	Boolean	True	파일 시작 부분에 UTF-8 바이트 순서 표시(BOM)를 기록합니다.

메서드

메서드	반환값	설명
GetCSV(aPages: TsgcReportPreparedPages)	string	메모리에 CSV 텍스트를 생성합니다.
SaveToStream(aPages, aStream)	—	CSV를 스트림에 기록합니다.
SaveToFile(aPages, const aFileName)	—	TsgcReportGridExport에서 상속됩니다. CSV를 파일에 기록합니다.

예제

TsgcReportImageExport

Unit: sgcReports_Export_Image
sgcPDF 래스터 렌더러를 통해 모든 플랫폼에서 선택한 DPI로 페이지당 하나의 PNG 이미지를 내보냅니다. `TComponent` 기반입니다.

속성

속성	형식	기본값	설명
DPI	Integer	150	이미지의 인치당 픽셀 수입니다.
PageFrom / PageTo	Integer	0	SaveToFiles를 위한 1부터 시작하는 페이지 범위입니다. 0은 첫/마지막 페이지를 의미합니다.
BackgroundColor	TsgcReportColor	sgcReportColorWhite	렌더링 전에 채울 색상입니다. sgcReportColorNone은 흰색을 의미합니다.
FontManager	TsgcReportFontManager	—	게시되지 않음(public). 명시적인 서체가 없는 항목의 서체를 결정합니다.

메서드

메서드	반환값	설명
PageCount(aPages: TsgcReportPreparedPages)	Integer	SaveToFiles가 기록할 페이지 수이며, PageFrom / PageTo가 적용됩니다.
SaveToStream(aPages, aPageIndex: Integer, aStream: TStream)	—	한 페이지(0부터 시작하며, PageFrom / PageTo는 무시됨)를 PNG로 기록합니다.
SaveToFiles(aPages, const aFileNamePattern: string)	—	범위의 페이지마다 하나의 PNG 파일을 기록합니다. aFileNamePattern에는 %d가 포함되며, 1부터 시작하는 페이지 번호로 치환됩니다.
ClearCache	—	내부적으로 래스터화에 사용되는 캐시된 PDF를 삭제합니다. 다음 내보내기에서 다시 생성됩니다.

예제

TsgcReportPreviewVCL

Unit: sgcReports_Preview_VCL
탐색, 확대/축소, 인쇄 및 PDF 내보내기 기능을 갖춘 VCL 미리보기 패널입니다. `TCustomPanel`을 상속하며 내부적으로 `TsgcPDFViewerControl`을 호스팅하므로, 보고서는 미리보기를 위해 항상 PDF로 렌더링됩니다.

속성

속성	형식	기본값	설명
Engine	TsgcReportEngine	—	Prepare가 실행하는 엔진입니다.
ShowToolbar	Boolean	True	기본 제공되는 탐색 및 확대/축소 도구 모음을 표시합니다.
OnPageChange	TNotifyEvent	—	표시되는 페이지가 바뀌면 발생합니다.
PageCount	Integer	—	게시되지 않음(public), 읽기 전용입니다.
CurrentPage	Integer	—	게시되지 않음(public). 1부터 시작하며, 표시되는 내용이 없으면 0입니다.
Zoom	Double	—	게시되지 않음(public). 백분율이며 100이 실제 크기입니다.
FontManager	TsgcReportFontManager	—	게시되지 않음(public). LoadPrepared를 참조하십시오.
Viewer	TsgcPDFViewerControl	—	게시되지 않음(public), 읽기 전용입니다. 내부에서 사용되는 sgcPDF 뷰어입니다.

메서드

메서드	설명
Prepare	Engine.Prepare를 실행하고 결과를 표시합니다.
LoadPrepared(aPages: TsgcReportPreparedPages)	다른 곳에서 준비된 페이지를 표시하며, FontManager로 렌더링하거나, 그것이 없으면 Engine.FontManager로, 그것도 없으면 시스템 글꼴을 사용하는 내부 관리자로 렌더링합니다. 페이지가 준비될 때 사용한 관리자를 전달하십시오.
Clear	미리보기를 지웁니다.
FirstPage / PriorPage / NextPage / LastPage	탐색 기능입니다.
ZoomIn / ZoomOut / FitWidth / FitPage	확대/축소 제어입니다.
Print(const aTitle: string = '')	표시된 문서를 인쇄합니다.
SaveToPDF(const aFileName: string)	미리보기에 표시된 PDF를 기록합니다.

예제

TsgcReportPreviewFMX

Unit: sgcReports_Preview_FMX
데스크톱과 모바일을 위한 FireMonkey 미리보기이며, `TControl` 및 `TsgcPDFViewerFMX` 위에서 TsgcReportPreviewVCL을 그대로 반영합니다. Windows, Android 및 iOS에서 실행됩니다.

속성

속성	형식	기본값	설명
Engine	TsgcReportEngine	—	Prepare가 실행하는 엔진입니다.
ShowToolbar	Boolean	True	기본 제공되는 탐색 및 확대/축소 도구 모음을 표시합니다.
OnPageChange	TNotifyEvent	—	표시되는 페이지가 바뀌면 발생합니다.
PageCount	Integer	—	게시되지 않음(public), 읽기 전용입니다.
CurrentPage	Integer	—	게시되지 않음(public). 1부터 시작합니다.
Zoom	Double	—	게시되지 않음(public). 백분율이며 100이 실제 크기입니다.
Viewer	TsgcPDFViewerFMX	—	게시되지 않음(public), 읽기 전용입니다.

메서드와 나머지 표면(Prepare, LoadPrepared, 탐색, 확대/축소, SaveToPDF)은 TsgcReportPreviewVCL과 동일합니다. iOS 및 Android에서의 글꼴 배포를 포함한 전체 예제는 사용법: 모바일 FMX 보고서를 참조하십시오.

TsgcReportHTTPEndpoint

Unit: sgcReports_HTTP (package sgcReportsHTML)
등록된 보고서를 sgcWebSockets HTTP 서버를 통해 제공합니다: PDF, XLSX, CSV, PNG 페이지 및 정보 JSON을 화이트리스트로 지정된 쿼리 매개변수와 준비된 페이지 캐시와 함께 제공합니다. sgcWebSockets Professional 에디션 이상이 필요합니다. `TComponent` 기반입니다.

속성

속성	형식	기본값	설명
Server	TsgcHTTPServer_Base	—	연결할 HTTP 서버입니다. 선택 사항이며, 대신 자체 OnCommandGet에서 HandleRequest를 호출할 수 있습니다.
BasePath	string	—	엔드포인트가 소유하는 URL 접두사이며, 예를 들어 / reports입니다.
CacheSize	Integer	8	보고서당 유지되는 준비된 매개변수 세트의 수이며, 0은 캐시를 비활성화합니다.
MaxParamLength	Integer	256	허용되는 쿼리 매개변수 값의 최대 길이입니다.
DefaultDPI / MaxDPI	Integer	96 / 300	PNG 페이지 경로에 사용되고 허용되는 DPI입니다.
OnBeforePrepare	TsgcReportHTTPBeforePrepareEvent	—	요청에 대해 보고서가 준비되기 전에 발생하며, 매개변수를 검증하거나 거부할 수 있습니다(예: 403 응답).
DesignerEnabled	Boolean	False	TsgcReportHTMLDesigner가 사용하는 디자이너 경로(스키마, 정의, 미리보기)를 활성화합니다.
MaxDefinitionSize	Integer	1048576	허용되는 보고서 정의 본문의 최대 크기이며, 바이트 단위입니다.
OnBeforeSaveDefinition	TsgcReportHTTPBeforeSaveDefinitionEvent	—	디자이너가 전송한 정의가 저장되기 전에 발생합니다.

메서드

메서드	반환값	설명
RegisterReport(const aName: string; aEngine: TsgcReportEngine)	—	엔진을 이름으로 등록합니다. 엔진은 소유되지 않습니다. aName: 문자, 숫자, '_' 및 '-'.

UnregisterReport(const aName: string)	—	보고서 등록을 해제합니다.
FindEngine(const aName: string)	TsgcReportEngine	등록된 엔진을 조회합니다.
GetReportParameters(const aName, aList: TStrings)	Boolean	보고서의 엔진 Variables(Name=Value)를 복사합니다. 알 수 없는 이름이면 False를 반환합니다.
ClearCache(const aName: string = '')	—	보고서 하나의 캐시된 준비 페이지를 삭제하며, aName이 비어 있으면 모든 보고서의 캐시를 삭제합니다.
GetReportPath(const aName: string)	string	확장자가 없는 보고서의 URL 경로이며, 예를 들어 /reports/sales입니다.
HandleRequest(ARequestInfo, AResponseInfo)	Boolean	BasePath에 속하는 요청을 처리하고 True를 반환합니다. Server가 지정되지 않은 경우 자체 OnCommandGet에서 호출하십시오.

Routes

BasePath "/reports" 아래에 "sales"로 등록된 보고서의 경우: `/reports/sales.pdf`, `/reports/sales.xlsx`, `/reports/sales.csv`, `/reports/sales/page/1.png?dpi=96` 및 `/reports/sales/info`(JSON: 이름, 제목, 페이지 수, 매개변수)입니다. 엔진의 Variables에 이름이 있는 쿼리 매개변수만 보고서에 전달됩니다.

Designer routes

`DesignerEnabled`가 `True`일 때만 사용할 수 있으며(그렇지 않으면 모든 경로가 404를 응답합니다), TsgcReportHTMLDesigner가 사용합니다.

메서드	Route	설명
GET	BasePath/schema.json	디자이너의 속성 검사기를 구동하는 데 사용되는 보고서 정의의 JSON 스키마입니다.
GET	BasePath/{name}/definition	현재 보고서 정의를 JSON으로 반환합니다.
PUT or POST	BasePath/{name}/definition	본문을 검증하고 `OnBeforeSaveDefinition`을 발생시킨 다음, 등록된 엔진의 보고서를 교체합니다.
POST	BasePath/{name}/preview.png?page=1&dpi=96	저장된 보고서를 교체하지 않고 요청 본문의 정의를 렌더링합니다. 응답에는 `X-Report-Page-Count` 헤더가 포함됩니다.

정의 및 미리보기 경로의 요청 본문은 `application/json`이어야 하며 `MaxDefinitionSize`를 초과할 수 없습니다. 둘 중 하나라도 실패하거나 스키마 검증에 실패한 요청은 JSON 오류 본문으로 거부됩니다(`ok`, `path`, `message`). 여기서 `path`는 문제가 된 멤버의 JSON 경로입니다.
완전한 서버는 사용법: HTTP로 보고서 제공하기를, 디자이너 자체는 사용법: 웹 보고서 디자이너를 참조하십시오.

TsgcReportHTMLViewer

Unit: sgcReports_HTML_Viewer (package sgcReportsHTML)
TsgcReportHTTPEndpoint가 제공하는 보고서를 위한 브라우저 뷰어입니다: 페이지 탐색, 매개변수 입력 폼 및 다운로드 링크가 sgcHTML 페이지에 포함됩니다. `TsgcHTMLComponent`를 상속합니다.

속성

속성	형식	기본값	설명
Endpoint	TsgcReportHTTPEndpoint	—	뷰어가 보고서 경로를 읽어오는 엔드포인트입니다.
ReportName	string	—	엔드포인트에 등록된 이름으로, 표시할 보고서의 이름입니다.
DPI	Integer	96	페이지 이미지에 사용되는 DPI입니다.
ShowParameters	Boolean	—	보고서의 매개변수(엔진 Variables)를 입력하는 폼을 표시합니다.
ShowToolbar	Boolean	—	페이지 탐색과 다운로드 링크를 표시합니다.

예제

엔드포인트 옆의 "/reports"에서 뷰어를 "/"에 호스팅하는 사용법: HTTP로 보고서 제공하기를 참조하십시오.

TsgcReportHTMLDesigner

Unit: sgcReports_HTML_Designer (package sgcReportsHTML)
스키마에 대해 JSON 보고서 정의를 편집하는 브라우저 기반 밴드 디자이너로, 동일한 엔드포인트를 통해 실시간 미리보기를 표시합니다. `TsgcHTMLComponent`를 상속합니다. `TsgcReportHTTPEndpoint.DesignerEnabled := True`가 필요합니다. 사용법: 웹 보고서 디자이너를 참조하십시오.
이 컴포넌트가 렌더링하는 페이지는 단순한 폼 편집기가 아니라 완전한 드래그 앤 드롭 밴드 디자이너입니다.

- 현재 정의에 따라 표시되는 왼쪽의 페이지, 밴드 및 항목 트리입니다.
- 보고서 자체의 `Units`와 관계없이 항상 밀리미터 단위로 표시되는 밴드 캔버스이며, 항목을 포인터로 끌어서 크기를 조절하고 그리드에 맞춰 정렬하며 키보드로 미세 조정할 수 있습니다.
- `schema.json`으로부터 생성되는 속성 검사기이므로, 고정된 폼이 아니라 보고서 정의의 필드와 열거형을 그대로 따릅니다.
- 정의를 `{name}/definition`으로 전송하는 저장 기능이며, 서버가 이를 거부하면 오류 응답의 JSON 경로를 이용해 문제가 된 필드를 강조 표시합니다.
- 정의를 `{name}/preview.png`로 전송하는 미리보기 기능이며, 저장하지 않고 렌더링된 페이지를 표시합니다.
- 메모리상의 편집 기록에 대한 실행 취소 및 다시 실행, 그리고 현재 정의를 파일로 저장하는 JSON 다운로드 버튼입니다.

속성

속성	형식	설명
Endpoint	TsgcReportHTTPEndpoint	디자이너가 스키마, 정의 및 미리보기 경로를 읽어오는 엔드포인트입니다. `DesignerEnabled` 가 `True` 로 설정되어 있어야 합니다.
ReportName	string	편집 중인 보고서 정의의 이름입니다.
DesignerHeight	Integer	포함된 디자이너의 높이이며, CSS 픽셀 단위이고 기본값은 640입니다.

예제

TsgcReportBuilder

Unit: sgcReports_Fluent
코드 우선(code-first) 플루언트 API: `Self` 또는 범위가 지정된 하위 빌더를 반환하는 체인 호출로 스타일, 페이지, 밴드 및 항목을 구성하므로, 전체 보고서가 하나의 표현식 트리가 됩니다. Delphi 7에서도 동작합니다(익명 메서드가 필요 없는 일반 클래스). 생성자에 기존 보고서가 전달되지 않는 한 자체 `TsgcReport`와 `TsgcReportEngine`을 생성하고 소유합니다.

생성자

생성자	설명
Create	새 TsgcReport를 생성하고 소유합니다.
Create(aReport: TsgcReport)	기존 보고서에 빌드하며, 소유하지 않습니다.

메서드

메서드	반환값	설명
Title / Author / Subject / Keywords / Language(const aValue: string)	TsgcReportBuilder	일치하는 TsgcReport 속성을 설정합니다.
Units(aUnits: TsgcReportUnits)	TsgcReportBuilder	TsgcReport.Units를 설정합니다.
Style(const aName: string)	TsgcReportStyleBuilder	스타일 aName을 찾거나 추가합니다. 결과에 대해 Font, Bold, Italic, Underline, Color, BackColor, Frame, Padding을 체이닝할 수 있습니다.
Page(const aName: string = '')	TsgcReportPageBuilder	호출할 때마다 페이지를 추가합니다. PaperSize, Margins, Band 및 Text/Barcode/Image/Line 항목 호출을 체이닝할 수 있습니다.
Data(aSource: TsgcReportDataSource)	TsgcReportBuilder	엔진에 데이터 소스를 추가합니다. 소유하지 않습니다.
Variable(const aName, aValue: string)	TsgcReportBuilder	엔진 변수를 설정합니다.
FontManager(aManager: TsgcReportFontManager)	TsgcReportBuilder	엔진의 글꼴 관리자를 지정합니다. 소유하지 않습니다.
Flush	—	현재 밴드 또는 항목의 대기 중인 수정자를 기록합니다.
Prepare	TsgcReportPreparedPages	보고서를 준비합니다. 호출자가 결과를 소유합니다.
Prepare(aPages: TsgcReportPreparedPages)	—	기존 페이지 객체로 준비하는 오버로드입니다.
SaveToPDF(const aFileName) / SaveToPDFStream(aStream)	—	PDFExport를 통해 준비하고 내보냅니다. 저장할 때마다 보고서의 Title, Author,

		Subject, Keywords 및 Language가 PDFExport로 복사됩니다.
SaveToXLSX / SaveToXLSXStream	—	준비한 다음 XLSX로 내보냅니다.
SaveToCSV / SaveToCSVStream	—	준비한 다음 CSV로 내보냅니다.
SaveToPNG(const aFileNamePattern: string; aDPI: Integer = 150)	—	페이지당 하나의 PNG 파일을 생성합니다. aFileNamePattern에는 %d가 포함됩니다.

속성

속성	형식	설명
Report	TsgcReport	대기 중인 수정자가 기록된 보고서입니다.
Engine	TsgcReportEngine	빌더의 엔진이며, TsgcReportPreviewVCL / TsgcReportPreviewFMX에 연결할 수 있습니다.
PDFExport	TsgcReportPDFExport	처음 사용할 때 생성되며 빌더가 소유합니다. 저장하기 전에 PDFA, PDFUA 또는 Compress를 설정하십시오.

예제

최소 예제는 시작하기를, 플루언트 API만으로 작성된 완전한 보고서는 사용법: 합계가 있는 그룹 보고서를 참조하십시오.

JSON 보고서 정의

Unit: sgcReports_Definition
완전한 TsgcReport를 JSON으로 불러오고 저장하므로, 모델 API나 플루언트 API로 한 번 작성한 보고서를 재컴파일 없이 편집, 저장 및 재배포할 수 있습니다.

엔벨로프

엔벨로프에는 선택적 `"$schema"` 문자열 멤버를 사용할 수 있습니다.

규칙

- 모델의 모든 게시된 속성은 Delphi 이름을 lowerCamelCase로 변환하여 기록됩니다(`DataSourceName`은 `dataSourceName`으로, `HAlign`은 `hAlign`으로); 보고서의 컴포넌트 `Name`은 정의에 포함되지 않습니다.
- 열거형은 Delphi 식별자 그대로 기록되며(`"rbtMasterData"`), 집합은 식별자의 배열로, 불리언은 `true` / `false`로, 정수와 실수는 JSON 숫자로 기록됩니다. 색상은 `$00RRGGBB` 정수입니다.
- `TPersistent` 하위 객체는 중첩된 객체로, 컬렉션은 항목 객체의 배열로, `TStrings`는 문자열의 배열로 기록됩니다.
- 기본값과 같은 값은 DFM에서와 마찬가지로 생략됩니다.
- 이미지 항목에 포함된 이미지 바이트는 해당 `image` 객체의 base64 문자열 `pictureData`로 기록됩니다.

불러오기는 먼저 보고서를 지웁니다. 오류가 발생하면 문제가 된 멤버의 JSON 경로(예: `report.pages[0].bands[2].height`)와 함께 `ESgcReportDefinitionError`가 발생합니다. `IgnoreUnknown`이 `True`가 아닌 한 알 수 없는 멤버는 오류입니다. 저장 시 스트림과 파일은 BOM 없는 UTF-8이며, 불러올 때는 BOM이 있어도 허용됩니다.

함수

함수	설명
sgcReportSaveToJSON(aReport: TsgcReport; aIndent: Boolean = True): string	보고서를 JSON 문자열로 직렬화합니다.
sgcReportLoadFromJSON(aReport: TsgcReport; const aJSON: string)	JSON 문자열에서 보고서를 불러옵니다.
sgcReportSaveToJSONFile / sgcReportLoadFromJSONFile	파일 변형입니다.
sgcReportSaveToJSONStream / sgcReportLoadFromJSONStream	스트림 변형입니다.
sgcReportGenerateSchema: string	RTTI로부터 생성된 정의의 JSON 스키마 (draft 2020-12)이며, sgcReports.schema.json으로 사용됩니다.

TsgcReportDefinition

`IgnoreUnknown` 또는 `Indent` 제어가 필요한 호출자를 위해 일반 함수 뒤에 있는 클래스입니다: `SaveToJSON`, `LoadFromJSON`, `SaveToStream` / `LoadFromStream`, `SaveToFile` / `LoadFromFile`, 그리고 속성 `IgnoreUnknown`(기본값 False)과 `Indent`(기본값 True, 공백 2칸 들여쓰기와 LF 줄바꿈).

예제

definition.json과 data.json을 편집하는 시작하기 및 사용법: 웹 보고서 디자이너를 참조하십시오.

사용법: 마스터-디테일 데이터를 사용한 인보이스

이 항목은 `ReportInvoice` 데모(`delphi\Demos\Delphi\ReportInvoice`)를 단계별로 설명합니다: 두 개의 TsgcReportDataSetSource를 통해 읽은 두 개의 `TClientDataSet`(인보이스와 인보이스 라인)을 `MasterSource`로 연결하여, 반복되는 열 머리글, 소계, 부가세 및 총계, 인보이스 번호가 있는 QR 코드, 그리고 Page X of Y와 함께 인보이스당 한 페이지로 인쇄합니다. 보고서는 TsgcReportPreviewVCL에 표시되고, PDF/UA-1이기도 한 PDF/A-3a 파일로 내보내집니다.

두 소스 연결하기

인보이스당 한 페이지

각 인보이스를 시작하는 그룹 머리글은 `StartNewPage`를 설정하며, 그 `GroupExpression`은 인보이스 번호이므로, 값이 바뀔 때마다 새 그룹(및 새 페이지)이 시작됩니다:
그 안의 제목 항목은 PDF/UA를 위해 `rtrHeading1`로 태그가 지정되며, QR 코드의 값은 표현식 하나만이 아니라 인보이스 번호가 포함된 텍스트 템플릿이므로 주변 리터럴 텍스트도 유지됩니다:

라인 표와 반복되는 머리글

라인 표의 열 머리글은 동일한 `GroupExpression`에 대한 두 번째 `rbtGroupHeader` 밴드이며, `RepeatOnEveryPage := True`로 설정되어 있어 긴 인보이스의 라인이 두 번째 페이지로 이어질 때마다 다시 인쇄됩니다. 마스터 밴드는 인보이스를 순회하며 아무것도 인쇄하지 않고, 디테일 밴드는 라인을 순회합니다:

합계와 부가세

그룹 바닥글은 마지막으로 인쇄된 라인의 필드와 인보이스 자체의 필드를 보므로, 마지막으로 순회된 레코드가 라인이었더라도 부가세율은 `Invoices.VATRate`에서 바로 가져옵니다:

접근 가능하고 아카이브 가능한 PDF

완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportInvoice\sgcReports_Demo_Invoice.dpr`입니다.

사용법: 합계가 있는 그룹 보고서

이 항목은 `ReportFluent` 데모(`delphi\Demos\Delphi\ReportFluent`)를 단계별로 설명합니다: `sales.json`에서 읽은 판매 목록을 지역별로 그룹화하고, 지역별 소계와 총계를 TsgcReportBuilder만으로 작성합니다. JSON 소스, 모델 API로 지역 및 고객별로 그룹화한 동등한 예는 `ReportGroups` 데모입니다.

데이터 소스와 스타일

페이지, 제목 및 열 캡션

지역별 그룹화, 판매당 한 줄, 소계

`GroupHeader`는 단순 표현식(`'Sales.Region'`) 또는 `[ ]`로 감싼 표현식을 받습니다. 그룹 바닥글의 `Sum` 집계는 지역마다 재설정되며, 아래의 요약 밴드에서 같은 표현식은 보고서 전체에 대해 실행됩니다.

총계와 페이지 번호

미리보기와 내보내기

완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportFluent\sgcReports_Demo_Fluent.dpr`입니다. `TDataSet`이나 모델 API 소스를 사용한 동일한 그룹화 및 합계 패턴은 `ReportGroups`에서도 볼 수 있습니다.

사용법: XLSX, CSV 및 PNG로 내보내기

이 항목은 `ReportExports` 데모(`delphi\Demos\Delphi\ReportExports`)를 단계별로 설명합니다: 보고서 하나를 한 번 준비한 다음 네 개의 서로 다른 컴포넌트를 통해 내보냅니다. 준비된 페이지(`TsgcReportPreparedPages`)는 모든 내보내기에서 재사용되므로, 레이아웃과 데이터 순회는 한 번만 실행됩니다.

PDF

XLSX

TsgcReportXLSXExport는 워크북을 기록하기 전에 준비된 페이지를 그리드(밴드 인스턴스당 워크시트 한 행)로 변환합니다:

CSV

PNG, 페이지당 파일 하나

참고

- 기본적으로(`ExcludePageBands = True`) 페이지 및 열 머리글과 바닥글은 XLSX 및 CSV 내보내기에서 그리드 행이 되지 않으며, 보고서 제목, 그룹 및 데이터 밴드만 됩니다.
- `DetectNumbers`(기본값 `True`)는 숫자로 해석되는 텍스트 셀을 텍스트가 아니라 숫자 XLSX 셀로 기록합니다.
- `SaveToFiles`는 패턴의 `%d`를 1부터 시작하는 페이지 번호로 치환하여 파일 이름을 지정하므로, `orders_%d.png`는 `orders_1.png`, `orders_2.png` 등을 생성합니다.

완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportExports\sgcReports_Demo_Exports.dpr`입니다.

사용법: veraPDF로 검증된 접근성 PDF

이 항목은 `ReportPDFUA` 데모(`delphi\Demos\Delphi\ReportPDFUA`)를 단계별로 설명합니다: H1 제목, H2 그룹 제목, 머리글 행이 있는 표, 대체 텍스트가 있는 이미지, 바코드를 PDF/UA-1로 내보낸 다음 PDF/A-3b로, 마지막으로 PDF/A-3a와 PDF/UA-1을 함께 내보냅니다. 태깅 규칙과 적합성 수준은 접근성과 PDF/A를 참조하십시오.

콘텐츠 태깅

그룹 또는 열 머리글 밴드 다음에 상세 행이 이어지는 표는 자동으로 태그가 지정됩니다. 머리글 셀이 `rtrTableHeader`로 표시되는 데 별도로 필요한 작업은 없습니다.

세 가지 변형 내보내기

veraPDF로 검증

veraPDF는 sgcReports와 별도로 명령줄 도구로 실행되는, ISO 기반의 오픈 소스 PDF/A 및 PDF/UA 검증기입니다. 각 파일을 해당하는 규격에 대해 확인하십시오:
통과한 실행은 요청한 규격에 대해 `compliant: true`를 보고합니다. 실패하면 위반한 ISO 조항과 객체를 나열하며, 이는 누락된 `AltText`나 `rtrArtifact`로 태그가 지정되었어야 할 항목을 찾는 가장 빠른 방법입니다.
완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportPDFUA\sgcReports_Demo_PDFUA.dpr`입니다.

사용법: HTTP로 보고서 제공하기

이 항목은 `ReportHTTPServer` 데모(`delphi\Demos\Delphi\ReportHTTPServer`)를 단계별로 설명합니다: 포트 8080의 `TsgcWebSocketHTTPServer`에 "sales" 보고서를 제공하는 TsgcReportHTTPEndpoint가 있으며, TsgcReportHTMLViewer를 호스팅하는 sgcHTML 페이지가 `/`에 있습니다. `sgcReportsHTML` 패키지가 필요합니다: sgcWebSockets(Professional 에디션 이상), sgcHTML 및 sgcReports.

보고서 등록 및 서버 연결

제공되는 경로

- `/reports/sales.pdf`, `/reports/sales.xlsx`, `/reports/sales.csv`
- `/reports/sales/page/1.png?dpi=96`
- `/reports/sales/info`(JSON: 이름, 제목, 페이지 수, 매개변수)

매개변수 제한하기

엔드포인트는 보고서 엔진의 `Variables`에 이름이 있는 쿼리 매개변수만 허용하므로, `Region`이라는 변수 하나를 가진 보고서는 `?Region=...`만 허용합니다. `OnBeforePrepare`는 값을 추가로 검증할 수 있으며, 예를 들어 보고서가 준비되기 전에 허용 목록 밖의 값을 거부할 수 있습니다:

뷰어 페이지 제공하기

빌드 참고 사항

검색 경로에 sgcWebSockets의 `Source` 폴더를 넣고 컴파일하십시오. sgcIndy 포크로 빌드된 배포판 sgcWebSockets 패키지에 링크할 때는 `SGC_CUSTOM_INDY`를, IDE에 포함된 Indy로 sgcWebSockets가 빌드된 경우에는 `SGC_REPORTS_STOCK_INDY`를 정의하십시오.
완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportHTTPServer\sgcReports_Demo_HTTPServer.dpr`입니다.

사용법: Linux64 보고서 데몬

이 항목은 `ReportLinuxDaemon` 데모(`delphi\Demos\Delphi\ReportLinuxDaemon`)를 단계별로 설명합니다. sgcReports 코어는 VCL이나 FMX에 의존하지 않으므로 Windows와 Linux64에서 변경 없이 실행되는 콘솔 애플리케이션입니다. 사용하는 풀링 패턴은 스레드 안전성과 서버 사용을 참조하십시오.

작업 파일

데몬은 작업 폴더에서 `*.json` 작업 파일을 감시합니다. 작업은 보고서 정의, 데이터 파일 및 출력 형식(`pdf`, `xlsx`, `csv` 또는 `png`)를 지정합니다. 작업 내 상대 경로는 작업 파일 자체의 폴더를 기준으로 해석됩니다. 작업은 이름을 `.json.working`으로 변경하여 확보되며, 출력은 그 옆에 기록되고, 작업은 `.json.done` 또는 `.json.failed`로 끝납니다.

워커 풀

4개의 워커 스레드 풀이 작업을 동시에 렌더링합니다. 각 스레드는 `sgcReportLoadFromJSON`으로 작업의 보고서 정의를 자신만의 `TsgcReport`에 로드하고, 자신만의 `TsgcReportEngine`과 데이터 소스를 생성하고, 준비하고, 내보낸 다음, 생성한 모든 것을 해제합니다. 이는 스레드 안전성과 서버 사용에서 설명한 공유 읽기 전용 패턴을 따릅니다.

실행하기

Linux에서는 SIGTERM 또는 SIGINT, Windows에서는 Ctrl+C가 데몬을 깔끔하게 중지합니다: 새 작업은 확보되지 않고, 실행 중인 작업은 완료되며, 프로세스는 코드 0으로 종료됩니다.

systemd 유닛

`sgcreports-daemon.service`는 전용 사용자로 컴파일된 실행 파일을 실행하며, `/var/lib/sgcreports/jobs`를 감시하고, 실패 시 재시작하며, 실행 중인 작업을 완료하기 위해 60초의 제한 시간을 두고 SIGTERM으로 중지합니다. 로그는 표준 출력으로 전송되며 다음으로 읽습니다:
완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportLinuxDaemon\sgcReports_Demo_LinuxDaemon.dpr`입니다.

사용법: 모바일 FMX 보고서

이 항목은 `ReportMobileFMX` 데모(`delphi\Demos\Delphi\ReportMobileFMX`)를 단계별로 설명합니다. 플루언트 API로 작성되어 TsgcReportPreviewFMX에 표시되는 작은 A5 배송 전표(전표 번호가 있는 QR 코드, 품목 줄, 총 수량, 서명란)입니다. 하나의 프로젝트로 Win32, Win64, Android64 및 iOS를 대상으로 합니다.

저장 위치

PDF 저장 버튼은 `sgcReports_DeliveryNote.pdf`를 앱의 문서 폴더에 기록하고 경로를 표시합니다:

모바일에서의 글꼴

이 데모는 글꼴 검색을 통해 시스템에 이미 있는 글꼴을 사용합니다(글꼴 참조). iOS는 샌드박스 앱에서 읽을 수 있는 시스템 글꼴 폴더가 없으므로, 적합한 TrueType 글꼴을 찾지 못하면 데모는 대신 자신의 폴더에서 `DejaVuSans.ttf`를 로드합니다. `DejaVu-LICENSE.txt`는 이 글꼴의 라이선스이며 함께 제공됩니다.

글꼴 배포하기

IDE에서 Project > Deployment를 열고, 모든 모바일 플랫폼에 대해 다음 원격 경로로 `DejaVuSans.ttf`를 추가하십시오:

플랫폼	원격 경로
Android64	assets\internal\
iOS	StartUp\Documents\

Windows에서는 실행 파일의 폴더에서 글꼴을 읽으므로, 시스템에 적합한 글꼴이 설치되어 있지 않으면 그 폴더에 복사하십시오.
완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportMobileFMX\sgcReports_Demo_MobileFMX.dpr`입니다. sgcReports, sgcPDF 및 FireMonkey가 필요합니다.

사용법: 웹 보고서 디자이너

TsgcReportHTMLDesigner는 `sgcReports.schema.json`에 대해 JSON 보고서 정의를 편집하는 브라우저 기반 밴드 디자이너이며, 뷰어나 애플리케이션이 완성된 보고서를 제공하는 데 사용하는 것과 동일한 TsgcReportHTTPEndpoint가 실시간 미리보기를 제공합니다.

활성화하기

`DesignerEnabled`는 디자이너 페이지가 호출하는 스키마, 정의 및 미리보기 경로를 활성화합니다. 완성된 보고서만 제공하는 프로덕션 엔드포인트에서는 `False`로 두고, 편집이 의도된 곳에서만 자체 인증으로 보호하여 활성화하십시오.

브라우저 디자이너가 제공하는 기능

`TsgcReportHTMLDesigner`가 제공하는 페이지는 완전한 드래그 앤 드롭 밴드 디자이너입니다: 페이지, 밴드 및 항목 트리; 드래그, 크기 조절, 스냅 및 키보드 미세 조정이 가능한 밀리미터 단위의 밴드 캔버스; `schema.json`으로부터 생성되는 속성 검사기; 저장(서버 측 검증 오류의 JSON 경로로부터 문제가 된 필드가 강조 표시됨)과 미리보기; 실행 취소 및 다시 실행; 그리고 JSON 다운로드 버튼입니다. 전체 속성 목록은 TsgcReportHTMLDesigner를, 디자이너가 호출하는 경로(`schema.json`, `{name}/definition` 및 `{name}/preview.png`)는 TsgcReportHTTPEndpoint를 참조하십시오.

브라우저 디자이너 없이 정의 편집하기

`ReportJSONDefinition` 데모(`delphi\Demos\Delphi\ReportJSONDefinition`)는 Delphi 코드에서 동일한 왕복 처리를 보여주며, 데스크톱 편집기나 정의의 자동화된 테스트에 유용합니다: 두 개의 메모에서 `definition.json`과 그 `data.json`을 편집하고, 렌더링 버튼은 `sgcReportLoadFromJSON`으로 정의를 로드하여 미리보기를 표시하며, 오류는 문제가 된 멤버의 JSON 경로와 함께 상태 표시줄에 보고되며(JSON 보고서 정의 참조), 정의 저장 버튼은 `sgcReportSaveToJSON`으로 현재 보고서를 JSON으로 다시 기록합니다.
코드 기반 편집기의 완전하고 컴파일 가능한 프로젝트는 `delphi\Demos\Delphi\ReportJSONDefinition\sgcReports_Demo_JSONDefinition.dpr`입니다.

소개

sgcCharts는 실시간 데이터 차트를 위한 Delphi 컴포넌트 라이브러리입니다. 데이터 소스는 점 또는 OHLC 캔들을 링 버퍼에 추가하고, 캔버스는 그 버퍼의 디시메이션된 스냅숏을 그리므로, 초당 수천 건의 갱신을 발행하는 피드도 안정적인 프레임 속도로 렌더링됩니다.
이 라이브러리는 VCL, FireMonkey, 그리고 창이 없는 헤드리스 렌더링(PNG 또는 SVG 출력)을 대상으로 하므로, 동일한 버퍼와 데이터 소스 코드가 폼, 모바일 화면, 또는 콘솔 애플리케이션이나 서비스 안에서 그려집니다.
이 도움말은 현재 존재하는 sgcCharts의 부분을 다룹니다: 버퍼에 데이터를 공급하는 데이터 소스(TsgcChartDataSource와 그 하위 클래스), 실시간 연결에서 버퍼로 데이터를 공급하는 sgcWebSockets 브리지(Bridge overview), 그리고 버퍼와 디시메이션 함수 자체(TsgcChartBuffer)입니다.

- Overview: 모델, 캔버스, 버퍼, 데이터 소스, 그리고 이들이 어떻게 맞물리는지입니다.
- Editions: Registered, Trial, Community 각각에 포함되는 내용입니다.
- Quick Start: 시작하기 경로의 현재 상태입니다.
- Data Sources: 수동, 데이터셋, JSON, CSV입니다.
- Bridge: sgcWebSockets를 통한 MQTT, WebSocket, SSE, Binance입니다.
- Reference: 버퍼 클래스와 디시메이션 함수입니다.

아키텍처

sgcCharts는 차트가 표시하는 데이터를 그것이 그려지는 표면과 분리하므로, 동일한 데이터가 VCL 폼, FireMonkey 뷰, 또는 헤드리스 PNG/SVG 렌더링에 공급될 수 있습니다.

하나의 모델, 여러 개의 캔버스

차트의 데이터는 컨트롤이 아니라 버퍼에 있습니다. **TsgcChartBuffer**는 X/Y 점을 보관하고 **TsgcChartOHLCBuffer**는 OHLC 캔들(시가, 고가, 저가, 종가, 거래량)을 보관합니다. 둘 다 VCL이나 FireMonkey에 의존하지 않는 단순한 클래스이므로, 콘솔 애플리케이션, 서비스, 백그라운드 스레드에서도 생성하고, 데이터를 추가하고, 읽을 수 있습니다. 그리기는 별도의 단계입니다. 캔버스는 버퍼의 디시메이션된 스냅숏을 읽어 그리며, 공유되는 `sgcUICore_Canvas` 표면 뒤에서 래스터 캔버스(`sgcUICore_Canvas_Raster`)나 SVG 캔버스(`sgcUICore_Canvas_SVG`)를 사용합니다.
이 차트를 그리는 캔버스, 래스터, TrueType 글꼴, SVG 및 테마 계층은 sgcCharts 자체에 포함되지 않습니다. 이 계층은 공유 패키지 **sgcUICore**입니다(런타임 패키지 sgcUICore의 Delphi 유닛 `sgcUICore_*`, .NET 패키지 `esegece.sgcUICore` 및 `esegece.sgcUICore.Rendering`). sgcCharts와 함께 자동으로 설치되며 eSeGeCe의 다른 비주얼 제품과 공유됩니다.

최소/최대 디시메이션을 사용하는 링 버퍼

두 버퍼 클래스 모두 고정 용량의 링 버퍼입니다. `cbmOverwrite` 모드에서는 버퍼가 가득 차면 다음 추가 시 가장 오래된 점을 삭제하고, `cbmGrow` 모드에서는 대신 용량이 두 배로 늘어납니다. 버퍼의 모든 공개 메서드는 내부 잠금을 사용하므로, 피드 스레드가 데이터를 추가하는 동안 렌더링 스레드가 스냅숏을 읽어도 서로 충돌하지 않습니다. **Snapshot**과 **SnapshotRange**는 호출자가 필요로 하는 데이터를 그 잠금 하에서 호출자가 소유한 레코드로 복사한 뒤 즉시 잠금을 해제합니다. 이후 호출자는 잠금 밖에서 그 복사본으로부터 디시메이션과 그리기를 수행합니다.
`sgcChartDecimate`는 스냅숏을 픽셀 열마다 대략 한 세트의 점으로 줄이며, `dmMinMax`(각 열의 첫 번째, 최소, 최대, 마지막 점을 유지하여 실시간 꺾은선을 시각적으로 정확하게 유지)나 `dmLTTB`(정적 시리즈를 위한 Largest Triangle Three Buckets)를 사용합니다. `sgcChartDecimateOHLC`는 같은 픽셀 열에 속하는 캔들들을 같은 방식으로 병합합니다. TsgcChartBuffer와 Decimation을 참조하십시오.

차트, 시리즈, 축

TsgcChart는 차트 모델입니다. 축, 시리즈, 제목, 범례, 여백, 테마, 팔레트를 보유하는 비시각적 컴포넌트이지만 그 자체로는 아무것도 그리지 않습니다. 시리즈는 차트에 중첩된 부분이 아니라 그 자체가 컴포넌트입니다. 시리즈의 **Chart** 속성을 지정하는 것이 시리즈를 등록하는 방법이며, 시리즈는 차트를 거치지 않고 (구체적인 시리즈 형식 중 하나를 통해) 자신의 버퍼를 직접 읽습니다. 뷰는 **TsgcChart.Render**를 호출하여 모델 전체를 TsgcUICanvas에 그리므로, 동일한 시리즈와 축을 가진 동일한 차트를 하나 이상의 뷰로 표시하거나, 뷰 없이 PNG나 SVG로 헤드리스 렌더링할 수도 있습니다. TsgcChart와 TsgcChartSeries를 참조하십시오.

데이터 소스

TsgcChartDataSource의 하위 클래스는 어딘가(애플리케이션이 직접 호출하는 **AppendValue**, `TDataSet`, JSON 문서, CSV 텍스트)에서 레코드를 읽어, 자신의 **Buffer**와 **OHLCBuffer** 속성에 지정된 버퍼에 해당 점이나 캔들을 추가합니다. 레코드의 필드가 어디에 있는지는 **Mapping**에 한 번만 기술되며, 모든 데이터 소스가 동일한 방식으로 이를 읽습니다. Data Sources를 참조하십시오.

sgcWebSockets 브리지

선택적인 `sgcChartsWS` 패키지는 실시간 sgcWebSockets 연결에서 읽어 오는 데이터 소스를 추가합니다: MQTT 토픽, WebSocket 메시지 스트림, Server-Sent Events 스트림, 또는 Binance 공개 시장 데이터 API입니다. 각각은 지정된 클라이언트 컴포넌트의 기존 이벤트를 후크하므로, 애플리케이션 자체의 핸들러도 차트와 함께 계속 실행됩니다. 이 패키지는 sgcWebSockets를 필요로 하지만, 핵심 `sgcCharts` 패키지는 그렇지 않습니다. Bridge를 참조하십시오.

에디션

Registered, Trial, Community 각각에 포함되는 내용은 Editions를 참조하십시오.

에디션

sgcCharts는 모든 에디션에서 동일한 소스로 빌드됩니다. 에디션은 빌드에 포함되는 유닛만 다르게 만듭니다.

에디션	포함 내용
Registered	모든 유닛: 핵심 데이터 소스, 재무 지표와 패널, 그리고 sgcWebSockets 브리지를 포함한 실시간(스트리밍) 데이터 소스입니다. 트라이얼 알림이 없습니다.
Trial	Registered와 동일한 유닛입니다. `SGC_TRIAL` 로 컴파일된 빌드는 빌드에 지정된 릴리스 기간이 지나면 일회성 알림을 표시하며, 그 외에는 아무것도 달라지지 않습니다.
Community	기능 제한이 전혀 없는 완전한 차팅 코어입니다: 여덟 개 시리즈 전부(라인, 고속 라인, 계단 라인, 영역, 막대/열, 산점도, 캔들스틱, OHLC), VCL/FMX/헤드리스 뷰, 버퍼, 핵심 데이터 소스(수동, 데이터셋, JSON, CSV), 상호작용입니다. `SGC_COMMUNITY` 로 컴파일된 빌드는 재무 팩(패널, 지표 시리즈, 거래량 시리즈, 애그리게이터, 그리기 도구, 최근값 마커, 매수/매도 밴드)을 제외하며, Community 에디션은 sgcWebSockets 브리지 유닛과 패키지를 제공하지 않습니다.

에디션은 `sgcVer.inc`의 컴파일 타임 선택(`SGC_COMMUNITY`, 이는 `SGC_TRIAL`도 함께 끕니다)이며, 런타임 전환 스위치가 아닙니다.

설치

sgcCharts는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcCharts**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcCharts에 포함된 항목

- **sgcUICore**는 sgcCharts와 레이아웃 제품의 공통 기반입니다. 자체 확인란은 없습니다. sgcCharts를 선택하면 설치 프로그램이 자동으로 설치합니다.
- **sgcCharts for sgcWebSockets**는 **sgcWebSockets bridges** 아래에 표시됩니다. sgcWebSockets 브리지의 실시간 데이터 소스가 포함된 sgcChartsWS 패키지를 설치합니다. 설치 프로그램은 sgcWebSockets가 이미 컴퓨터에 설치되어 있고 라이선스에 sgcCharts가 포함된 경우에만 이 항목을 제공합니다. 그렇지 않으면 회색으로 표시됩니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드입니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Charts`	sgcCharts 데모입니다. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcCharts도 포함됩니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 sgcCharts가 포함되며, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard도 함께 포함됩니다. sgcWebSockets 브리지는 Community 설치 프로그램에 포함되지 않으며, sgcPDF와 sgcMaps도 Community에 포함되지 않습니다. Community 빌드에서 빠지는 기능은 에디션을 참조하십시오.

독립 실행형 sgcCharts에서 마이그레이션하기

sgcCharts가 이전에 자체 독립 실행형 설치 프로그램으로 설치된 경우, sgcUI 설치 프로그램은 그 설치와 독립 실행형 설치 프로그램이 만든 sgcUICore 폴더를 감지합니다. 그리고 그 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다. 이 페이지는 이전 설치가 발견된 경우에만 표시됩니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcChartsD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcCharts<V>` / `dclsgcCharts<V>`	sgcCharts 컴포넌트(런타임/디자인 타임).
`sgcChartsWS<V>` / `dclsgcChartsWS<V>`	sgcWebSockets 브리지(런타임/디자인 타임). sgcWebSockets가 필요합니다.

의존 순서대로 빌드합니다. `sgcUICore`, 그다음 `sgcCharts`, 그다음 `dclsgcCharts`, 마지막으로 브리지 패키지입니다. 그런 다음 `dcl*` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder의 경우 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcChartsC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

.NET

.NET에서는 달라지는 것이 없습니다. sgcCharts는 여전히 자체 NuGet 패키지(`esegece.sgcCharts`와 그 보조 패키지)로 배포됩니다. .NET 개요를 참조하십시오.

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows(Win32 및 Win64)에서 TsgcVCLChart. 모든 Delphi 버전을 지원합니다.
FireMonkey	Delphi XE6부터 TsgcFMXChart(SGC_CHARTS_FMX). Win32, Win64, macOS, iOS, Android를 지원합니다.
Linux64	프레임워크에 독립적인 유닛, 즉 TsgcChart와 헤드리스 PNG 및 SVG 출력을 갖춘 TsgcChartRenderer. Delphi 10.2부터 지원합니다(SGC_CHARTS_LINUX).
.NET	WinForms, WPF, Avalonia용 C# 포트와 헤드리스 PNG 및 SVG 출력을 제공합니다. .NET 개요를 참조하십시오.

패키지

sgcCharts에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcChartsD13.dpk, C++Builder 13에서는 sgcChartsC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcCharts	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, dbrtl, sgcUICore
sgcChartsVCL	런타임, VCL	sgcCharts_VCL (TsgcVCLChart)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcCharts
sgcChartsFMX	런타임, FireMonkey, XE6부터	sgcCharts_FMX (TsgcFMXChart)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcCharts
dclsgcCharts	디자인 타임, VCL	**SGC Charts** 팔레트 페이지에 등록하는 컴포넌트는 TsgcChart, TsgcChartRenderer 및 TsgcVCLChart입니다.	designide, rtl, vcl, sgcCharts, sgcChartsVCL
dclsgcChartsFMX	디자인 타임, FireMonkey, XE6부터	**SGC Charts** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXChart입니다.	designide, rtl, fmx, sgcCharts, sgcChartsFMX, dclsgcCharts
sgcChartsWS, dclsgcChartsWS	런타임 및 디자인 타임, sgcWebSockets 브리지	라이브 데이터 브리지, 프레임워크 독립.	rtl, sgcCharts, sgcWebSockets

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcChartsFMX에는 dclsgcCharts가 필요하므로 dclsgcCharts를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcCharts	sgcChartsVCL	sgcChartsFMX	dclsgcCharts, dclsgcChartsFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcCharts, sgcChartsVCL	sgcChartsFMX	dclsgcCharts, dclsgcChartsFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcCharts만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcCharts만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcChartsC13.cbproj, sgcChartsVCLC13.cbproj, sgcChartsFMXC13.cbproj, dclsgcChartsC13.cbproj, dclsgcChartsFMXC13.cbproj 입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXChart)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcChartsVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcCharts_FMX)을 uses 절에 추가합니다.

- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcCharts는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 TsgcChart와, 창 없이 차트를 PNG 또는 SVG로 렌더링하는 TsgcChartRenderer입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

참고 항목

- 설치
- 에디션
- 아키텍처
- 문제 해결

Quick Start

sgcCharts는 차트 **모델**을 이를 그리는 **뷰**와 분리합니다. TsgcChart는 축, 시리즈, 데이터를 보유하지만 그 자체로는 아무것도 그리지 않습니다. VCL 컨트롤, FireMonkey 컨트롤, 또는 헤드리스 렌더러인 뷰가 모델이 변경될 때마다 그것을 그립니다. 이 안내는 하나의 차트를 단계별로 만들어 갑니다: 먼저 VCL 폼에서, 그다음 실시간 데이터로 공급하고, 마지막으로 서비스나 콘솔 애플리케이션을 위해 뷰 없이 렌더링합니다.

1. 차트 모델과 VCL 뷰

TsgcChart를 만들고 하단 축과 왼쪽 축을 지정한 뒤, **TsgcVCLChart**를 만들고 그 **Chart** 속성이 모델을 가리키게 합니다. 뷰는 스스로를 리스너로 등록하고 모델이 변경될 때마다 다시 그립니다. 폼에서 빈 차트를 보는 데는 그 외에 아무것도 필요하지 않습니다.

2. 시리즈와 정적 데이터 추가

시리즈는 그 자체가 컴포넌트입니다. TsgcChartSeries_Line을 만들고, 그 **Chart** 속성이 모델을 가리키게 하고, 시리즈가 그려질 축을 지정한 뒤, 점마다 한 번씩 **AddXY**를 호출합니다. 뷰가 만들어진 뒤, 위의 생성자에 아래 두 줄을 추가합니다.

3. 실시간 데이터 공급

폼이 표시된 이후에도 계속 점을 추가하려면 `TTimer`를 배치하고 그 **OnTimer** 이벤트에서 **AddY**를 호출합니다. **AddY**는 다음 순차 X 값에 점을 추가하며, 이는 X가 이미 가진 값이 아니라 샘플 인덱스나 틱 카운트일 때 편리합니다.
AddY를 호출하는 타이머는 빠른 테스트에는 적합하지만, 실시간 피드를 차트로 표시하는 실제 애플리케이션은 대신 데이터 소스 컴포넌트 중 하나를 사용해야 합니다: 애플리케이션이 이미 가지고 있는 값에는 **TsgcChartDataSource_Manual**을, `TDataSet` 에 는 **TsgcChartDataSource_DataSet** 을, 문 서 나 파 일 에 는 **TsgcChartDataSource_JSON** / **TsgcChartDataSource_CSV**를 사용합니다. 이들은 스레드에 안전한 추가, 제한된 알림, (데이터셋 소스의 경우) 단순 타이머에는 없는 필드 매핑을 제공합니다. MQTT, WebSocket, Server-Sent Events, 또는 Binance 시장 스트림처럼 이미 도착하는 피드에는 브리지 데이터 소스가 대응하는 sgcWebSockets 클라이언트에 곧바로 연결되며 폴링이 전혀 필요하지 않습니다.
1단계부터 3단계까지를 종합하면 아래의 완전하고 컴파일 가능한 프로그램이 됩니다. 이 폼은 `.dfm`을 전혀 사용하지 않고 코드로 자신의 레이아웃을 구성하므로, 생성자가 일반적인 상속된 **Create** 대신 **CreateNew**를 호출합니다.

4. 뷰 없는 헤드리스 렌더링

서비스, ISAPI 모듈, 콘솔 애플리케이션에는 뷰를 배치할 폼이 없으며, 필요하지도 않습니다. 동일한 **TsgcChart**와 시리즈가 **TsgcVCLChart** 대신 TsgcChartRenderer를 통해 렌더링됩니다. **TsgcChartRenderer**는 VCL이나 FMX에 의존하지 않으므로, 이 프로그램은 sgcCharts 핵심 유닛 외에는 아무것도 링크하지 않습니다.
실행하면 창도, 메시지 루프도, 데스크톱도 필요 없이 실행 파일 옆에 `quickstart.png`가 작성됩니다. SVG로, 스트림으로, 또는 원시 픽셀 버퍼로 렌더링하는 방법은 TsgcChartRenderer를 참조하십시오.

다음 단계

- Series: 라인, 고속 라인, 계단 라인, 영역, 막대, 산점도, 캔들스틱 시리즈, 그리고 각각이 그리는 것입니다.
- Views: **TsgcVCLChart**, **TsgcFMXChart**, **TsgcChartRenderer**입니다.
- Interaction: 시각적 뷰에서의 확대/축소, 팬, 크로스헤어, 툴팁입니다.
- Financial: 지표, 오실레이터, 거래량, 다중 패널 재무 차트입니다.
- Data Sources: 수동, 데이터셋, JSON, CSV 피드입니다.
- Bridge Data Sources: sgcWebSockets를 통한 실시간 MQTT, WebSocket, SSE, Binance 피드입니다.
- Demos: 이 quick start의 기반이 된 실행 가능한 데모 프로젝트입니다.

참고 항목

- TsgcChart

- TsgcChartSeries
- TsgcVCLChart
- TsgcChartRenderer
- TsgcChartBuffer

데모

sgcCharts 배포판은 이 도움말의 모든 시나리오에 대한 데모 프로젝트를 제공합니다: 실시간 피드, 다중 축, CSV에서 불러온 캔들스틱 차트, 데이터셋에 바인딩된 차트, 테마와 HiDPI, sgcWebSockets에서 브리지된 실시간 피드, FireMonkey 모바일 차트, 그리고 뷰가 전혀 없는 헤드리스 렌더링입니다. Delphi 프로젝트는 `Demos\Delphi\<Name>\` 아래에, C++Builder에 대응하는 프로젝트는 `Demos\CBuilder\<Name>\` 아래에 있습니다. 모든 데모 폴더에는 세부 사항을 담은 `readme.html`이 있습니다.

VCL 데모

이 데모들은 디자인 타임에 **TsgcChart**, **TsgcVCLChart**, 시리즈, 데이터 소스를 폼에 배치합니다. Delphi 7부터 Delphi 13까지, Win32와 Win64에서 빌드됩니다.

데모	보여 주는 내용	폴더
LiveLine	100만 점의 링 버퍼를 가진 **TsgcChartSeries_FastLine**이 **TsgcChartDataSource_Manual**을 통해 백그라운드 스레드에서 초당 1,000에서 200,000점씩 공급받으며, 10초/60초/전체 버퍼의 **TimeWindow**, 밝은/어두운 테마 전환, 그리고 백만 개의 점 위에서도 동작하는 크로스헤어와 툴팁을 보여 줍니다.	Delphi\LiveLine\
MultiAxis	세 개의 값 축(왼쪽, 오른쪽, 그리고 두 번째 왼쪽 축)에 세 개의 라인 시리즈, 날짜/시간 하단 축, 오른쪽의 범례를 보여 주며, 각 시리즈는 **AxisX**와 **AxisY**로 자신의 축을 선택하고, 모든 축에서 휠 확대/축소, 박스 확대/축소, 팬, 리셋을 지원합니다.	Delphi\MultiAxis\
Candlestick	실행 파일에 컴파일된 CSV의 1년치 일별 OHLCV 데이터를 **TsgcChartDataSource_CSV**로 불러와, 거래량을 두 번째 축의 막대로 표시하는 캔들스틱 시리즈에 로드합니다. 열 이름은 ISO 8601 날짜와 함께 X, 시가, 고가, 저가, 종가, 거래량에 매핑되며, 스위치로 캔들스틱과 OHLC 막대를 전환할 수 있습니다.	Delphi\Candlestick\
DataSetChart	1,000행을 가진 `TClientDataSet`을 `TDBGrid`에 표시하고, 두 개의 **TsgcChartDataSource_DataSet** 컴포넌트가 이를 읽어 라인 시리즈와 막대 시리즈에 넣습니다. **XField**와 **YField**가 데이터셋 필드를 지정하며, 그리드에서 행을 편집하고 다시 불러오면 차트가 갱신됩니다.	Delphi\DataSetChart\
Themes	동일한 차트를 밝은 테마와 어두운 테마로 나란히 보여 주며, 배율 선택기(1.0, 1.5, 2.0)와 **TsgcVCLChart.SaveToPNG** 및 **SaveToSVG**를 통한 PNG/SVG 내보내기를 제공합니다. 프로젝트는 `Themes\ChartThemes.dpr` 입니다.	Delphi\Themes\
PieDonut	**TsgcChartSeries_Pie**와 도넛 차트를 나란히 배치한 예제입니다. **TsgcChartDataSource_Table**을 통해 문자열 그리드에서 편집하는 작은 `IsgcUITable`에서 라이브로 데이터를 공급받으며, Others 임계값 슬라이더, 클릭 시 분리, 레이블 모드 전환, PNG와 SVG 내보내기를 갖추고 있습니다.	Delphi\PieDonut\
RadarCompare	**TsgcChartSeries_Radar**로 8개 기준에서 세 제품을 비교하는 예제입니다. 채우기 또는 윤곽선만, 다각형 또는 원형 그리드, 점수를 움직이는 타이머를 갖추고 있습니다.	Delphi\RadarCompare\
HeatmapLive	**Capacity** 300의 날짜/시간 축에 초당 30개의 라인을 추가하는 **TsgcChartSeries_Heatmap** 스펙트로그램과, 카테고리 축, 값 레이블, 연속 또는 단계형 색상 스케일로 `IsgcUITable`에서 로드하는 지역별 월별 행렬입니다.	Delphi\HeatmapLive\

TreemapFunnel	드릴다운과 이동 경로가 있는 3단계 **TsgcChartSeries_Treemap**과, 6단계 라이브 영업 깔때기인 **TsgcChartSeries_Funnel**입니다.	Delphi\TreemapFunnel\

sgcWebSockets 브리지 데모

이 데모들은 브리지 데이터 소스를 통해 sgcWebSockets 컴포넌트의 실시간 데이터를 차트에 공급합니다. sgcWebSockets 설치가 필요하며 Delphi 2010부터 Delphi 13까지, Win32와 Win64에서 빌드됩니다. sgcWebSockets 클라이언트, 서버, 데이터 소스는 디자인 타임에 폼에 배치됩니다.

데모	보여 주는 내용	폴더
MQTTSensors	세 개의 **TsgcChartDataSource_MQTT**가 하나의 `TsgcWSPClient_MQTT`를 공유하며 `sgccharts/sensors/temperature`, `humidity`, `pressure` 토픽을 차트로 표시합니다. 한 번에 `sgccharts/sensors/+`를 구독하며, WebSocket을 통한 test.mosquitto.org나 일반 TCP를 통한 로컬 브로커를 대상으로 합니다. Simulate 버튼은 100ms마다 임의의 값을 발행하므로 이 데모는 어떤 브로커에서도 동작하며, pressure는 자체 오른쪽 축을 가집니다.	Delphi\MQTTSensors\
BinanceLive	하나의 `TsgcWSAPI_Binance`에 두 개의 **TsgcChartDataSource_Binance**를 사용합니다: kline 스트림은 메시지마다 형성 중인 캔들을 갱신하는 캔들스틱 시리즈에 공급되고, 거래 스트림은 라인 시리즈에 공급됩니다. 구독 전에 Binance REST API를 통해 최근 60개 캔들을 불러오며, 연결된 상태에서도 심볼(BTCUSDT 또는 ETHUSDT)과 간격(1분 또는 5분)을 전환할 수 있습니다.	Delphi\BinanceLive\
WebSocketFeed	데모 안의 `TsgcWebSocketServer`가 20ms마다 `{"t":<unix ms>,"v":<value>}`를 브로드캐스트하고, `TsgcWebSocketClient`의 **TsgcChartDataSource_WebSocket**이 이를 표시하며, **XPath** `t`는 Unix 밀리초로, **YPath** `v`는 값으로 읽습니다. 서버와 클라이언트는 127.0.0.1의 같은 루프백 프로그램에서 실행됩니다.	Delphi\WebSocketFeed\
SSEFeed	데모 내부에서 제공되는 이벤트 스트림을 워커 스레드의 `TsgcHTTP1Client.GetSSE`로 읽고, **TsgcChartDataSource_SSE**가 이를 차트로 표시합니다. 이 소스는 원시 스트림을 이벤트로 분리하여 JSON 데이터를 읽으며, 그동안 폼 자체의 **OnSSEMessage**도 계속 실행되어 Stop에서 스트림을 취소합니다.	Delphi\SSEFeed\

FMX 데모

이 데모는 FireMonkey 폼에 TsgcFMXChart를 배치합니다. Delphi 12와 13으로, Win32, Win64, macOS(OSX64, OSXARM64), iOSDevice64, Android64를 대상으로 빌드됩니다.

데모	보여 주는 내용	폴더
FMXMobile	실시간 **TsgcChartSeries_FastLine**(백그라운드 스레드에서 초당 20,000점씩 공급되는 20만 점 링 버퍼)과 500개 캔들을 가진 **TsgcChartSeries_Candlestick**을 가진 탭 컨트롤이며, 터치에 맞게 크기가 조정됩니다. 시간 축에서 핀치 확대/축소와 팬, 크로스헤어와 툴팁, Reset 버튼, 10초/30초 **TimeWindow**, 밝은/어두운 테마 전환, 그리고 **TsgcFMXChart.FrameCount**로부터의 초당 프레임 수를 제공합니다.	Delphi\FMXMobile\

헤드리스 렌더링 데모

이 데모들은 VCL이나 FMX에 전혀 의존하지 않아 둘 다 설치할 필요가 없는 TsgcChartRenderer를 통해 시각적 컨트롤 없이 차트를 렌더링합니다. 콘솔 프로그램, Windows 서비스, ISAPI 모듈, Linux64 데몬에서 실행됩니다.

데모	보여 주는 내용	폴더
Headless	10,000점의 라인 시리즈와 500개 캔들의 캔들스틱 시리즈를 메모리에서 구성하고, 각각을 PNG와 SVG 파일(`line.png` , `line.svg` , `candles.png` , `candles.svg`)로 렌더링하며, 각 파일의 크기와 CRC32를 출력하는 콘솔 프로그램입니다. 데이터는 정수 연산으로 생성되고 래스터라이저는 플랫폼 간에 비트 단위로 동일하므로, Win64와 Linux64에서 작성된 PNG 파일은 같은 CRC32를 공유합니다. Win32, Win64, Linux64에서 실행됩니다.	Delphi\Headless\
HeadlessService	`HeadlessService.exe /install` 로 설치하고 `/uninstall` 로 제거하는 Windows 서비스로, 10초마다 라인 차트에 값을 추가하고 이를 `output\chart.png` 로 렌더링합니다. 서비스에는 데스크톱이 없으므로 차트는 시각적 컨트롤 대신 **TsgcChartRenderer**로 그려지며, 렌더링 오류는 Windows 이벤트 로그로 전달됩니다. Win32와 Win64에서 실행됩니다.	Delphi\HeadlessService\

참고 항목

- Quick Start
- Data Sources
- Bridge Data Sources
- TsgcChartRenderer

TsgcChart

Unit: sgcCharts_Chart
TsgcChart는 차트 모델입니다. 축, 시리즈, 제목, 부제목, 범례, 여백, 테마와 팔레트를 보유하는 `TComponent`의 하위 클래스이지만, 그 자체로는 아무것도 그리지 않으며 어떤 시각적 컨트롤에도 의존하지 않습니다. 뷰(VCL 또는 FireMonkey 컨트롤, 또는 헤드리스 렌더러)는 모델이 변경되었을 때 이를 알기 위해 **AddListener**를 호출하고, TsgcUICanvas에 그리기 위해 **Render**를 호출합니다. 모델과 드로잉이 분리되어 있으므로, 동일한 TsgcChart를 둘 이상의 뷰에서 동시에 표시하거나, 뷰 없이 곧바로 PNG 또는 SVG 파일로 렌더링할 수 있습니다.

속성

속성	형식	설명
Theme	TsgcUITheme	`utLight`, `utDark` 또는 `utCustom`. 기본값은 `utLight` 입니다. 테마와 팔레트를 참조하십시오.
Axes	TsgcChartAxes	축 컬렉션입니다. 축을 참조하십시오.
Title, Subtitle	TsgcChartTitle	제목, 범례, 여백을 참조하십시오.
Legend	TsgcChartLegend	제목, 범례, 여백을 참조하십시오.
Margins	TsgcChartMargins	제목, 범례, 여백을 참조하십시오.
BackgroundColor, PlotBackgroundColor	TsgcUIColor	**Theme**가 `utCustom` 일 때만 저장됩니다. 컴포넌트가 스트림에서 로드되는 중이 아닌 한, 둘 중 하나를 직접 설정하면 **Theme**가 `utCustom` 으로 전환됩니다.
Palette	TsgcChartPalette	8가지 자동 시리즈 색상입니다. 테마와 팔레트를 참조하십시오.
MaxFPS	Integer	뷰가 준수해야 하는 다시 그리기 상한입니다. 기본값은 60입니다.
TimeWindow	Double	초 단위이며 기본값은 0(꺼짐)입니다. 0이 아닌 경우, 자동 숫자형 또는 날짜/시간형 아래쪽 축은 전체 데이터 범위 대신 `[newest X - TimeWindow, newest X]` 를 표시합니다.
PlotRect	TsgcUIRect(읽기 전용)	마지막 **Render**로 계산된 플롯 사각형입니다.
Scale	Double	HiDPI 배율입니다. 뷰는 **Render**를 호출하기 전에 이 값을 설정합니다. 값을 변경해도 알림이 발생하지 않습니다.
SeriesCount	Integer(읽기 전용)	**Chart** 속성이 이 차트를 가리키는 시리즈의 개수입니다.
Series[Index: Integer]	TsgcChartSeries(읽기 전용, 기본 속성)	0부터 시작하는 Index 위치의 시리즈입니다.

메서드

메서드	반환 값	설명
BeginUpdate / EndUpdate		여러 변경 사항을 하나로 묶어, 변경할 때마다 알리는 대신 **EndUpdate** 시점에 한 번만 리스너에 알립니다.

Invalidate		리스너에게 즉시 알리거나, 업데이트가 진행 중이면 해당 **EndUpdate**까지 알림을 지연시킵니다.
AddListener(aEvent: TNotifyEvent)		차트를 다시 그려야 할 때 뷰가 콜백으로 호출할 이벤트를 등록합니다.
RemoveListener(aEvent: TNotifyEvent)		**AddListener**로 추가된 리스너를 등록 해제합니다.
Render(aCanvas: TsgcUICanvas; const aRect: TsgcUIRect)		**aCanvas**의 **aRect**(디바이스 픽셀) 안에 배경, 제목, 부제목, 범례, 축, 표시되는 모든 시리즈를 그립니다. Canvas를 참조하십시오.
FindAxis(const aName: string)	TsgcChartAxis	이 Name을 가진 축입니다(대소문자 구분 안 함). 없으면 nil입니다.
AxisRect(aAxis: TsgcChartAxis)	TsgcUIRect	마지막 **Render**에서 aAxis를 위해 플롯 바깥쪽에 예약된 대역입니다. 알 수 없는 경우 빈 사각형입니다.
AreaCount, AreaRect(aIndex), AreaMode(aIndex), AreaAt(X, Y)	Integer, TsgcUIRect, TsgcChartPlotMode, Integer	마지막 **Render**의 플롯 영역, 그 사각형과 모드, 그리고 픽셀 아래의 영역입니다. 플롯 영역을 참조하십시오.
SeriesAreaRect(aSeries), IsSeriesDrawn(aSeries)	TsgcUIRect, Boolean	마지막 **Render**가 시리즈를 그린 위치와 그렸는지 여부입니다.

모델과 뷰

TsgcChart는 자체 드로잉 표면을 갖지 않습니다. 컨트롤이 아니라 순수한 컴포넌트이므로, 폼에서와 마찬가지로 서비스나 콘솔 애플리케이션에서도 그대로 생성하고 데이터를 채우고 렌더링할 수 있습니다. 시리즈 자체도 **Chart** 속성을 가진 컴포넌트입니다(TsgcChartSeries 참조). 이 속성을 할당하면 시리즈가 차트에 등록되고, 지우거나(또는 시리즈를 파괴하면) 등록이 해제됩니다. 뷰가 모델로부터 필요로 하는 것은 단 세 가지입니다. 다시 그려야 할 때를 통보받기 위한 **AddListener**, 실제로 그리기 위한 **Render**, 그리고 미리 HiDPI 배율로 설정해 두는 **Scale**입니다.

업데이트와 알림

차트, 시리즈, 축에 대한 모든 변경은 **Invalidate**를 호출하며, 이는 등록된 리스너에게 즉시 알립니다. **BeginUpdate**와 **EndUpdate** 사이에 여러 변경 사항을 묶으면 해당 **EndUpdate**까지 알림이 지연되므로, 뷰는 변경할 때마다가 아니라 한 번만 다시 그립니다.

예제: 코드에서 차트 구성 및 렌더링

참고 항목

- TsgcChartSeries
- 축
- 플롯 영역
- Canvas

TsgcChartSeries

Unit: sgcCharts_Chart
TsgcChartSeries는 모든 시리즈의 추상 기반 클래스입니다. TsgcChart가 자신의 **Chart** 속성을 통해 표시하는 `TComponent`의 하위 클래스입니다. 모든 시리즈가 공유하는 색상, 표시 여부, 범례, 축 바인딩 속성을 정의하며, 구체적인 타입이 구현하는 두 개의 추상 멤버인 **GetDataRange**와 **Draw**를 갖습니다. 구체적인 시리즈 타입(라인, 영역, 스텝 라인, 막대, 산점도, 캔들스틱, OHLC 등)은 그 자체로 독립적인 컴포넌트이며, `sgcCharts_Series`에서 선언되고 별도로 문서화되어 있습니다.

속성

속성	형식	설명
Chart	TsgcChart	이 시리즈를 표시하는 차트입니다. 이 속성을 할당하면 시리즈가 차트에 등록됩니다. 지우거나 다른 차트로 설정하면 먼저 이전 차트에서 등록이 해제됩니다.
Title	string	범례에 표시되는 텍스트입니다. **LegendText**를 참조하십시오.
Color	TsgcUIColor	0(기본값)은 자동 팔레트 색상을 의미합니다. **EffectiveColor**를 참조하십시오.
Visible	Boolean	기본값은 True입니다.
ShowInLegend	Boolean	기본값은 True입니다.
AxisX, AxisY	string	이 시리즈가 그려지는 대상 축의 **Name**입니다. 축을 참조하십시오.
Decimation	TsgcChartDecimation	`dmNone`, `dmMinMax` 또는 `dmLTTB`. 기본값은 `dmMinMax` 입니다. 디시메이션을 참조하십시오.
Index	Integer(읽기 전용)	차트의 시리즈 목록에서 이 시리즈의 위치입니다. 차트에 바인딩되지 않은 경우 -1입니다.

메서드

메서드	반환 값	설명
Changed		**Chart**에 다시 그리도록 요청합니다(그 **Invalidate**를 호출함). **Chart**가 nil이면 아무 동작도 하지 않습니다.
EffectiveColor	TsgcUIColor	**Color**이며, **Color**가 0이면 **Index** 위치의 차트 팔레트 색상입니다.
XAxis	TsgcChartAxis	**AxisX**로 지정된 이름의 축이며, **Chart**에서 확인됩니다. **AxisX**가 비어 있으면 첫 번째 아래쪽 축이고, 찾을 수 없으면 nil입니다.
YAxis	TsgcChartAxis	**AxisY**로 지정된 이름의 축이며, **Chart**에서 확인됩니다. **AxisY**가 비어 있으면 첫 번째 왼쪽 축이고, 찾을 수 없으면 nil입니다.
GetDataRange(var aXMin, aXMax, aYMin, aYMax: Double)	Boolean(추상)	시리즈의 데이터 범위입니다. 데이터가 없으면 False입니다. 구체적인 시리즈 타입에서 구현됩니다.

Draw(aCanvas: TsgcUICanvas; aXAxis, aYAxis: TsgcChartAxis; const aPlot: TsgcUIRect; aScale: Double)		추상 메서드입니다. 이미 그 영역으로 클리핑된 **aPlot** 안에 시리즈를 그립니다. 값은 **aXAxis.ValueToPixel**과 **aYAxis.ValueToPixel**로 매핑됩니다.
DrawLegendSymbol(aCanvas: TsgcUICanvas; const aRect: TsgcUIRect; aScale: Double)		범례 기호를 그립니다. 기본값은 **EffectiveColor**로 채워진 사각형이며, 구체적인 타입에서 이를 재정의합니다.
LegendText	string	설정된 경우 **Title**이고, 그렇지 않으면 컴포넌트 **Name**입니다.
PlotMode	TsgcChartPlotMode(가상)	이 시리즈가 자신의 플롯 영역에 지정하는 모드입니다. `cpmXY` (기본값), `cpmPolar` 또는 `cpmCell` 입니다. 플롯 영역을 참조하십시오.
DrawArea(aCanvas; const aRect; const aSeries: TsgcChartSeriesArray; aCount: Integer; aScale: Double)	(가상)	극좌표 영역(영역의 첫 번째 레이더에서 그려진 모든 레이더를 **aSeries**에 담아 호출됩니다) 또는 셀(**aSeries**에는 이 시리즈만 있음)을 그립니다. 캔버스는 영역으로 잘립니다. 기본 구현은 아무것도 하지 않습니다.
HitTestArea(const aRect; const aSeries; aCount: Integer; X, Y: Double; var aHit: TsgcChartHitInfo)	Boolean(가상)	극좌표 영역 또는 셀의 히트 테스트이며, **DrawArea**와 같은 인수를 사용합니다. 기본 구현은 False를 반환합니다.
HitTestPoint(const aPlot: TsgcUIRect; X, Y: Double; var aHit: TsgcChartHitInfo)	Boolean(가상)	히트맵처럼 인터랙션이 알지 못하는 XY 시리즈의 히트 테스트입니다. 기본 구현은 False를 반환합니다.
TooltipText(const aHit: TsgcChartHitInfo)	string(가상)	이 시리즈의 히트에 대한 툴팁입니다. 빈 문자열(기본값)이면 인터랙션의 형식을 사용합니다. 여러 줄은 일반 줄로 그려집니다.
AreaClick(const aHit: TsgcChartHitInfo)	(가상)	이 시리즈의 히트에 대한 클릭이며 **OnSeriesClick** 뒤에 실행됩니다. 분리(원형 차트)와 드릴다운(트리맵)이 여기서 실행됩니다. 기본 구현은 아무것도 하지 않습니다.
LegendItemCount	Integer(가상)	시리즈의 범례 항목 수이며 기본값은 1입니다. 원형 차트는 슬라이스마다 하나를 반환합니다.
MeasureLegendItem(aCanvas; aItem: Integer; const aFont; aScale: Double; out aWidth, aHeight: Double)	(가상)	범례 항목 **aItem**의 크기입니다. 기본값은 심볼, 간격, 그리고 한 줄의 **LegendText**입니다.
DrawLegendItem(aCanvas; aItem: Integer; const aRect; const aFont; aScale: Double)	(가상)	범례 항목 **aItem**을 **aRect** 안에 그립니다. 기본 구현은 **DrawLegendSymbol**을 호출하고 **LegendText**를 그립니다.
GetDataCategories(aAxis: TsgcChartAxis; aList: TStrings)	Boolean(가상)	**Categories**가 없는 카테고리 축을 위한 런타임 카테고리 텍스트입니다. **aList**가 채워졌으면 True입니다. 히트맵은 이를 라인 및 셀 레이블에 사용합니다. 기본 구현은 False를 반환합니다.

차트 및 축에 바인딩하기

시리즈는 차트가 소유하지 않습니다. **Chart**를 설정하는 것은 단지 등록만 할 뿐이므로, 애플리케이션이 여전히 시리즈 인스턴스를 소유하고 해제합니다(또는 일반적인 Delphi 방식으로 소유자를 지정합니다). **AxisX**와 **AxisY**는 **Name**(대소문자 구분 안 함)으로 시리즈를 **Chart.Axes**의 축 중 하나에 바인딩합니다. 이름이 비어 있으면 X는 첫 번째 아래쪽 축, Y는 첫 번째 왼쪽 축으로 대체되므로, 각각 하나의 축만 있는 차트에서는 모든 시리즈에 이름을 지정할 필요가 없습니다.

색상과 팔레트

Color를 0으로 두면 시리즈는 자신의 **Index**에 대응하는 차트 **Palette** 색상을 사용하며, 8가지 팔레트 색상을 순환합니다. **Color**를 0이 아닌 값으로 설정하면 해당 시리즈에 대해서만 팔레트를 재정의합니다. 테마와 팔레트를 참조하십시오.

디시메이션

Decimation은 그리기 전에 구체적인 시리즈가 큰 버퍼를 픽셀 열당 대략 한 세트의 점으로 줄이는 방법을 제어합니다. 각 모드가 유지하는 내용은 디시메이션을 참조하십시오.

플롯 영역 후크

모든 후크의 기본 동작은 XY 동작이므로 **GetDataRange**와 **Draw**만 재정의하는 시리즈는 XY 시리즈로 남습니다. 원형, 레이더, 깔때기, 트리맵 시리즈는 **PlotMode**, **DrawArea**, **HitTestArea**를 재정의하며, 히트맵은 XY 시리즈로 남아 **HitTestPoint**와 **GetDataCategories**를 재정의합니다. 모드가 선택되는 방식은 플롯 영역을 참조하십시오.

예제: 이름이 지정된 축에 바인딩된 라인 시리즈

참고 항목

- TsgcChart
- 축
- 디시메이션
- 플롯 영역

TsgcChartAxis 및 TsgcChartAxes

Unit: sgcCharts_Axis
TsgcChartAxes는 TsgcChart.**Axes**로 노출되는 `TOwnedCollection`입니다. 각 항목은 **TsgcChartAxis**(`TCollectionItem`)입니다. 차트는 같은 쪽에 하나 이상의 축을 가질 수 있으며, 이는 차트가 두 개의 값 스케일을 동시에 표시하는 방법입니다. 시리즈는 이름을 통해 각각 하나의 축에 바인딩됩니다(TsgcChartSeries 참조).

TsgcChartAxis 속성

속성	형식	설명
Name	string	**AxisX** / **AxisY**를 통해 시리즈가 바인딩하는 식별자이며, **TsgcChart.FindAxis**가 조회하는 대상입니다.
AxisType	TsgcChartAxisType	`catNumeric`(기본값), `catLogarithmic`, `catDateTime` 또는 `catCategory`.
Position	TsgcChartAxisPosition	`capLeft`(기본값), `capRight`, `capTop` 또는 `capBottom`. 위쪽과 아래쪽 축은 수평(X)이고, 왼쪽과 오른쪽 축은 수직(Y)입니다. **IsHorizontal**을 참조하십시오.
Title, TitleStyle	string, TsgcUITextStyle	축 제목과 그 텍스트 스타일입니다.
LabelStyle	TsgcUITextStyle	눈금 레이블의 텍스트 스타일입니다.
Visible	Boolean	기본값은 True입니다.
Automatic	Boolean	기본값은 True입니다. True이면 범위는 이 축에 바인딩된 시리즈의 데이터로부터 결정됩니다. False이면 **Minimum** / **Maximum**이 지정된 대로 사용됩니다.
Minimum, Maximum	Double	**Automatic**이 False일 때 사용되는 수동 범위입니다.
Inverted	Boolean	기본값은 False입니다.
LabelFormat	string	비워 두면 자동 서식이 적용됩니다. `catNumeric` / `catLogarithmic`의 경우 로케일과 무관한 `FormatFloat` 마스크(`'0.00'`, `'#,##0.0'`)입니다. `catDateTime`의 경우 아래의 날짜/시간 형식 문자를 사용합니다.
GridVisible, GridColor, LineColor	Boolean, TsgcUIColor	그리드 선의 표시 여부와 그리드 및 축 선의 색상입니다. **GridVisible**의 기본값은 True입니다.
TickLength	Double	스케일 1에서의 디바이스 픽셀 단위 눈금 길이입니다.
MinTickSpacing	Double	두 눈금 사이의 스케일 1에서의 최소 디바이스 픽셀 간격입니다. 기본값은 50입니다.
Categories	TStrings	`catCategory` 축의 눈금 레이블로, 위치마다 하나씩 사용됩니다.

TsgcChartAxis 메서드

메서드	반환 값	설명
IsHorizontal	Boolean	**Position**이 `capTop` 또는 `capBottom` 일 때 True입니다.
SetDataRange(aMin, aMax: Double; aExact: Boolean = False)		자동 축이 값을 반올림하기 위해 확장해야 하는 데이터 범위를 보고합니다. **aExact**가 True이면 확장 없이 **[aMin, aMax]**를 그대로 사용합니다 (**TimeWindow**에 사용됨).
ClearDataRange		**SetDataRange**로 설정된 범위를 지웁니다.
SetDataCategories(aList: TStrings)		**Categories**가 없는 카테고리 축의 런타임 카테고리 텍스트입니다. 차트는 렌더링 중에 **SetDataRange**처럼 시리즈의 **GetDataCategories**에서 이를 설정합니다. 스트림에는 저장되지 않습니다.
ClearDataCategories		**SetDataCategories**가 설정한 텍스트를 지웁니다.
ZoomRange(aMin, aMax: Double)		표시 범위를 **[aMin, aMax]**로 제한합니다.
ResetZoom		**ZoomRange**로 설정된 범위를 지웁니다.
Zoomed	Boolean(읽기 전용)	확대/축소 범위가 활성 상태인 동안 True입니다.
VisibleMin, VisibleMax	Double(읽기 전용)	현재 적용 중인 범위입니다. 확대/축소 범위가 있으면 그 범위이고, 없으면 수동 범위 또는 데이터 범위입니다.
Calculate(aCanvas: TsgcUICanvas; aPixelLength: Double; aScale: Double = 1)		이 축을 따라 **aPixelLength** 픽셀 길이의 플롯에 대한 눈금을 계산합니다. **aCanvas**는 레이블을 솎아내기(thinning) 위해 레이블 크기를 측정하며 nil일 수 있습니다.
TickCount, TickValue[Index], TickLabel[Index]	Integer, Double, string	마지막 **Calculate** 호출로 계산된 눈금입니다. 솎아내기로 제거된 레이블의 경우 **TickLabel**은 비어 있습니다.
Step, DateTimeUnit	Double, TsgcChartDateTimeUnit	눈금 사이의 간격이며, `catDateTime` 축의 경우 그 간격이 반올림된 단위입니다.
ValueToPixel(aValue, aStart, aLength: Double)	Double	데이터 값을 디바이스 픽셀 위치로 매핑합니다.
PixelToValue(aPixel, aStart, aLength: Double)	Double	**ValueToPixel**의 역함수입니다.
MeasureSize(aCanvas: TsgcUICanvas; aScale: Double = 1)	Double	이 축이 플롯 바깥쪽에서 필요로 하는 두께(눈금, 레이블, 제목 포함)입니다. **Visible**이 아니면 0입니다.
Draw, DrawGrid		마지막 **Calculate**에 대해 축 또는 그 그리드 선을 그립니다.

TsgcChartAxes 멤버

멤버	설명
Add: TsgcChartAxis	새 축을 추가하고 이를 반환합니다.
Find(const aName: string): TsgcChartAxis	이 Name을 가진 첫 번째 축입니다(대소문자 구분 안 함). 없으면 nil입니다.
Items[Index: Integer]: TsgcChartAxis (기본 속성)	Index에 있는 축입니다.

자동 범위와 확대/축소

Automatic이 True인 동안, 차트는 매 렌더링마다 **SetDataRange**를 통해 이 축에 바인딩된 모든 시리즈의 결합된 데이터 범위를 축에 공급합니다. 자동 축은 차트가 **aExact**를 True로 하여 **SetDataRange**를 호출하지 않는 한 (**TsgcChart.TimeWindow**에 대해서는 그렇게 호출합니다) 눈금 값을 반올림하기 위해 그 범위를 확장합니다. **Automatic**을 False로 설정하고 **Minimum** / **Maximum**을 지정하면 대신 범위가 고정됩니다. **ZoomRange**와 **ResetZoom**은 대화형 확대/축소를 위해 둘 중 어느 쪽 위에도 적용될 수 있습니다. **VisibleMin** / **VisibleMax**는 실제로 적용 중인 범위를 보고합니다.

레이블 형식

`catDateTime` 축의 경우, **LabelFormat**은 런타임 로케일과 무관하게 로케일에 영향받지 않는 날짜/시간 문자를 사용합니다: `yyyy`, `yy`, `mmmm`, `mmm`, `mm`, `m`, `dd`, `d`, `hh`, `h`, `nn`, `n`, `ss`, `s`, `zzz`. 따옴표로 묶인 텍스트는 그대로 복사됩니다. `mm`은 항상 월을 나타내며, 분에는 `nn`을 사용하십시오. 비워 두면 축이 눈금 간격의 **DateTimeUnit**으로부터 자동으로 형식을 선택합니다.

Categories

`catCategory` 축은 숫자 눈금을 계산하는 대신 **Categories**의 각 항목마다 목록 순서대로 하나의 눈금을 배치합니다. 이 축에 바인딩된 시리즈는 해당 항목과 일치하는 정수 위치에 그려집니다. **Categories**가 비어 있으면 시리즈가 런타임에 텍스트를 제공할 수 있습니다. 히트맵은 이 방식으로 라인과 셀에 레이블을 붙입니다.

여러 개의 값 축

차트는 하나의 X축과 하나의 Y축으로 제한되지 않습니다. 기존 축과 같은 **Position**을 가진 두 번째 축을 추가하고 **AxisX** / **AxisY**를 통해 일부 시리즈를 그 축에 지정하면, 해당 시리즈는 자신만의 스케일을 갖게 되고 나머지는 첫 번째 축을 계속 사용합니다. **AxisX** 또는 **AxisY**를 비워 두는 시리즈는 각각 첫 번째 아래쪽 축과 첫 번째 왼쪽 축에 바인딩됩니다.

예제: 날짜/시간 아래쪽 축과 숫자 왼쪽 축

참고 항목

- TsgcChart
- TsgcChartSeries

제목, 범례, 여백

Unit: sgcCharts_Chart
세 개의 작은 `TPersistent` 클래스가 차트 주변 장식 요소를 담고 있습니다: **TsgcChartTitle**(TsgcChart.**Title**과 **Subtitle** 모두에 사용됨), **TsgcChartLegend**(**TsgcChart.Legend**), **TsgcChartMargins**(**TsgcChart.Margins**)입니다. 세 클래스 모두 차트를 무효화하는 내부 변경 알림을 발생시키므로, 런타임에 이들을 편집하면 자동으로 다시 그려집니다.

TsgcChartTitle 속성

속성	형식	설명
Text	string	제목 텍스트입니다. **Text**가 비어 있어도 공간을 차지하지 않습니다. 비어 있거나 **Visible**이 아니면 제목이 생략됩니다.
TextStyle	TsgcUITextStyle	글꼴 크기, 굵기, 색상입니다. 아래를 참조하십시오.
Visible	Boolean	기본값은 True입니다.
Alignment	TAlignment	기본값은 `taCenter` 입니다.

TsgcChartLegend 속성

속성	형식	설명
Visible	Boolean	기본값은 True입니다.
Position	TsgcChartLegendPosition	`clpTop` (기본값), `clpBottom`, `clpLeft` 또는 `clpRight`.
TextStyle	TsgcUITextStyle	범례 항목의 글꼴입니다.
BackgroundColor	TsgcUIColor	기본값은 0입니다(알파 0, 투명).
Padding	Integer	기본값은 4이며, 스케일 1에서의 디바이스 픽셀 단위입니다.

시리즈는 자신의 **ShowInLegend**가 True일 때 범례에 나타납니다(TsgcChartSeries 참조). 범례 텍스트는 시리즈의 **LegendText**입니다.

TsgcChartMargins 속성

속성	형식	설명
Left, Top, Right, Bottom	Integer	제목, 범례, 축 바깥쪽, 차트 전체를 둘러싼 스케일 1에서의 디바이스 픽셀 단위 외부 여백입니다. 기본값은 모든 방향에서 10입니다.

TsgcUITextStyle 속성

Unit: sgcCharts_Types. **TsgcChartTitle.TextStyle**, **TsgcChartLegend.TextStyle**, **TsgcChartAxis.TitleStyle** / **LabelStyle**에서 사용됩니다.

속성	형식	설명
Size	Double	포인트 단위의 글꼴 크기입니다. 기본값은 11입니다.
Bold	Boolean	기본값은 False입니다.
Color	TsgcUIColor	

ToFontStyle(aScale: Double = 1)은 캔버스 호출에 사용할 수 있는 대응 **TsgcUIFontStyle**을 반환하며, 이때 **Size**에 **aScale**이 곱해집니다.

범례 항목과 색상 막대

시리즈는 범례에 여러 항목을 추가할 수 있으며, 범례의 행은 가장 높은 항목의 높이를 따릅니다. 원형 차트는 그려진 슬라이스마다 항목 하나를 추가하며 Others가 마지막입니다. 깔때기는 그려진 단계마다 하나를, `ctcGroup` 모드의 트리맵은 최상위 그룹마다 하나를 추가합니다. 히트맵과 `ctcValue` 모드의 트리맵은 색상 스케일의 최솟값과 최댓값이 있는 색상 막대 하나를 추가하며, 범례 위치에 맞춰 가로 또는 세로입니다. **ShowInLegend**가 False이면 시리즈의 모든 항목이 숨겨지고, 플롯 영역이 건너뛴 시리즈는 항목을 표시하지 않습니다.

예제: 제목, 부제목, 범례 위치, 여백

참고 항목

- TsgcChart
- TsgcChartSeries

플롯 영역

단위: sgcCharts_Chart
플롯 영역은 차트의 플롯이거나, 차트에 패널이 있으면 보이는 패널 하나입니다. 렌더링할 때마다 차트는 영역에 바인딩된 보이는 시리즈로부터 계산한 모드를 각 영역에 지정합니다.

모드	시리즈	영역이 그리는 것
`cpmXY`	꺾은선, 영역, 막대, 분산형, 캔들스틱, 히트맵 및 기타 XY 시리즈	축, 그리드, 그리고 **Draw**를 통한 모든 XY 시리즈입니다.
`cpmPolar`	TsgcChartSeries_Radar	하나의 극좌표 그리드입니다. 첫 번째 레이더가 영역의 모든 레이더를 그 위에 그립니다.
`cpmCell`	TsgcChartSeries_Pie, TsgcChartSeries_Funnel, TsgcChartSeries_Treemap	시리즈마다 셀 하나가 있는 셀 그리드이며, 각 셀에는 제목 띠가 있습니다. 축은 없습니다.

혼합 규칙

영역에 XY 시리즈가 보이면 항상 XY가 우선합니다. 그렇지 않으면 처음 보이는 시리즈가 극좌표와 셀 중 무엇으로 할지 결정합니다. 영역의 모드와 맞지 않는 시리즈는 건너뜁니다. 그려지지 않고, 히트 테스트되지 않으며, 범례에도 표시되지 않습니다. 원형 차트를 꺾은선 차트 옆에 표시하려면 차트에 두 개의 패널을 두고 각 시리즈를 해당 패널의 축에 바인딩하거나, 두 개의 차트를 사용하십시오.

TsgcChart의 영역 멤버

메서드	반환 값	설명
AreaCount	Integer(읽기 전용)	마지막 **Render**의 플롯 영역입니다. 플롯에 대해 1개이거나, 패널이 있으면 보이는 패널마다 컬렉션 순서대로 1개입니다.
AreaRect(aIndex: Integer)	TsgcUIRect	영역의 사각형입니다. **aIndex**가 범위를 벗어나면 비어 있습니다.
AreaMode(aIndex: Integer)	TsgcChartPlotMode	영역의 모드입니다. **aIndex**가 범위를 벗어나면 `cpmXY` 입니다.
AreaAt(X, Y: Double)	Integer	차트 픽셀 아래에 있는 영역의 인덱스이며, 없으면 -1입니다.
SeriesAreaRect(aSeries: TsgcChartSeries)	TsgcUIRect	그려진 셀 시리즈의 셀, 그려진 극좌표 시리즈의 영역, XY 시리즈의 패널 플롯 사각형입니다. 다른 시리즈는 비어 있습니다.
IsSeriesDrawn(aSeries: TsgcChartSeries)	Boolean	마지막 **Render**가 시리즈를 그렸으면 True입니다. 시리즈가 보이고, 영역의 모드와 일치하며, XY 시리즈는 두 축을 모두 가진 경우입니다.

셀은 영역을 채우는 그리드로 배치됩니다. 셀이 여러 개이면 각 셀은 시리즈의 **Title**을 담은 제목 띠 아래에서 시작합니다. 인터랙션은 호버, 클릭, 툴팁을 모든 모드의 시리즈로 전달하지만, 확대/축소, 이동, 십자선은 XY 영역에서만 동작합니다. 시리즈 작성자는 TsgcChartSeries에 나열된 후크를 구현합니다.

예제: 두 개의 셀 시리즈와 그 뒤의 XY 시리즈

참고 항목

- TsgcChart
- TsgcChartSeries
- 패널
- TsgcChartInteraction

색상 스케일

단위: sgcCharts_ColorScale
TsgcChartColorScale은 두 개 또는 세 개의 색상 정지점으로 만든 256개 항목의 조회 테이블(LUT)을 통해 값을 색상에 대응시킵니다. TsgcChartSeries_Heatmap과 `ctcValue` 모드의 TsgcChartSeries_Treemap의 **ColorScale** 속성이며, 범례에는 최솟값과 최댓값이 있는 색상 막대로 표시됩니다.

속성

속성	형식	설명
MinColor	TsgcUIColor	최솟값의 색상입니다. 0(기본값)은 테마의 플롯 배경을 사용합니다.
MidColor	TsgcUIColor	범위 중간의 색상입니다. 0(기본값)은 정지점이 두 개라는 뜻입니다.
MaxColor	TsgcUIColor	최댓값의 색상입니다. 0(기본값)은 시리즈 색상을 사용하며, 없으면 첫 번째 팔레트 색상을 사용합니다.
Steps	Integer	0(기본값)은 연속 스케일이고, 2에서 32는 그 수만큼의 구간이 있는 단계형 팔레트입니다.
Automatic	Boolean	기본값은 True입니다. 범위는 보이는 셀 또는 타일의 최솟값과 최댓값입니다. False이면 **Minimum**과 **Maximum**을 사용합니다.
Minimum, Maximum	Double	고정 범위입니다. 기본값은 0과 100입니다.
NullColor	TsgcUIColor	NaN 셀의 색상입니다. 0(기본값)이면 칠하지 않습니다.
RangeMin, RangeMax	Double(읽기 전용)	마지막 렌더링에서 설정된 범위입니다.
LUT[aIndex]	TsgcUIColor(읽기 전용)	조회 테이블의 항목이며 0에서 255입니다.
OnChange	TNotifyEvent	속성이 변경될 때 발생합니다. 소유한 시리즈가 다시 그려집니다.

메서드

메서드	반환 값	설명
Prepare(aDefaultMinColor, aDefaultMaxColor: TsgcUIColor; const aDataMin, aDataMax: Double)		시리즈가 그리기 전에 호출합니다. 0인 색상을 확정하고, 정지점 또는 **Steps**가 변경되었으면 LUT를 다시 만들고, 범위를 설정합니다. 너비가 0인 범위는 모든 값을 항목 128에 대응시킵니다.
IndexOf(const aValue: Double)	Integer	값의 LUT 항목이며 0에서 255입니다. NaN이면 -1입니다.
ColorOf(const aValue: Double)	TsgcUIColor	값의 LUT 색상입니다. NaN이면 **NullColor**입니다.
DrawBar, MeasureColorBar, DrawColorBar		막대와 레이블이 있는 범례 색상 막대를 그립니다. 시리즈의 범례 항목이 사용합니다.

예제: 고정 범위의 단계형 신호등 스케일

참고 항목

- TsgcChartSeries_Heatmap
- TsgcChartSeries_Treemap
- 제목, 범례, 여백
- 테마와 팔레트

테마와 팔레트

Unit: sgcCharts_Chart (TsgcChartPalette), sgcCharts_Types (TsgcUITheme)
TsgcChart.**Theme**는 내장 배경, 텍스트, 그리드, 축 선 색상과 기본 시리즈 팔레트를 한 번에 선택합니다. **Palette**는 시리즈가 인덱스에 따라 자동으로 사용하는 8가지 색상 목록입니다.

TsgcUITheme

값	설명
utLight	기본값입니다. 밝은 배경, 어두운 텍스트, 밝은 그리드.
utDark	어두운 배경, 밝은 텍스트, 어두운 그리드.
utCustom	**BackgroundColor**, **PlotBackgroundColor**, **Palette**를 그대로 둡니다. 차트는 내장 색상을 적용하지 않습니다.

`utLight`와 `utDark`는 각각 자체 기본 **Palette**(시리즈 인덱스 순서로 사용되는 8가지 색상)를 가지고 있습니다. 차트가 스트림에서 로드하는 중이 아닌 한, **BackgroundColor**, **PlotBackgroundColor** 또는 임의의 **Palette** 색상을 직접 설정하면 **Theme**가 자동으로 `utCustom`으로 전환됩니다. 그 시점 이후 **Theme**를 다시 `utLight` 또는 `utDark`로 변경하면 해당 테마의 색상이 다시 적용되어 사용자 지정 색상을 덮어씁니다.

TsgcChartPalette 속성

속성	형식	설명
Colors[Index: Integer] (기본 속성)	TsgcUIColor	인덱스는 8의 나머지로 계산되므로 범위를 벗어난 시리즈 인덱스에 대해서도 예외가 발생하지 않습니다.
Color1 .. Color8	TsgcUIColor	공개 속성(published property)으로 제공되는 8가지 색상이며, 인덱스는 0부터 7까지입니다.

TsgcChartPalette 메서드

메서드	설명
LoadColors(const aColors: array of TsgcUIColor)	**aColors**에서 최대 8가지 색상을 순서대로 **Color1**..**Color8**에 복사합니다.

Color = 0(기본값)인 시리즈는 **EffectiveColor**를 통해 **Palette.Colors[Series.Index]**를 사용합니다. TsgcChartSeries를 참조하십시오.

예제: 다크 테마와 사용자 지정 팔레트

참고 항목

- TsgcChart
- TsgcChartSeries

TsgcChartSeries_Line, TsgcChartSeries_FastLine, TsgcChartSeries_StepLine

단위: sgcCharts_Series
이 세 시리즈는 모든 X/Y 시리즈(라인, 고속 라인, 계단 라인, 영역, 막대, 산점도)의 공통 기반인 **TsgcChartSeries_XY**에서 파생됩니다. 모든 시리즈가 공유하는 속성은 TsgcChartSeries를 참조하십시오. TsgcChartSeries_XY는 점 버퍼와 이를 채우는 메서드를 가지고 있습니다. **TsgcChartSeries_Line**은 선택적 대시 패턴과 점 마커를 가진 꺾은선을 그립니다. **TsgcChartSeries_FastLine**은 대량의 실시간 데이터를 위한 더 얇은 변형입니다. **TsgcChartSeries_StepLine**은 TsgcChartSeries_Line에서 파생되며 직선 구간 대신 계단형 선을 그립니다.

TsgcChartSeries_XY

TsgcChartSeries_XY는 X/Y 점의 **Buffer**를 가지고 있으며, 모든 라인, 영역, 계단 라인, 막대, 산점도 시리즈가 공유하는 데이터 메서드를 구현합니다. 추상 클래스는 아니지만 일반적으로 단독으로 사용되기보다는 하위 클래스 중 하나를 통해 사용됩니다.

속성	형식	설명
Buffer	TsgcChartBuffer(읽기 전용)	기반이 되는 링 버퍼입니다. TsgcChartBuffer and TsgcChartOHLCBuffer를 참조하십시오.
Capacity	Integer	버퍼가 보관하는 점의 최대 개수입니다. 기본값은 100000입니다.
BufferMode	TsgcChartBufferMode	`cbmOverwrite` (기본값)는 버퍼가 가득 차면 가장 오래된 점을 삭제합니다. `cbmGrow` 는 대신 용량을 두 배로 늘립니다.

메서드

메서드	반환 값	설명
AddXY(const aX, aY: Double)		점 하나를 추가하고 **Changed**를 호출합니다.
AddY(const aY: Double)		가장 최근 점의 X 값에 1을 더한 위치(버퍼가 비어 있으면 0)에 점을 추가하고 **Changed**를 호출합니다. X가 실제 좌표가 아니라 순차 인덱스일 때 유용합니다.
Clear		버퍼를 비우고 **Changed**를 호출합니다.

TsgcChartSeries_Line

TsgcChartSeries_Line은 버퍼의 점들을 꺾은선으로 그리며, 선택적 대시 패턴과 선택적 점 마커를 사용할 수 있습니다.

속성	형식	설명
LineWidth	Double	배율 1에서의 장치 픽셀 수입니다. 기본값은 2입니다.
Dash	TsgcChartDashStyle	`cdsSolid`, `cdsDash`, `cdsDot` 또는 `cdsDashDot` 입니다. 기본값은 `cdsSolid` 입니다.

Markers	TsgcChartSeriesMarkers	기본적으로 숨겨져 있습니다(**Visible**이 False). 아래의 TsgcChartSeriesMarkers를 참조하십시오.

TsgcChartSeries_FastLine

TsgcChartSeries_FastLine은 대량의 실시간 데이터를 위한 얇은 선입니다. 마커와 대시 패턴이 없으며, TsgcChartSeries_Line이 사용하는 둥근 연결 대신 베벨 연결로 구간을 잇기 때문에 높은 갱신 속도에서도 다시 그리는 비용이 저렴합니다.

속성	형식	설명
LineWidth	Double	배율 1에서의 장치 픽셀 수입니다. 기본값은 1입니다.

TsgcChartSeries_StepLine

TsgcChartSeries_StepLine은 TsgcChartSeries_Line에서 파생되므로 **LineWidth**, **Dash**, **Markers**도 가지고 있지만, 연속된 점들을 직선 대각선 대신 수평 및 수직 구간으로 연결합니다.

속성	형식	설명
StepMode	TsgcChartStepMode	`csmBefore`, `csmAfter` 또는 `csmCenter` 입니다. 기본값은 `csmAfter` 입니다.

csmAfter에서는 다음 점에 도달할 때까지 값이 유지되므로 각 점 바로 위치에서 계단이 발생합니다. **csmBefore**에서는 이전 점에서 값에 도달하므로 계단이 그 직전에 발생합니다. **csmCenter**에서는 두 점의 중간 지점에서 계단이 발생합니다.

TsgcChartSeriesMarkers

TsgcChartSeriesMarkers는 TsgcChartSeries_Line과 TsgcChartSeries_Scatter의 **Markers** 속성이 노출하는 작은 영속 객체입니다. 표시되는 모든 점의 중심에 도형 하나를 그립니다.

속성	형식	설명
Visible	Boolean	마커를 그릴지 여부입니다. 기본값은 소유 시리즈에 따라 다릅니다. TsgcChartSeries_Line과 그 하위 클래스에서는 False이고, TsgcChartSeries_Scatter에서는 True입니다.
Shape	TsgcChartMarkerShape	`cmsSquare`, `cmsCircle`, `cmsTriangle` 또는 `cmsDiamond` 입니다. 기본값은 `cmsCircle` 입니다.
Size	Integer	배율 1에서의 장치 픽셀 단위 지름입니다. 기본값은 6입니다.

예제: 하나의 차트에 표시된 라인, 고속 라인, 계단 라인 시리즈

참고 항목

- TsgcChart
- TsgcChartSeries
- TsgcChartSeries_Area
- TsgcChartSeries_Scatter

TsgcChartSeries_Area

Unit: sgcCharts_Series
TsgcChartSeries_Area는 TsgcChartSeries_Line에서 파생되므로 동일한 폴리라인을 그리고(**LineWidth**, **Dash**, **Markers** 포함), 추가로 그 선과 기준값 사이의 영역을 채웁니다.

속성

속성	형식	설명
Baseline	Double	채우기는 선에서 이 값까지 아래쪽으로(선이 이 값보다 아래에 있으면 위쪽으로) 진행됩니다. 기본값은 0입니다.
FillOpacity	Double	0..1. 기본값은 0.35입니다.
Gradient	Boolean	False(기본값)는 **FillOpacity**의 단색으로 채웁니다. True는 **FillOpacity**의 채우기 색상에서 시작하여 **Baseline**에서 완전히 투명해지는 수직 그라디언트로 채웁니다.

예제: 기준선을 올리고 그라디언트 채우기를 적용한 영역 시리즈

참고 항목

- TsgcChart
- TsgcChartSeries
- TsgcChartSeries_Line, TsgcChartSeries_FastLine, TsgcChartSeries_StepLine

TsgcChartSeries_Bar

단위: sgcCharts_Series
TsgcChartSeries_Bar는 **TsgcChartSeries_XY**에서 파생되며(**Buffer**, **AddXY**, **AddY**, **Clear**, **Capacity**, **BufferMode**에 대해서는 TsgcChartSeries_XY 항목 참조), 점마다 하나의 막대를 그립니다.

속성

속성	형식	설명
Decimation	TsgcChartDecimation	막대 시리즈에서는 기본값이 `dmNone` 이며, 기본 TsgcChartSeries의 기본값인 `dmMinMax` 와 다릅니다. 막대는 디시메이션으로 사라지지 않고 픽셀 열마다 하나 이상 표시되어야 하기 때문입니다. Decimation을 참조하십시오.
Orientation	TsgcChartBarOrientation	`cboVertical` (기본값) 또는 `cboHorizontal` 입니다. 수평 막대 시리즈는 X(위치)를 수직 축에, Y(값)를 수평 축에 매핑하므로 막대가 위쪽이 아니라 옆으로 자랍니다.
BarWidthPercent	Integer	막대 하나(또는 막대 그룹)가 X 슬롯에서 차지하는 비율이며, 1..100입니다. 기본값은 70입니다.
Grouping	TsgcChartBarGrouping	`cbgNone` (기본값), `cbgGrouped` 또는 `cbgStacked` 입니다. 아래를 참조하십시오.

그룹화 및 누적

cbgGrouped를 사용하면 동일한 축, 방향, 그룹화를 공유하는 표시된 막대 시리즈들이 각 X 슬롯 안에서 나란히 배치됩니다. **cbgStacked**를 사용하면 해당 시리즈들이 시리즈 순서대로 쌓이며, 양수 값은 양의 방향으로, 음수 값은 음의 방향으로 누적되므로 특정 X에서 음수 값을 가진 시리즈가 양의 누적을 밀어 올리지 않습니다. **cbgNone**은 모든 막대 시리즈를 각각 독립적으로 그리며, 동일한 X에서 서로 겹칩니다.

예제: 그룹화된 두 개의 막대 시리즈

참고 항목

- TsgcChart
- TsgcChartSeries
- Decimation

TsgcChartSeries_Scatter

단위: sgcCharts_Series
TsgcChartSeries_Scatter 는 **TsgcChartSeries_XY** 에 서 파 생 되 며(**Buffer**, **AddXY**, **AddY**, **Clear**, **Capacity**, **BufferMode**에 대해서는 TsgcChartSeries_XY 항목 참조), 연결선 없이 점을 마커로만 그립니다.

속성

속성	형식	설명
Decimation	TsgcChartDecimation	산점도 시리즈에서는 기본값이 `dmNone` 이며, 기본 TsgcChartSeries의 기본값인 `dmMinMax` 와 다릅니다. 산점도에는 시각적으로 정확하게 유지해야 할 선이 없으므로 모든 점이 유지됩니다. Decimation을 참조하십시오.
Markers	TsgcChartSeriesMarkers	기본적으로 표시됩니다(라인 시리즈의 Markers와 다름). **Shape**와 **Size**는 TsgcChartSeriesMarkers를 참조하십시오. **Visible**을 False로 설정하면 점이 숨겨집니다.

예제: 삼각형 마커를 사용하는 산점도 시리즈

참고 항목

- TsgcChart
- TsgcChartSeries
- TsgcChartSeries_Line, TsgcChartSeries_FastLine, TsgcChartSeries_StepLine
- Decimation

TsgcChartSeries_Candlestick, TsgcChartSeries_OHLC

단위: sgcCharts_Series
이 두 시리즈는 모든 open/high/low/close 시리즈의 공통 기반인 **TsgcChartSeries_OHLCBase**에서 파생됩니다. TsgcChartSeries_OHLCBase는 라인 계열 시리즈가 사용하는 X/Y 버퍼 대신 OHLC 버퍼를 가지며, 이를 채우는 메서드를 구현합니다. **TsgcChartSeries_Candlestick**은 각 막대를 심지가 있는 채워진 또는 속이 빈 캔들 몸통으로 그리고, **TsgcChartSeries_OHLC**는 각 막대를 시가와 종가 눈금이 있는 수직선으로 그립니다.

TsgcChartSeries_OHLCBase

속성	형식	설명
OHLCBuffer	TsgcChartOHLCBuffer(읽기 전용)	캔들의 기반이 되는 링 버퍼입니다. TsgcChartBuffer and TsgcChartOHLCBuffer를 참조하십시오.
Capacity	Integer	버퍼가 보관하는 캔들의 최대 개수입니다. 기본값은 100000입니다.
BufferMode	TsgcChartBufferMode	`cbmOverwrite` (기본값)는 버퍼가 가득 차면 가장 오래된 캔들을 삭제합니다. `cbmGrow` 는 대신 용량을 두 배로 늘립니다.
UpColor	TsgcUIColor	캔들의 종가가 시가보다 크거나 같을 때 사용하는 색상입니다. 기본값은 녹색인 $FF1BAF7A입니다.
DownColor	TsgcUIColor	캔들의 종가가 시가보다 낮을 때 사용하는 색상입니다. 기본값은 빨간색인 $FFE34948입니다.

메서드

메서드	반환값	설명
AddOHLC(const aX, aOpen, aHigh, aLow, aClose: Double; const aVolume: Double = 0)		캔들 하나를 추가하고 **Changed**를 호출합니다.
UpdateLastOHLC(const aX, aOpen, aHigh, aLow, aClose: Double; const aVolume: Double = 0)		X가 **aX**와 같으면 가장 최근 캔들을 교체하고, 그렇지 않으면 새 캔들을 추가한 뒤 **Changed**를 호출합니다. 주기가 끝날 때까지 진행 중인 캔들을 계속 갱신하는 실시간 피드에 사용하기 위한 것입니다.
Clear		버퍼를 비우고 **Changed**를 호출합니다.

TsgcChartSeries_Candlestick

TsgcChartSeries_Candlestick은 각 캔들을 시가와 종가 사이의 몸통으로 그리고, **UpColor** 또는 **DownColor**로 색칠하며, 고가에서 저가까지 이어지는 심지를 함께 그립니다.

속성	형식	설명

WickColor	TsgcUIColor	기본값은 0이며, 이는 심지가 캔들 몸통의 색상을 그대로 사용함을 의미합니다. 0이 아닌 값을 지정하면 모든 캔들에 대해 이를 재정의합니다.
BodyWidthPercent	Integer	캔들 슬롯에서 몸통이 차지하는 비율이며, 1..100입니다. 기본값은 70입니다.
HollowUp	Boolean	기본값은 False입니다. True로 설정하면 상승 캔들(종가 >= 시가)을 채워진 몸통 대신 **UpColor**의 윤곽선으로 그립니다.

TsgcChartSeries_OHLC

TsgcChartSeries_OHLC는 각 막대를 고가에서 저가까지 이어지는 수직선으로 그리고, 왼쪽에는 시가를 나타내는 짧은 눈금을, 오른쪽에는 종가를 나타내는 짧은 눈금을 표시하며, **UpColor** 또는 **DownColor**로 색칠합니다.

속성	형식	설명
TickWidthPercent	Integer	캔들 슬롯에서 시가/종가 눈금이 차지하는 비율이며, 1..100입니다. 기본값은 70입니다.
LineWidth	Double	배율 1에서의 장치 픽셀 수입니다. 기본값은 1입니다.

예제: 캔들스틱 시리즈와 OHLC 시리즈

참고 항목

- TsgcChart
- TsgcChartSeries
- Panes, 가격 축 아래의 별도 패널로 캔들스틱 시리즈를 분리하는 방법

TsgcChartSeries_Pie

단위: sgcCharts_Series_Pie
TsgcChartSeries_Pie는 원형 차트를 그리며, **InnerRadiusPercent**가 0보다 크면 도넛 차트를 그립니다. 이름이 있는 값을 담는 모든 시리즈의 기반인 **TsgcChartSeries_Category**(유닛 `sgcCharts_Series_Category`)에서 파생되며, 슬라이스마다 항목 하나를 **Slices** 컬렉션에 보관합니다. 이 컬렉션은 디자인 타임 편집기이자 라이브 데이터를 위한 스레드 안전 저장소이기도 합니다. 원형 차트는 셀 시리즈입니다. 플롯 영역에서 자신만의 셀을 차지하며 축을 그리지 않습니다. 플롯 영역을 참조하십시오.

속성

속성	형식	설명
Slices	TsgcChartCategoryItems	슬라이스이며, 각각 하나의 **TsgcChartPieSlice**(**Text**, **Value**, **Color**, **Exploded**)입니다. 슬라이스 색상은 컬렉션 인덱스를 따르므로 라이브 데이터에서도 슬라이스가 색상을 유지합니다.
InnerRadiusPercent	Double	0(기본값)은 원형 차트를 그리고, 90까지의 값은 구멍의 크기가 반지름의 이 비율인 도넛 링을 그립니다.
StartAngle	Double	첫 번째 슬라이스 가장자리의 각도(도)입니다. 0(기본값)은 12시 방향입니다.
Clockwise	Boolean	기본값은 True입니다.
Labels	TsgcChartSliceLabels	`cslNone` , `cslInside` , `cslOutside` (기본값) 또는 `cslAuto` 입니다. 아래의 레이블 항목을 참조하십시오.
LabelFormat	string	기본값은 `{text} ({percent})` 입니다. 토큰은 중괄호로 묶은 이름으로 작성합니다. `{text}` , `{value}` , `{percent}` 를 사용할 수 있습니다.
ValueFormat	string	값 토큰의 FormatFloat 마스크입니다. 비어 있으면(기본값) 소수점 이하 2자리까지 표시합니다.
PercentDecimals	Integer	퍼센트 토큰의 소수 자릿수입니다. 기본값은 1입니다.
LabelStyle	TsgcUITextStyle	레이블의 글꼴입니다. **Color**가 0이면 원 바깥에서는 테마 텍스트 색상을, 안쪽에서는 슬라이스 색상에 따라 선택한 흰색 또는 어두운 텍스트를 사용합니다.
LeaderLineColor	TsgcUIColor	바깥쪽 레이블 지시선의 색상입니다. 0(기본값)은 슬라이스 색상을 사용합니다.
BorderColor	TsgcUIColor	슬라이스 테두리의 색상입니다. 0(기본값)은 차트 배경색을 사용합니다.
BorderWidth	Double	슬라이스 테두리의 너비입니다. 기본값은 1입니다.
ExplodeOffset	Integer	분리된 슬라이스를 끌어내는 거리이며, 스케일 1에서의 장치 픽셀 단위입니다. 기본값은 10입니다.
ExplodeOnClick	Boolean	기본값은 False입니다. True이면 슬라이스를 클릭할 때 **Exploded** 플래그가 전환됩니다(Others 슬라이스는 **OthersExploded**).

OthersThreshold	Double	퍼센트 값입니다. 0(기본값)은 꺼짐입니다. 이 값보다 작은 슬라이스는 두 개 이상이 해당할 때 마지막에 그려지는 하나의 Others 슬라이스로 합쳐집니다.
OthersText	string	Others 슬라이스의 텍스트입니다. 기본값은 `Others` 입니다.
OthersColor	TsgcUIColor	Others 슬라이스의 색상입니다. 0(기본값)은 테마의 흐린 텍스트 색상을 사용합니다.
OthersExploded	Boolean(public)	Others 슬라이스를 원에서 끌어냅니다.
Total	Double(읽기 전용)	마지막 렌더링에서 그려진 값의 합계입니다.

메서드

메서드	반환 값	설명
SliceColor(aIndex: Integer)	TsgcUIColor	슬라이스 **aIndex**(컬렉션 인덱스 또는 **CS_CHART_OTHERS_INDEX**)를 그리는 색상입니다. 해당 슬라이스의 **Color**이며, 없으면 **aIndex**에 대한 차트 팔레트 색상입니다. Others 슬라이스는 **OthersColor**를 사용하며, 없으면 테마의 흐린 텍스트 색상을 사용합니다.

카테고리 시리즈 메서드

원형, 레이더, 깔때기, 트리맵 시리즈는 **TsgcChartSeries_Category**의 다음 메서드를 공유합니다. 이 메서드는 모두 어느 스레드에서든 호출할 수 있습니다. 저장소에는 자체 잠금이 있으며, 값을 바꾸지 않는 쓰기는 다시 그리기를 유발하지 않습니다.

메서드	반환 값	설명
SetValue(const aText: string; aValue: Double)	Integer	이 텍스트를 가진 첫 번째 항목의 값을 설정하거나(대소문자 구분), 새 항목을 추가합니다. 항목 인덱스를 반환합니다. 같은 텍스트에서는 최신 값이 우선하며, 값이 누적되지 않습니다.
SetValueAt(aIndex: Integer; aValue: Double)		**aIndex**의 항목 값을 설정합니다. 컬렉션이 더 짧으면 **aIndex**까지 빈 텍스트와 값 0의 항목을 추가합니다.
AddItem(const aText: string; aValue: Double; aColor: TsgcUIColor = 0)	Integer	텍스트가 이미 사용 중이더라도 항목을 추가합니다. 해당 인덱스를 반환합니다.
DeleteItem(aIndex: Integer)		항목 하나를 삭제합니다.
ClearItems		모든 항목을 삭제합니다.
IndexOfText(const aText: string)	Integer	이 텍스트를 가진 첫 번째 항목의 인덱스이며(대소문자 구분), 없으면 -1입니다.
ItemCount	Integer	항목 수입니다.
BeginUpdate / EndUpdate		이 사이의 쓰기는 마지막 **EndUpdate**에서 차트를 한 번만 다시 그립니다.
BeginReload / EndReload		마크 앤 스윕 방식입니다. 마지막 **EndReload**는 첫 번째 **BeginReload** 이후 기록되지 않은 항목을 삭제하므로, 다시 로드해도 남아 있는 항목의 색상과 플래그가 유지됩니다. 이것도 업데이트이며 마지막에 한 번만 다시 그립니다.

GetSnapshot(var aSnap: TsgcChartCategorySnapshot)		마지막 복사 이후 항목이 변경되었으면 잠금 상태에서 항목을 **aSnap**에 복사합니다.

모든 카테고리 시리즈의 레이블 토큰은 **sgcChartFormatLabel**이 확장합니다. 이 함수는 퍼센트와 이전 비율을 퍼센트 단위(25는 25%를 의미)로 받아 % 기호를 붙입니다.

Labels

cslInside는 각 레이블을 슬라이스 가운데에 배치합니다. **cslOutside**는 레이블을 원의 왼쪽과 오른쪽 두 열에 배치하고 지시선으로 슬라이스에 연결하며, 한 열의 레이블은 겹치지 않도록 간격이 조정됩니다. **cslAuto**는 레이블이 현과 링 두께에 들어맞으면 안쪽에, 그렇지 않으면 바깥쪽에 그립니다. 공간보다 너비가 큰 레이블은 맞도록 줄여집니다.

히트 테스트, 툴팁, 범례

슬라이스에 대한 히트는 **Index**와 **SubIndex**를 컬렉션 인덱스(Others 슬라이스는 **CS_CHART_OTHERS_INDEX**, 즉 -2)로, **YValue**를 값으로, **Percent**를 비율로, **Text**를 슬라이스 텍스트로 채웁니다. 툴팁은 `{text}: {value} ({percent})` 형식으로 표시됩니다. 범례에는 그려진 슬라이스마다 항목이 하나씩 표시되며 Others가 마지막입니다. TsgcChartInteraction을 참조하십시오.

예제: Others와 분리된 슬라이스가 있는 도넛 차트

참고 항목

- TsgcChartSeries
- 플롯 영역
- TsgcChartInteraction
- 제목, 범례, 여백

TsgcChartSeries_Radar

단위: sgcCharts_Series_Radar
TsgcChartSeries_Radar는 중심 주위의 스포크 위에 값을 다각형 선 또는 채워진 다각형으로 그립니다. 카테고리 시리즈입니다(원형 차트 항목의 카테고리 시리즈 메서드 참조). **Points**의 각 항목이 스포크 하나입니다. 레이더는 플롯 영역에 극좌표 모드를 지정하며, 한 영역의 모든 레이더는 하나의 극좌표 그리드를 공유합니다. 이 그리드의 스포크는 항목 텍스트를 처음 사용된 순서대로 합친 것입니다. 방사형 스케일은 시리즈의 **YAxis**이므로 해당 축의 **Automatic**, **Minimum**, **Maximum**, 눈금 수, 레이블 형식이 적용되며, 범위를 벗어난 값은 가장자리로 고정됩니다.

속성

속성	형식	설명
Points	TsgcChartCategoryItems	스포크마다 항목 하나입니다. **Text**는 스포크의 이름이고 **Value**는 방사형 값입니다.
Filled	Boolean	기본값은 False입니다. True이면 다각형을 채웁니다.
FillOpacity	Byte	채우기의 알파 값입니다. 기본값은 $40입니다.
LineWidth	Double	스케일 1에서의 장치 픽셀 단위 윤곽선 너비입니다. 0이면 윤곽선을 그리지 않습니다.
Markers	TsgcChartSeriesMarkers	꼭짓점 마커이며 기본적으로 표시됩니다.
GridShape	TsgcChartRadarGrid	`crgPolygon`(기본값) 또는 `crgCircle` 링입니다. 이 레이더가 영역에서 처음 그려지는 레이더일 때 사용됩니다.
StartAngle	Double	첫 번째 스포크의 각도(도)입니다. 0(기본값)은 12시 방향이며 시계 방향입니다. 이 레이더가 영역에서 처음 그려지는 레이더일 때 사용됩니다.
SpokeLabelStyle	TsgcUITextStyle	스포크 텍스트의 글꼴입니다. **Color**가 0이면 테마 텍스트 색상을 사용합니다. 이 레이더가 영역에서 처음 그려지는 레이더일 때 사용됩니다.

Y 축의 **GridVisible** 속성과 **GridColor** 속성이 링과 스포크를 제어하며, **Visible** 속성은 눈금 레이블을 켜고 끕니다. 가장자리에 눈금이 없으면 바깥쪽 링이 추가됩니다.

마지막 렌더링의 지오메트리

메서드	반환 값	설명
CenterX, CenterY, Radius	Double(읽기 전용)	극좌표 그리드의 중심과 반지름입니다.
SpokeCount	Integer(읽기 전용)	스포크 수입니다.
SpokeText[Index]	string(읽기 전용)	스포크의 텍스트입니다.
VertexPoint(aSpoke: Integer; out aX, aY: Double)	Boolean	스포크 **aSpoke**에서 이 시리즈의 꼭짓점 픽셀입니다. 스포크가 없거나 값이 유한하지 않거나 시리즈가 그려지지 않았으면 False(그리고 중심)입니다.

히트 테스트와 툴팁

히트는 마지막 렌더링의 지오메트리를 바탕으로, 클릭 반경 안에 있는 영역 레이더의 가장 가까운 꼭짓점을 반환합니다. **Index**는 스포크, **SubIndex**는 항목, **YValue**는 값, **Text**는 스포크 텍스트, **Distance**는 픽셀 거리입니다. **Percent**와 **ZValue**는 NaN입니다. 툴팁은 `Product A, Speed: 80` 형식으로 표시됩니다.

예제: 6개 기준으로 비교한 두 제품

참고 항목

- TsgcChartSeries
- 플롯 영역
- TsgcChartInteraction
- Axes

TsgcChartSeries_Heatmap

단위: sgcCharts_Series_Heatmap
TsgcChartSeries_Heatmap은 값의 행렬을 X 축과 Y 축 위의 색칠된 셀로 그립니다. XY 시리즈(**TsgcChartSeries**의 직접 하위 클래스)이므로 꺾은선, 막대, 분산형 시리즈와 플롯을 공유하며 숫자, 날짜/시간, 카테고리 축을 그대로 사용합니다. 값은 라인의 스레드 안전 저장소인 **Data**(**TsgcChartHeatmapData**)에 있습니다. **Orientation**이 `chmColumns`이면 라인은 X 키 위치의 열이고 그 셀은 Y를 따라 이어지며, `chmRows`이면 라인은 Y 키 위치의 행입니다. **Capacity**가 0보다 크면 저장소는 최신 라인만 보관합니다. 이는 링 버퍼 개념을 행렬에 적용한 것으로, 히트맵을 롤링 스펙트로그램으로 만듭니다. 색상은 색상 스케일에서 가져옵니다.

속성

속성	형식	설명
Orientation	TsgcChartHeatmapOrientation	`chmColumns` (기본값) 또는 `chmRows` 입니다.
Capacity	Integer	0(기본값)이면 저장소가 계속 늘어나고, 0보다 크면 최신 라인만 보관합니다.
CellOrigin	Double	셀 축에서 셀 0의 중심입니다. 기본값은 0입니다.
CellStep	Double	셀 중심 사이의 거리이며 0보다 커야 합니다. 기본값은 1입니다.
LineStep	Double	라인 축에서 라인이 차지하는 폭입니다. 0(기본값)이면 이웃 라인까지 거리의 절반에 이릅니다.
ColorScale	TsgcChartColorScale	셀 색상입니다. **MaxColor**가 0이면 시리즈의 **Color**가 되고, 그것도 없으면 첫 번째 팔레트 색상이 됩니다. **MinColor**가 0이면 플롯 배경이 됩니다. 색상 스케일을 참조하십시오.
ShowValues	Boolean	기본값은 False입니다. True이면 벡터 경로에서 칠해지는 모든 셀에 값을 씁니다.
ValueFormat	string	값 텍스트의 FormatFloat 마스크입니다. 비어 있으면(기본값) 소수점 이하 2자리까지 표시합니다.
ValueStyle	TsgcUITextStyle	값 텍스트의 글꼴입니다. **Color**가 0이면 셀의 휘도에 따라 흰색 또는 어두운 텍스트를 고릅니다.
CellBorderColor	TsgcUIColor	벡터 경로에서 칠해지는 모든 셀 안쪽에 그려지는 테두리입니다. 0(기본값)이면 그리지 않습니다.
Data	TsgcChartHeatmapData(읽기 전용)	라인 저장소입니다. **Data**에 직접 쓰면 차트가 다시 그려지지 않으며, 아래의 시리즈 메서드는 다시 그립니다.
Cells[aLine, aCell]	Double	셀 값입니다. 범위를 벗어나면 NaN입니다. 범위를 벗어난 쓰기는 무시됩니다. NaN은 null 셀이며 칠해지지도 히트되지도 않습니다.
ImagePath	Boolean(읽기 전용)	마지막 렌더링이 하나의 이미지를 채웠으면 True이고, 사각형을 그렸으면 False입니다.
DrawnCells	Integer(읽기 전용)	마지막 렌더링에서 보이는 셀입니다.

메서드

메서드	반환 값	설명
BeginUpdate / EndUpdate		이 사이의 쓰기는 차트를 한 번만 다시 그립니다. 어느 스레드에서든 호출할 수 있습니다.
Clear		모든 라인을 제거합니다.
SetSize(aLines, aCells: Integer)		행렬의 크기를 **aLines**개의 라인, 라인당 **aCells**개의 셀로 조정합니다.
AppendLine(const aValues: array of Double)		저장소의 실행 중인 라인 카운터를 키로 하여 라인을 추가합니다.
AppendLine(const aKey: Double; const aValues: array of Double)		X(열) 또는 Y(행) 키 **aKey** 위치에 라인을 추가합니다. 예를 들어 타임스탬프입니다. 현재 셀 수보다 긴 라인은 모든 라인을 넓히고 null 셀로 채웁니다.
SetLineLabels(const aLabels: array of string) / SetLineLabels(aLabels: TStrings)		라인의 텍스트이며, 카테고리 라인 축에 표시됩니다.
SetCellLabels(const aLabels: array of string) / SetCellLabels(aLabels: TStrings)		셀의 텍스트이며, 카테고리 셀 축에 표시됩니다.

축과 그리기

카테고리 축에서는 라인이나 셀이 키나 **CellOrigin**과 관계없이 논리 인덱스 위치에 놓이므로, 롤링 저장소가 레이블과 정렬된 상태를 유지하며 레이블은 **GetDataCategories**를 통해 축에 전달됩니다. 모든 셀 가장자리는 정수 장치 픽셀에 맞춰집니다. 시리즈는 캔버스가 아니라 보이는 셀의 수와 크기에 따라 그리기 경로를 선택합니다. 크기가 **CS_CHART_HEATMAP_VECTOR_MIN_SIZE**(3)픽셀 이상인 셀이 **CS_CHART_HEATMAP_VECTOR_CELLS**(20,000)개까지는 테두리와 값 텍스트가 있는 사각형으로 그려지고, 그보다 큰 행렬은 테두리나 텍스트 없이 하나의 이미지로 채워집니다. 출력은 래스터, SVG, VCL, FMX 캔버스에서 동일합니다.

히트 테스트, 툴팁, 범례

히트는 **Area**를 `chaPlot`으로, **Index**를 라인으로, **SubIndex**를 셀로, **XValue**와 **YValue**를 셀 중심으로, **ZValue**를 값으로, **Text**를 `line, cell`(레이블이 비어 있으면 축 텍스트)로 채웁니다. **Distance**는 0이고 **Percent**는 NaN입니다. 툴팁은 `Sales, Mar, South: 64` 형식으로 표시됩니다. 꺾은선 시리즈가 위에 그려져 있으면 클릭 반경 안의 꺾은선 점이 셀보다 우선합니다. 범례에는 스케일의 최솟값과 최댓값을 가진 색상 막대 하나가 표시됩니다.

예제: 지역별 월별 행렬

참고 항목

- TsgcChartSeries
- 플롯 영역
- TsgcChartInteraction
- 색상 스케일
- Axes

TsgcChartSeries_Treemap

단위: sgcCharts_Series_Treemap
TsgcChartSeries_Treemap은 계층 구조를 면적이 값에 비례하는 중첩 사각형으로 그립니다. 카테고리 시리즈입니다(원형 차트 항목의 카테고리 시리즈 메서드 참조). **Nodes**의 각 항목의 **Text**는 `Europe/Spain/Madrid` 같은 경로이며 **PathDelimiter**로 나뉘고, 그룹은 자식의 합계입니다. 레이아웃은 수준별로 squarified 방식이며 패딩과 헤더 띠가 있고, 라이브 값 업데이트 중에도 형제 순서를 유지합니다. 클릭하면 그룹으로 드릴다운하고, 타일 위의 이동 경로(breadcrumb)로 되돌아갑니다. 트리맵은 셀 시리즈입니다. 플롯 영역을 참조하십시오.

속성

속성	형식	설명
Nodes	TsgcChartCategoryItems	리프입니다. **Text**에 경로를, **Value**에 값을 지정합니다. 최상위 항목의 **Color**는 팔레트 색상보다 우선합니다.
PathDelimiter	Char	경로 세그먼트의 구분 기호입니다. 기본값은 `/` 입니다.
MaxDepth	Integer	현재 루트 아래에 그려지는 수준 수입니다. 1 미만은 제한이 없습니다. 기본값은 2입니다. 더 깊은 그룹은 접힌 상태로 하나의 타일로 그려집니다.
ColorMode	TsgcChartTreemapColorMode	`ctcGroup` (기본값)은 최상위 그룹마다 팔레트 색상을 쓰고, 수준마다 더 밝게(라이트 테마) 또는 더 어둡게(다크 테마) 합니다. `ctcValue` 는 루트 아래 타일의 범위에 대해 **ColorScale**로 타일을 칠합니다.
ColorScale	TsgcChartColorScale	`ctcValue` 의 색상입니다. **MaxColor**가 0이면 시리즈의 **Color**이고, 그것도 없으면 첫 번째 팔레트 색상입니다. 색상 스케일을 참조하십시오.
Levels	TsgcChartTreemapLevels	수준별 테두리, 패딩, 헤더입니다. **Items[0]**은 수준 1, 즉 현재 루트의 자식입니다. 아래의 TsgcChartTreemapLevel을 참조하십시오.
LabelFormat	string	기본값은 `{text}` 입니다. 토큰은 중괄호로 묶은 이름으로 작성합니다. `{text}` , `{value}` , `{percent}` 를 사용할 수 있습니다. text는 경로의 마지막 세그먼트이고 percent는 현재 루트에 대한 비율입니다.
ValueFormat	string	값 토큰의 FormatFloat 마스크입니다. 비어 있으면(기본값) 소수점 이하 2자리까지 표시합니다.
PercentDecimals	Integer	기본값은 1입니다.
LabelStyle	TsgcUITextStyle	레이블의 글꼴입니다. **Color**가 0이면 채우기 색상에 따라 흰색 또는 어두운 텍스트를 고릅니다.
StableLayout	Boolean	기본값은 True입니다. 값을 업데이트해도 형제 순서가 유지되므로 라이브 데이터에서 타일이 이리저리 튀지 않습니다.
DrillOnClick	Boolean	기본값은 True입니다. 타일을 클릭하면 그 수준 1 조상으로 드릴다운하고, 이동 경로 항목을 클릭하면 그곳으로 올라갑니다.
OnDrill	TNotifyEvent	**DrillPath**가 변경될 때 발생합니다.

DrillPath	string(public)	현재 루트의 경로입니다. 비어 있으면 트리 루트입니다. 설정하면 **OnDrill**이 발생합니다.
Layout	TsgcChartTreemapLayout(읽기 전용)	마지막 렌더링의 트리와 지오메트리입니다. 노드, 그려진 노드, 이동 경로(**Crumbs**)와 그 상자를 포함합니다.

TsgcChartTreemapLevel

속성	형식	설명
BorderColor	TsgcUIColor	0(기본값)은 차트 배경색을 사용합니다.
BorderWidth	Double	스케일 1에서의 장치 픽셀 단위입니다. 0이면 테두리를 그리지 않습니다.
Padding	Double	이 수준의 그룹과 그 자식 사이의 간격이며, 스케일 1에서의 장치 픽셀 단위입니다.
ShowHeader	Boolean	자식 위에 그룹 이름을 표시하는 헤더 띠입니다.

새 수준 항목은 해당 수준의 기본값을 갖습니다. 첫 번째 항목은 패딩 2픽셀과 헤더가 있고, 나머지는 1픽셀에 헤더가 없습니다. 항목이 없는 수준은 1픽셀 테두리를 사용합니다.

메서드

메서드	반환 값	설명
DrillDown(const aPath: string)		노드 경로를 현재 루트로 만듭니다. 빈 경로는 트리 루트입니다.
DrillUp		현재 루트의 부모를 현재 루트로 만듭니다.
DrillReset		트리 루트를 현재 루트로 만듭니다.
NodeColor(aNode: Integer)	TsgcUIColor	마지막 렌더링에서 레이아웃 노드 **aNode**의 채우기입니다. 그려지지 않았으면 0입니다.
CrumbRect(aIndex: Integer)	TsgcUIRect	마지막 렌더링에서 이동 경로 항목 **aIndex**(**Layout.Crumbs** 순서)의 상자입니다. 그려지지 않았으면 비어 있습니다. 첫 번째 항목은 루트이며 텍스트는 `All` 입니다.

레이블, 히트 테스트, 범례

레이블은 상자가 문자 3개 너비와 한 줄 높이 이상일 때 표시됩니다. 더 긴 텍스트는 맞도록 줄여지고, 더 작은 타일에는 레이블이 표시되지 않습니다. 히트는 지점 아래의 가장 깊은 타일을 반환합니다. **Index**는 항목(자체 항목이 없는 그룹은 -1), **SubIndex**는 깊이, **YValue**는 값, **Percent**는 현재 루트에 대한 비율, **Text**는 경로입니다. 이동 경로 항목에 대한 히트에서는 **Percent**가 NaN입니다. `ctcGroup`에서는 범례가 전체 트리의 최상위 그룹을 나열하고, `ctcValue`에서는 색상 막대 하나를 표시합니다.

예제: 드릴다운이 있는 3단계 트리

참고 항목

- TsgcChartSeries
- 플롯 영역
- TsgcChartInteraction

- 색상 스케일

TsgcChartSeries_Funnel

단위: sgcCharts_Series_Funnel
TsgcChartSeries_Funnel은 영업 파이프라인 같은 프로세스의 단계를 목 부분으로 좁아지는 깔때기로 그립니다. 카테고리 시리즈입니다(원형 차트 항목의 카테고리 시리즈 메서드 참조). **Stages**의 각 항목이 하나의 단계이며, 컬렉션 순서대로 위에서 아래로 배치됩니다. 깔때기는 셀 시리즈입니다. 플롯 영역을 참조하십시오.

속성

속성	형식	설명
Stages	TsgcChartCategoryItems	단계이며 위에서 아래 순서입니다.
NeckWidthPercent	Double	목의 너비이며 깔때기 너비에 대한 퍼센트입니다. 기본값은 30입니다.
NeckHeightPercent	Double	목의 높이이며 깔때기 높이에 대한 퍼센트입니다. 기본값은 25입니다.
ProportionalHeight	Boolean	False(기본값)는 모든 단계의 높이가 같습니다. True이면 값에 따라 높이가 정해집니다.
Gap	Integer	두 단계 사이의 간격이며, 스케일 1에서의 장치 픽셀 단위입니다. 기본값은 2입니다.
Labels	TsgcChartSliceLabels	`cslOutside`(기본값)는 오른쪽에 단계 색상의 지시선이 있는 열을 그립니다. `cslInside`, `cslAuto`, `cslNone`은 원형 차트와 같이 동작합니다.
LabelFormat	string	기본값은 `{text}: {value} ({percent})` 입니다. 토큰은 중괄호로 묶은 이름으로 작성합니다. `{text}`, `{value}`, `{percent}`(첫 번째 단계에 대한 비율), `{previous}`(바로 위 단계에 대한 비율이며 첫 번째 단계는 100%)를 사용할 수 있습니다.
ValueFormat	string	값 토큰의 FormatFloat 마스크입니다. 비어 있으면(기본값) 소수점 이하 2자리까지 표시합니다.
PercentDecimals	Integer	기본값은 1입니다.
LabelStyle	TsgcUITextStyle	**Color**가 0이면 깔때기 바깥에서는 테마 텍스트 색상을, 안쪽에서는 단계 색상에 따라 선택한 흰색 또는 어두운 텍스트를 사용합니다.
BorderColor	TsgcUIColor	0(기본값)은 차트 배경색을 사용합니다.
BorderWidth	Double	기본값은 1입니다.

마지막 렌더링의 메서드와 지오메트리

메서드	반환 값	설명
StageColor(aIndex: Integer)	TsgcUIColor	단계 **aIndex**를 그리는 색상입니다. 해당 단계의 **Color**이며, 없으면 **aIndex**에 대한 차트 팔레트 색상입니다.

HalfWidthAt(aY: Double)	Double	**aY**에서 도형의 절반 너비입니다. 깔때기 위나 아래에서는 0입니다.
StageAt(X, Y: Double)	Integer	**X, Y** 위치에 그려진 단계(또는 그 바깥쪽 레이블)의 위치이며, 없으면 -1입니다.
DrawnCount	Integer(읽기 전용)	그려진 단계 수입니다.
DrawnStages[aPos]	TsgcChartFunnelStage(읽기 전용)	그려진 단계이며 위에서 아래 순서입니다. **Index**(컬렉션 인덱스), **Value**, **Percent**(첫 번째 단계에 대한 비율), **Previous**(바로 위 단계에 대한 비율), **Top**, **Bottom**, 절반 너비, 레이블 위치와 상자, 지시선 점을 포함합니다.
CenterX, FunnelRect, NeckTop, NeckWidth	Double, TsgcUIRect(읽기 전용)	가로 중심, 도형의 경계 사각형, 목의 위쪽과 그 너비입니다.

히트 테스트, 툴팁, 범례

단계 또는 그 바깥쪽 레이블에 대한 히트는 **Index**와 **SubIndex**를 컬렉션 인덱스로, **YValue**를 값으로, **Percent**를 첫 번째 단계에 대한 비율로, **Text**를 단계 텍스트로 채웁니다. 툴팁은 `{text}: {value} ({percent})` 형식으로 표시됩니다. 범례에는 그려진 단계마다 항목이 하나씩 표시됩니다.

예제: 영업 파이프라인

참고 항목

- TsgcChartSeries
- 플롯 영역
- TsgcChartInteraction

TsgcVCLChart

단위: sgcCharts_VCL
TsgcVCLChart는 TsgcChart 모델의 VCL 뷰입니다. Windows 전용 `TCustomControl`의 하위 클래스로, 자신의 **Chart** 속성에 지정된 차트를 그립니다. 컨트롤은 차트에 리스너를 등록하고 모델이 변경될 때마다 스스로 다시 그립니다. 모델 자체는 그리기 코드를 전혀 가지고 있지 않으므로, 같은 차트를 동시에 TsgcFMXChart로 표시하거나 TsgcChartRenderer로 헤드리스 렌더링할 수도 있습니다.

속성

속성	형식	설명
Chart	TsgcChart	이 컨트롤이 그리는 차트 모델입니다. 이 값을 지정하면 컨트롤이 리스너로 등록됩니다. 값을 지우거나 차트를 파괴하면 컨트롤이 더 이상 아무것도 그리지 않습니다.
FrameCount	Cardinal(읽기 전용)	컨트롤이 생성된 이후 렌더링된 전체 프레임 수입니다.
OverlayFrameCount	Cardinal(읽기 전용)	컨트롤이 생성된 이후 렌더링된 오버레이 전용 프레임 수입니다. 아래의 **Overlay-only repaints**를 참조하십시오.

일반적인 `TCustomControl` 속성들도 변경 없이 공개됩니다: **Align**, **Anchors**, **Color**, **Constraints**, **DragCursor**, **DragKind**, **DragMode**, **Enabled**, **ParentColor**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **TabOrder**, **TabStop**, **Visible**, 그리고 마우스, 드래그, 크기 조정 이벤트입니다.

메서드

메서드	반환 값	설명
SaveToPNG(const aFileName: string)		컨트롤의 현재 클라이언트 크기와 PPI로 **Chart**를 렌더링하여 새 PNG 파일로 저장합니다. **Chart**가 nil이면 아무 동작도 하지 않습니다.
SaveToSVG(const aFileName: string)		컨트롤의 현재 클라이언트 크기와 PPI로 **Chart**를 렌더링하여 SVG 파일로 저장합니다. **Chart**가 nil이면 아무 동작도 하지 않습니다.

MaxFPS 제한

MaxFPS는 이 컨트롤의 속성이 아니라 차트 모델에 속합니다(TsgcChart 참조). 차트의 변경 알림은 컨트롤을 더티로 표시하고 다시 그리기를 예약할 뿐이며, 초당 최대 **Chart.MaxFPS** 회로 제한됩니다. 다음 프레임이 이미 도래한 경우 컨트롤은 즉시 무효화되고, 그렇지 않으면 프레임이 도래할 때 일회성 타이머가 무효화합니다. 프레임은 마지막 그리기로부터의 고정 간격이 아니라, 진행 중인 마감 시각(이전 마감 시각에 1000을 **MaxFPS**로 나눈 값을 더한 값)을 기준으로 배분되므로, Windows 타이머의 정밀도로 인해 실제 속도가 상한 아래로 떨어지지 않습니다. 피드 스레드에서 도착한 알림은 컨트롤을 직접 건드리지 않고, 메인 스레드가 다시 그리기를 예약하기 전에 처리하는 전용 메시지를 게시합니다.

오버레이 전용 다시 그리기

컨트롤은 차트에 두 번째 오버레이 리스너도 등록합니다. 크로스헤어, 툴팁 또는 러버 밴드 선택만 이동시키는 변경(**TsgcChart.InvalidateOverlay**)은 전체 다시 그리기를 실행하지 않습니다. 축, 시리즈, 배경, 제목, 범례는 다시 그려지지 않습니다. 대신 마지막 전체 렌더링의 픽셀이 내부 래스터로 다시 복사되고 **TsgcChart.RenderOverlay**만 그 위에 그려집니다. 이 때문에 **OverlayFrameCount**와 **FrameCount**가 별도로 집계됩니다. 크기 조정, PPI 변경, 차트의 데이터/축/테마/레이아웃 변경 등 다른 이유로 컨트롤이 더티가 될 때는 여전히 전체 렌더링이 일어납니다.

예제: VCL 폼에 차트 그리기

참고 항목

- TsgcChart
- TsgcFMXChart
- TsgcChartRenderer
- TsgcChartSeries_Line
- TsgcChartInteraction
- Panes

TsgcFMXChart

단위: sgcCharts_FMX
TsgcFMXChart는 TsgcChart 모델의 FireMonkey 뷰입니다. Windows, macOS, iOS, Android에서 자신의 **Chart** 속성에 지정된 차트를 그리는 `TControl`의 하위 클래스입니다. 이 유닛은 `SGC_CHARTS_FMX`가 정의되지 않으면 아무것도 컴파일되지 않으며, 이는 Delphi XE6 이상을 필요로 합니다. 모델 자체는 그리기 코드를 전혀 가지고 있지 않으므로, 같은 차트를 동시에 TsgcVCLChart로 표시하거나 TsgcChartRenderer로 헤드리스 렌더링할 수도 있습니다.

속성

속성	형식	설명
Chart	TsgcChart	이 컨트롤이 그리는 차트 모델입니다. 이 값을 지정하면 컨트롤이 리스너로 등록됩니다. 값을 지우거나 차트를 파괴하면 컨트롤이 더 이상 아무것도 그리지 않습니다.
FrameCount	Cardinal(읽기 전용)	컨트롤이 생성된 이후 렌더링된 전체 프레임 수입니다.
OverlayFrameCount	Cardinal(읽기 전용)	컨트롤이 생성된 이후 렌더링된 오버레이 전용 프레임 수입니다. 아래의 **Overlay-only repaints**를 참조하십시오.

일반적인 `TControl` 속성들도 변경 없이 공개됩니다: **Align**, **Anchors**, **ClipChildren**, **ClipParent**, **Cursor**, **DragMode**, **EnableDragHighlight**, **Enabled**, **Locked**, **Height**, **HitTest**, **Padding**, **Opacity**, **Margins**, **PopupMenu**, **Position**, **RotationAngle**, **RotationCenter**, **Scale**, **Visible**, **Width**, **TouchTargetExpansion**, 그리고 마우스, 드래그, 크기 조정 이벤트입니다.

메서드

메서드	반환값	설명
SaveToPNG(const aFileName: string)		컨트롤의 현재 크기와 씬 배율로 **Chart**를 렌더링하여 새 PNG 파일로 저장합니다. **Chart**가 nil이면 아무 동작도 하지 않습니다.
SaveToSVG(const aFileName: string)		컨트롤의 현재 크기와 씬 배율로 **Chart**를 렌더링하여 SVG 파일로 저장합니다. **Chart**가 nil이면 아무 동작도 하지 않습니다.
DetectPixelOrder(var aData: TBitmapData) [class function]	TsgcFMXChartPixelOrder	매핑된 비트맵 데이터에 테스트 픽셀을 기록한 뒤 다시 읽어, 플랫폼 비트맵이 `fpoBGRA`, `fpoRGBA`, 또는 그 밖의 (`fpoOther`) 채널 순서 중 무엇을 사용하는지 알아냅니다. 렌더링된 픽셀을 FMX 비트맵에 복사하기 전에 내부적으로 사용되며, 대부분의 애플리케이션은 이를 직접 호출하지 않습니다.
CopyToBitmapData(aRasterizer: TsgcUIRasterizer; var aData: TBitmapData) [class procedure]		**aRasterizer**의 직선 ARGB 픽셀을, 두 크기 중 더 작은 쪽에 맞춰 매핑된 비트맵 데이터의 사전 곱셈된 픽셀로 변환합니다. **Paint**에서 내부적으로 사용되며, 대부분의 애플리케이션은 이를 직접 호출하지 않습니다.

MaxFPS 제한

MaxFPS는 이 컨트롤의 속성이 아니라 차트 모델에 속합니다(TsgcChart 참조). 차트의 변경 알림은 컨트롤을 더티로 표시하고 다시 그리기를 예약할 뿐이며, 초당 최대 **Chart.MaxFPS** 회로 제한됩니다. 다음 프레임이 이미 도래한 경우 컨트롤은 즉시 다시 그려지고, 그렇지 않으면 프레임이 도래할 때 일회성 타이머가 다시 그립니다. 프레임은 마지막 그리기로부터의 고정 간격이 아니라, 진행 중인 마감 시각(이전 마감 시각에 1000을 **MaxFPS**로 나눈 값을 더한 값)을 기준으로 배분됩니다. 피드 스레드에서 도착한 알림은 컨트롤을 직접 건드리지 않고, **TThread.Queue**로 메인 스레드에 호출을 큐에 넣습니다(대기 중인 호출은 최대 하나). 메인 스레드가 다시 그리기를 예약하며, 소멸자는 아직 큐에 남아 있는 호출을 제거합니다.

오버레이 전용 다시 그리기

컨트롤은 차트에 두 번째 오버레이 리스너도 등록합니다. 크로스헤어, 툴팁 또는 러버 밴드 선택만 이동시키는 변경(**TsgcChart.InvalidateOverlay**)은 전체 다시 그리기를 실행하지 않습니다. 축, 시리즈, 배경, 제목, 범례는 다시 그려지지 않습니다. 대신 마지막 전체 렌더링의 픽셀이 내부 래스터로 다시 복사되고 **TsgcChart.RenderOverlay**만 그 위에 그려집니다. 이 때문에 **OverlayFrameCount**와 **FrameCount**가 별도로 집계됩니다. 크기 조정, 씬 배율 변경, 차트의 데이터/축/테마/레이아웃 변경 등 다른 이유로 컨트롤이 더티가 될 때는 여전히 전체 렌더링이 일어납니다.

예제: FMX 폼에 차트 그리기

참고 항목

- TsgcChart
- TsgcVCLChart
- TsgcChartRenderer
- TsgcChartSeries_Line
- TsgcChartInteraction
- Panes

TsgcChartRenderer

단위: sgcCharts_Renderer
TsgcChartRenderer는 TsgcChart 모델의 헤드리스 뷰입니다. VCL이나 FMX에 의존하지 않고 자체 창도 없는 단순한 `TComponent`로서, 콘솔 프로그램, Windows 서비스, ISAPI 모듈 또는 Linux64 데몬에서 차트를 PNG 파일이나 스트림, SVG 문서 또는 원시 ARGB 픽셀 버퍼로 렌더링합니다. 시각적 표면이 없으므로 **Chart.MaxFPS**의 영향을 받지 않습니다. 프레임 루프나 타이머가 없으며, 각 렌더링 호출은 호출되는 즉시 동기적으로 차트를 그립니다.

속성

속성	형식	설명
Chart	TsgcChart	이 렌더러가 그리는 차트 모델입니다. 이 값이 nil이면 모든 Render* 메서드가 **EsgcChartRendererError**를 발생시킵니다.
Width	Integer	크기를 지정하지 않는 오버로드에서 사용되는 기본 논리 너비(픽셀)입니다. 기본값은 800입니다.
Height	Integer	크기를 지정하지 않는 오버로드에서 사용되는 기본 논리 높이(픽셀)입니다. 기본값은 500입니다.
Scale	Double	논리 픽셀당 장치 픽셀 수입니다. 기본값은 1입니다. 출력 크기는 `Round(Width * Scale)` x `Round(Height * Scale)` 장치 픽셀이 됩니다. **Chart.Scale**은 렌더링 동안 이 값으로 설정되었다가 이후 이전 값으로 복원됩니다.
CompressPNG	Boolean	PNG 출력을 deflate로 압축할지 여부입니다. 라이브러리가 `SGC_CHARTS_DEFLATE` 로 빌드된 경우 기본값은 True이고, 그렇지 않으면 기본값은 False입니다.

메서드

메서드	반환 값	설명
RenderToPNGFile(const aFileName: string) RenderToPNGFile(const aFileName: string; aWidth, aHeight: Integer)		**Width/Height** 크기로, 또는 **Width**와 **Height**를 변경하지 않고 지정된 논리 크기로 PNG 파일에 렌더링합니다.
RenderToPNGStream(aStream: TStream) RenderToPNGStream(aStream: TStream; aWidth, aHeight: Integer)		PNG(RGBA, 8비트)를 스트림에 렌더링합니다.
RenderToSVG RenderToSVG(aWidth, aHeight: Integer)	string	렌더링하여 SVG 1.1 문서(순수 ASCII)를 문자열로 반환합니다.
RenderToSVGFile(const aFileName: string) RenderToSVGFile(const aFileName: string; aWidth, aHeight: Integer)		SVG를 파일에 렌더링합니다.
RenderToSVGStream(aStream: TStream) RenderToSVGStream(aStream: TStream; aWidth, aHeight: Integer)		SVG를 스트림에 렌더링합니다.
RenderToBuffer(var aBuffer: TsgcUICardinalArray; out aPixelWidth,		직선(사전 곱셈되지 않은) $AARRGGBB 픽셀로, 위에서 아래 순서로, **aPixelWidth**와 같은 행 스트라이드

aPixelHeight: Integer) RenderToBuffer(var aBuffer: TsgcUICardinalArray; out aPixelWidth, aPixelHeight: Integer; aWidth, aHeight: Integer)		로 렌더링합니다. **aPixelWidth**와 **aPixelHeight**는 장치 크기로 반환됩니다.

스레딩

모든 렌더링 메서드는 어느 스레드에서나 호출할 수 있습니다. 각 **TsgcChartRenderer**는 래스터용과 SVG용 각각 하나씩 자체 캔버스를 가지며, 이는 해당 렌더러 인스턴스 전용 잠금으로 보호됩니다. 이에 더해, **TsgcChart.Render**가 실행되는 동안 레이아웃 상태를 차트 자체에 보관하기 때문에, 프로세스 내 모든 **TsgcChartRenderer**의 모든 렌더링은 유닛 전체가 공유하는 하나의 추가 잠금을 거칩니다. 렌더링이 진행되는 동안에는 차트를 구조적으로 변경해서는 안 됩니다. 즉 시리즈나 축을 추가, 제거, 해제하거나 레이아웃에 영향을 주는 속성을 변경해서는 안 됩니다. 반면 시리즈 버퍼(**Buffer.Append**, **OHLCBuffer.Append**, 데이터 소스)를 통해 데이터를 추가하는 것은 안전합니다. 시리즈는 버퍼 자체의 잠금 하에 만들어진 스냅숏에서 그려지기 때문입니다.
공유 렌더링 잠금은 **TsgcChartRenderer**만 사용합니다. TsgcVCLChart나 TsgcFMXChart 같은 시각적 뷰는 이 잠금을 사용하지 않으므로, 시각적 뷰가 같은 차트를 그리고 있는 순간에 다른 스레드에서 렌더러로 같은 차트를 렌더링하지 마십시오.

예제: PNG와 SVG로의 헤드리스 렌더링

참고 항목

- TsgcChart
- TsgcVCLChart
- TsgcFMXChart
- TsgcChartSeries_Line
- Panes

TsgcChartInteraction

단위: sgcCharts_Interaction
TsgcChartInteraction은 TsgcChart의 입력 API입니다. 프레임워크에 의존하지 않는 `TPersistent`로서, 원시 마우스, 휠, 제스처 호출을 크로스헤어 판독, 점 클릭, 확대/축소, 팬, 그리고 재무 에디션에서는 그리기 도구와 패널 분할선 동작으로 변환합니다. VCL, FMX나 어떤 윈도잉 API에도 의존하지 않습니다. 뷰(TsgcVCLChart, TsgcFMXChart, 또는 사용자 정의 호스트 루프)는 자신의 입력 이벤트에서 받은 것을 차트 픽셀 좌표로, 즉 **TsgcChart.Render**에 주어지는 사각형과 동일한 장치 픽셀 공간으로 전달합니다. 모든 차트는 정확히 하나의 인터랙션 인스턴스를 소유하며, 이는 생성자에서 만들어져 **TsgcChart.Interaction**으로 노출됩니다. 애플리케이션은 자체적으로 인스턴스를 만드는 대신 그곳에서 설정합니다.

속성

속성	형식	설명
Crosshair	Boolean	플롯 위에 마우스를 올리는 동안 크로스헤어와 축 판독값을 그립니다. 기본값은 True입니다.
CrosshairColor	TsgcUIColor	0(기본값)은 테마 색상을 사용합니다.
Tooltip	Boolean	마우스를 올리는 동안 가장 가까운 점의 툴팁을 표시합니다. 기본값은 True입니다.
ZoomMode	TsgcChartZoomMode	`czmNone`, `czmX`, `czmY` 또는 `czmXY` 입니다. 확대/축소와 팬이 작용하는 축을 결정합니다. 기본값은 `czmX` 입니다.
WheelZoom	Boolean	휠은 커서 주위로 확대/축소하고, Shift + 휠은 대신 수평으로 팬합니다. 기본값은 True입니다.
BoxZoom	Boolean	왼쪽 버튼 드래그가 러버 밴드를 그려 그 영역으로 확대합니다. 기본값은 True입니다. Ctrl + 왼쪽 드래그는 항상 대신 팬하며, 오른쪽이나 가운데 버튼 드래그는 이 속성과 무관하게 팬합니다.
Pan	Boolean	드래그에 의한 팬을 활성화합니다. 기본값은 True입니다.
AutoScrollResumeDelay	Integer	수평 확대/축소가 리셋되고 **TsgcChart.TimeWindow**가 다시 스크롤되기까지 상호작용이 없어야 하는 밀리초 수입니다. 0이면 자동으로 재개되지 않습니다. 기본값은 3000입니다. Auto-scroll을 참조하십시오.
SnapToPoint	Boolean	크로스헤어가 가장 가까운 점으로 이동합니다: 먼저 X에서 가장 가까운 점을, 그다음 그 X에 있는 시리즈들 중 Y에서 가장 가까운 점을 찾습니다(False이면 픽셀 거리상 가장 가까운 점). 기본값은 True입니다.
DrawingTool	TsgcChartDrawingTool	재무 에디션 전용입니다. `cdtNone` (기본값)에서는 왼쪽 버튼 누름으로 기존 드로잉을 선택하고 드래그할 수 있습니다. 그 외의 값(`cdtTrendLine`, `cdtHorizontalLine`, `cdtVerticalLine`, `cdtRectangle`, `cdtFibonacci`)은 플롯 안에서의 다음 왼쪽 드래그가 그 종류의 드로잉을 만들도록 합니다(누름 = 첫 번째 기준점, 뗌 = 두 번째 기준점). 이후 속성은 다시 `cdtNone` 으로 돌아갑니다.
Paused	Boolean(읽기 전용)	확대/축소나 팬으로 자동 스크롤이 일시 정지된 동안 True입니다. Auto-scroll을 참조하십시오.

Dragging	Boolean(읽기 전용)	박스 확대/축소, 팬, 분할선, 드로잉 드래그가 진행 중인 동안 True입니다.
DragMode	TsgcChartDragMode(읽기 전용)	`cdmNone` , `cdmBox` , `cdmPan` , 또는 재무 에디션에서는 `cdmSplitter` (패널 분할선이 드래그됨, Panes 참조), `cdmDrawCreate` 나 `cdmDrawMove` (드로잉이나 그 기준점 중 하나가 드래그됨)입니다.
Chart	TComponent(읽기 전용)	이 인터랙션이 연결된 차트입니다.

메서드

메서드	반환값	설명
MouseMove(X, Y: Double)		차트 픽셀 단위의 포인터 이동을 전달합니다. 크로스헤어를 갱신하거나, 진행 중인 드래그를 계속하거나, 플롯 위에서 유휴 상태일 때 **TsgcChart.OnCrosshairMove**를 발생시킵니다.
MouseDown(X, Y: Double; aButton: TsgcChartMouseButton; aShift: TsgcChartShiftState)		버튼 누름을 전달합니다. **aButton**은 `cmbLeft` , `cmbRight` 또는 `cmbMiddle` 입니다. **aShift**는 `cskShift` , `cskCtrl` , `cskAlt` 의 집합입니다. 위의 속성들에 따라 박스 확대/축소, 팬, 분할선 드래그, 또는 드로잉 드래그를 시작합니다.
MouseUp(X, Y: Double; aButton: TsgcChartMouseButton; aShift: TsgcChartShiftState)		버튼 뗌을 전달합니다. **MouseDown**이 시작한 드래그를 완료하거나, 포인터가 드래그 임계값을 넘어 이동한 적이 없으면 클릭으로 처리하여 **TsgcChart.OnSeriesClick**을 발생시킵니다.
MouseWheel(X, Y: Double; aDelta: Integer; aShift: TsgcChartShiftState)		휠 노치를 전달합니다. **aDelta**는 노치당 120이며, 양수는 사용자로부터 멀어지는 방향(확대)입니다.
MouseLeave		포인터가 컨트롤을 벗어남을 전달하여 크로스헤어와 툴팁을 숨깁니다.
DoubleClick(X, Y: Double)		차트 픽셀 단위의 더블 클릭을 전달합니다. **ResetZoom**을 호출합니다.
Gesture(aKind: TsgcChartGestureKind; aCenterX, aCenterY, aFactorOrDX, aDY: Double)		터치 제스처를 전달합니다. `cgZoom` 의 경우 **aFactorOrDX**는 **aCenterX, aCenterY**를 중심으로 한 핀치 배율입니다(1보다 크면 확대). `cgPan` 의 경우 **aFactorOrDX**와 **aDY**는 픽셀 단위의 손가락 이동량입니다.
ResetZoom		모든 축의 확대/축소를 리셋하고 자동 스크롤을 재개합니다.
DeleteSelectedDrawing	Boolean	재무 에디션 전용입니다. Locked가 아닌 한 **TsgcChart.Drawings**에서 선택된 드로잉을 제거합니다. 뷰는 Delete 키에 대해 이를 호출합니다. 드로잉이 제거되었으면 True입니다.
HitTest(X, Y: Double)	TsgcChartHitInfo	마지막 **TsgcChart.Render** 이후 차트 픽셀 위치 아래에 무엇이 있는지를, 부작용 없이 알려 줍니다. HitTest and TsgcChartHitInfo를 참조하십시오.
CheckAutoScroll	Boolean	범위가 계산되기 전에 **TsgcChart.Render**가 호출합니다. 마지막 상호작용 이후 **AutoScrollResumeDelay**가 지나

		면 수평 확대/축소를 리셋합니다. 리셋했으면 True입니다. Auto-scroll을 참조하십시오.
DrawOverlay(aCanvas: TsgcUICanvas; const aRect: TsgcUIRect; aScale: Double)		차트 내용이 그려진 뒤 **TsgcChart.Render**가 호출합니다. **aCanvas**의 **aRect** 안에 크로스헤어, 툴팁, 러버 밴드를 그립니다.

이벤트

이 세 이벤트는 TsgcChartInteraction이 아니라 **TsgcChart** 자체에 공개되어 있습니다. 인터랙션은 이를 자신이 연결된 차트에서 발생시키므로, 애플리케이션은 차트에 이를 지정합니다. TsgcChart를 참조하십시오.

이벤트	형식	설명
TsgcChart.OnCrosshairMove	TsgcChartHitEvent	유휴 상태(진행 중인 드래그 없음)에서 포인터가 플롯 위로 이동했습니다. **aHit**은 포인터 위치에서의 **HitTest** 결과입니다.
TsgcChart.OnSeriesClick	TsgcChartHitEvent	왼쪽 클릭이 점의 클릭 반경 안에 떨어졌습니다. **aHit.Series**와 **aHit.Index**가 이를 식별합니다.
TsgcChart.OnZoom	TNotifyEvent	대화형 확대/축소, 팬, 또는 **ResetZoom** 이후에 발생합니다.

TsgcChartHitEvent는 다음과 같이 선언되어 있습니다:

HitTest and TsgcChartHitInfo

HitTest는 마지막 **TsgcChart.Render**가 계산한 플롯 사각형을 사용하여 차트 픽셀 위치가 무엇 위에 있는지 알려 줍니다. 아무것도 변경하지 않으므로 **MouseMove**, **MouseDown**, **MouseUp** 밖에서도 포함해 언제든지 안전하게 호출할 수 있습니다. 다음 필드를 가진 **TsgcChartHitInfo** 레코드를 반환합니다:

필드	형식	설명
Area	TsgcChartHitArea	해당 위치 아래의 차트 부분입니다: `chaNone`, `chaPlot`, `chaAxis`, `chaLegend` 또는 `chaTitle` 입니다.
Axis	TsgcChartAxis	**Area**가 `chaAxis` 일 때 해당 위치 아래의 축입니다.
Series	TComponent	가장 가까운 점의 **TsgcChartSeries**이며, 없으면 nil입니다. TsgcChartInteraction이 TsgcChart와 시리즈 형식들보다 먼저 컴파일되므로 TComponent로 선언되어 있습니다.
SeriesIndex	Integer	차트에서 **Series**의 인덱스이며, 없으면 -1입니다.
Index	Integer	점의 논리적 버퍼 인덱스이며, 0이 가장 오래된 점이고, 없으면 -1입니다.
XValue, YValue	Double	점의 값입니다. **YValue**는 캔들의 종가입니다.
IsOHLC	Boolean	점이 OHLC나 캔들스틱 시리즈에서 왔을 때 True입니다.
Open, High, Low, Close	Double	**IsOHLC**가 True일 때만 설정됩니다.
PixelX, PixelY	Double	차트 픽셀 단위의 점 위치입니다.

Distance	Double	조회한 위치와 점 사이의 픽셀 거리입니다. 막대나 캔들 내부에서는 0입니다.
CursorX, CursorY	Double	**HitTest**에 주어진 조회 위치입니다.
SubIndex	Integer	위치 아래에 있는 시리즈의 항목입니다. 슬라이스, 레이더 점, 히트맵 셀, 트리맵 깊이 또는 깔때기 단계이며, 없으면 -1입니다.
ZValue	Double	항목의 세 번째 값(히트맵 셀 값)이며, 없으면 NaN입니다.
Percent	Double	시리즈에서 항목이 차지하는 비율(원형, 깔때기, 트리맵)이며, 없으면 NaN입니다.
Text	string	항목의 카테고리 텍스트(슬라이스, 스포크, 셀, 노드 경로 또는 단계)이며, 없으면 비어 있습니다.

TsgcChart.OnSeriesClick은 **Distance**가 점의 작은 클릭 반경 안에 있을 때만 발생합니다. 빈 플롯 영역에 대한 클릭도 내부적으로 **HitTest**를 호출하지만 이벤트는 발생시키지 않습니다.

플롯 영역: 호버, 클릭, 시리즈 동작

호버, 클릭, 툴팁은 모든 플롯 영역 모드의 시리즈에 전달됩니다. 원형, 깔때기, 트리맵 셀, 레이더 극좌표 영역, XY 영역 안의 히트맵입니다. 확대/축소, 이동, 십자선, 더블 클릭 확대/축소 초기화는 XY 영역에서만 동작합니다. 새로운 차트 이벤트는 없습니다. 호버는 **OnCrosshairMove**를, 클릭은 **OnSeriesClick**을 발생시키며 둘 다 위의 히트 필드를 가집니다. **OnSeriesClick** 뒤에 시리즈가 **AreaClick**을 통해 자신의 클릭 동작을 실행하므로, **ExplodeOnClick**이 있는 원형 차트는 슬라이스를 끌어내고 **DrillOnClick**이 있는 트리맵은 그룹으로 드릴다운합니다. 시리즈는 자신의 툴팁 텍스트를 제공할 수 있으며 일반 줄로 그려집니다. 꺾은선 시리즈 아래에 히트맵이 있으면 클릭 반경 안의 꺾은선 점이 우선하고, 그렇지 않으면 히트맵 셀이 우선합니다.

Auto-scroll

0이 아닌 **TsgcChart.TimeWindow**를 가진 차트는 보통 수평 축이 최신 데이터를 계속 표시하도록 유지합니다. 수평 축을 확대/축소하거나 팬하면 이 동작이 멈추어, 새 데이터가 계속 도착하는 동안에도 확대/축소된 범위가 화면에 유지됩니다. 이 상태인 동안 **Paused**는 True를 보고합니다. 별도의 일시 정지나 재개 메서드는 없습니다. **TsgcChart.Render**는 매 렌더링마다 **CheckAutoScroll**을 호출하며, 상호작용 없이 **AutoScrollResumeDelay** 밀리초가 지나면 수평 확대/축소가 리셋되고 시간 윈도우가 다시 스크롤됩니다. **AutoScrollResumeDelay**를 0으로 설정하면 자동 재개가 비활성화되어 **ResetZoom**만이 되돌아가는 유일한 방법이 됩니다. 타이머는 전혀 관여하지 않으며, 검사는 렌더링 시점에만 일어납니다.

예제: 합성 입력을 공급하고 히트 읽기

TsgcChartInteraction은 단순한 좌표만 받으므로 이를 다루는 데 VCL이나 FMX 폼이 필요하지 않습니다. 콘솔 프로그램도 차트가 최소 한 번 렌더링된 뒤에는(그래야 **PlotRect**를 알 수 있으므로) 이를 직접 다룰 수 있습니다:
이는 크로스헤어 히트, 이에 대응하는 **HitTest** 결과, 그리고 클릭으로 확인된 시리즈와 인덱스를 출력합니다. 예를 들면:

참고 항목

- TsgcChart
- Axes
- TsgcChartSeries_Line
- TsgcVCLChart
- TsgcFMXChart
- Panes
- 플롯 영역

TsgcChartDataSource

Unit: sgcCharts_DataSource
모 든 차 트 데 이 터 소 스 의 기 반 클 래 스 입 니 다. **Buffer**(TsgcChartBuffer)에 점 을 추 가 하 고 **OHLCBuffer**(**TsgcChartOHLCBuffer**)에 캔들을 추가합니다. 두 속성 모두 코드에서 설정되며 데이터 소스가 소유하지 않습니다. 모든 추가 작업은 버퍼 자체의 잠금을 거치므로, 렌더링 스레드가 스냅샷을 읽는 동안 피드 스레드가 추가 작업을 수행할 수 있습니다.
Mapping은 각 수신 레코드에서 X, Y, OHLC 값의 위치를 데이터 소스에 알려줍니다. JSON 입력의 경우 JSON 경로, CSV 입력의 경우 열 이름 또는 0부터 시작하는 열 인덱스, 데이터셋의 경우 필드 이름입니다. **XPath**가 비어 있으면 X는 0부터 시작하는 레코드 인덱스가 됩니다. **Buffer**가 할당되어 있고 **YPath**가 설정되어 있으면 레코드는 점을 생성하고, **OHLCBuffer**가 할당되어 있고 **OpenPath**, **HighPath**, **LowPath**, **ClosePath**가 설정되어 있으면 캔들을 생성합니다(**VolumePath**는 선택 사항이며, 없으면 거래량은 0입니다). X 또는 값을 읽거나 변환할 수 없는 레코드는 건너뜁니다.
Mapping.TimeFormat은 X를 변환합니다. `ctfNumber`는 숫자를 그대로 유지하고, `ctfUnixSeconds`와 `ctfUnixMilliseconds`는 유닉스 시간을 UTC `TDateTime`으로 변환하며, `ctfISO8601`은 ISO 8601 텍스트를 파싱합니다(`Z` 또는 `+hh:mm` 오프셋은 UTC로 변환되고, 시간대가 없는 텍스트는 그대로 유지됩니다). 일반 숫자는 `TDateTime`으로 그대로 유지됩니다. 숫자는 로케일에 관계없이 항상 소수점 구분 기호로 `.`을 사용하여 파싱됩니다.

속성

속성	형식	기본값	설명
Active	Boolean	True	False이면 데이터 소스가 모든 추가 작업을 무시합니다.
Mapping	TsgcChartDataMapping	—	XPath, YPath, OpenPath, HighPath, LowPath, ClosePath, VolumePath, LabelPath, TimeFormat입니다. 위를 참조하십시오. **LabelPath**에 대해서는 아래의 카테고리 시리즈와 히트맵을 참조하십시오.
Throttle	Integer	0	**OnDataChanged** 알림 사이의 밀리초입니다. 아래 내용을 참조하십시오.
OnBeforeAppend	TsgcChartBeforeAppendEvent	nil	점이 추가되기 전에 발생하며, `aX`와 `aY`는 쓰기 가능하고 `aAccept`는 True로 설정되어 있습니다. `aAccept`를 False로 설정하면 해당 점을 건너뜁니다.
OnBeforeAppendOHLC	TsgcChartBeforeAppendOHLCEvent	nil	캔들이 추가되기 전에 발생하며, `aX`, `aOpen`, `aHigh`, `aLow`, `aClose`, `aVolume`은 쓰기 가능하고 `aAccept`는 True로 설정되어 있습니다. `aAccept`를 False로 설정하면 해당 캔들을 건너뜁니다.
OnDataChanged	TNotifyEvent	nil	하나 이상의 점 또는 캔들이 추가된 후 발생합니다. 아래의 알림 섹션을 참조하십시오.

메서드

메서드	설명
BeginUpdate / EndUpdate	해당 **EndUpdate**까지 알림을 지연시키며, 그 사이에 무언가 추가되었다면 하나의 (제한된) 알림을 보냅니다. 자유롭게 중첩할 수 있으며, 가장 바깥쪽 **EndUpdate**만 발생시킬 수 있습니다.
Clear	**Buffer**와 **OHLCBuffer**를 지우고, 바인딩된 카테고리 시리즈 또는 히트맵을 비우며, **XPath**가 비어 있을 때 사용되는 레코드 인덱스를 초기화합니다.

알림과 스레딩

OnDataChanged는 제한(throttled)됩니다. 추가 작업 후, 마지막 알림이 최소 **Throttle** ms 이전이면 이벤트가 즉시 발생하고, 그렇지 않으면 제한 창이 닫힐 때 도우미 스레드가 이를 발생시킵니다. 대기 중인 알림은 항상 하나 이하이므로, 연속된 추가 작업은 창당 하나의 이벤트만 생성합니다. **Throttle** = 0이면 모든 추가마다 알립니다(알림이 대기 중인 동안에는 여전히 하나로 합쳐짐).
Delphi 2010 이상에서는 `TThread.Queue`를 통해 **OnDataChanged**를 메인 스레드에서 전달합니다. 콘솔 애플리케이션이나 서비스는 이를 수신하려면 `CheckSynchronize`를 호출해야 합니다. Delphi 7부터 2009까지는 `TThread.Queue`가 없으므로, 이벤트는 추가 작업을 수행한 스레드나 제한 스레드에서 발생하며, 핸들러는 컨트롤을 다루기 전에 스스로 동기화해야 합니다.

카테고리 시리즈와 히트맵

Series 속성은 X/Y 시리즈 또는 캔들 시리즈뿐 아니라 카테고리 시리즈(원형, 레이더, 깔때기, 트리맵)나 히트맵도 바인딩합니다. **Mapping.LabelPath**는 항목 텍스트의 JSON 경로 또는 CSV 열입니다. 레이블이 있는 카테고리 레코드는 **SetValue**(label, Y)를 호출하므로 같은 텍스트에서는 최신 값이 우선합니다. 레이블 매핑이 없으면 가장 가까운 인덱스로 반올림한 X 위치의 항목을 설정합니다(**XPath**가 비어 있으면 X는 레코드 인덱스입니다). 레이블이 매핑되어 있어도 없거나 비어 있는 레코드는 항목을 쓰지 않으며, **OnBeforeAppend**는 여전히 (X, Y)를 필터링합니다. 히트맵 레코드는 라인 하나를 추가합니다. **YPath**가 배열인 JSON 레코드(키는 **XPath**가 설정되었으면 X이고, 아니면 실행 중인 라인 카운터)이거나 CSV 레코드의 셀입니다. 숫자가 아닌 값은 null 셀이 됩니다. 로드는 바인딩된 시리즈를 **BeginUpdate** / **EndUpdate** 안에서 쓰므로 한 번만 다시 그리고, 데이터 소스는 디자인 타임에 시리즈에 쓰지 않으므로 라이브 데이터가 DFM에 들어가지 않습니다. dataset 및 table 소스는 **LabelPath** 대신 **LabelField**와 **LabelColumn**을 사용합니다.

하위 클래스

- TsgcChartDataSource_Manual: 코드에서 직접 추가합니다.
- TsgcChartDataSource_DataSet: `TDataSet`에서 로드합니다.
- TsgcChartDataSource_JSON: JSON 문서에서 로드합니다.
- TsgcChartDataSource_CSV: CSV 텍스트에서 로드합니다.
- TsgcChartDataSource_Table: sgcGrid 데이터 모델처럼 `IsgcUITable`을 구현하는 테이블에서 불러옵니다.
- 실시간 sgcWebSockets 연결로 공급되는 브리지 데이터 소스입니다.

TsgcChartDataSource_Manual

Unit: sgcCharts_DataSource
이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **OHLCBuffer**, **Mapping**, **Active**, **Throttle**, **BeginUpdate**/**EndUpdate**, **Clear** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다. 새로운 속성은 추가하지 않으며, 애플리케이션이 이미 값을 가지고 있고 레코드에서 값을 찾기 위해 **Mapping**이 필요하지 않을 때 직접 호출하는 세 가지 메서드만 추가합니다.

메서드

메서드	반환 값	설명
AppendValue(const aX, aY: Double)	Boolean	**Buffer**에 점 하나를 추가합니다. 데이터 소스가 비활성 상태이거나, **Buffer**가 nil이거나, **OnBeforeAppend**가 점을 거부한 경우 False를 반환합니다.
AppendOHLC(const aX, aOpen, aHigh, aLow, aClose: Double; const aVolume: Double = 0)	Boolean	**OHLCBuffer**에 캔들 하나를 추가합니다. 데이터 소스가 비활성 상태이거나, **OHLCBuffer**가 nil이거나, **OnBeforeAppendOHLC**가 캔들을 거부한 경우 False를 반환합니다.
AppendJSON(const aJSON: string)	Boolean	TsgcChartDataSource_JSON이 문서를 읽는 것과 같은 방식으로, **Mapping**을 통해 JSON 객체 또는 객체나 배열의 배열을 읽습니다. 잘못된 형식의 JSON에 대해서는 False를 반환하며, 이에 대해 예외를 발생시키는 일은 없습니다.
AppendCategory(const aText: string; aValue: Double)	Boolean	바인딩된 카테고리 시리즈에 **aText**의 값을 씁니다(upsert이며 **OnBeforeAppend**의 X는 항목 인덱스입니다). 데이터 소스가 비활성이거나, 바인딩된 카테고리 시리즈가 없거나, **OnBeforeAppend**가 거부했으면 False입니다.
AppendLine(aKey: Double; const aValues: array of Double)	Boolean	바인딩된 히트맵의 **aKey** 위치에 라인 하나를 추가합니다. 데이터 소스가 비활성이거나 바인딩된 히트맵이 없으면 False입니다.

예제: 코드에서 점 추가

예제: 원형 차트와 히트맵에 데이터 공급하기

참고 항목

- TsgcChartDataSource
- TsgcChartBuffer

TsgcChartDataSource_DataSet

Unit: sgcCharts_DataSource_DataSet
`TDataSet`의 행에서 차트 데이터를 로드합니다. 이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **OHLCBuffer**, **Active**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다. 필드가 직접 이름으로 지정되므로 상속된 **Mapping** 속성은 사용하지 않습니다.
XField, **YField**, 그리고 **OpenField**부터 **VolumeField**까지 필드 이름을 지정합니다. **XField**가 비어 있으면 X는 0부터 시작하는 레코드 인덱스가 됩니다. 날짜/시간 필드는 그 `TDateTime`을 X로 제공합니다. 숫자 필드는 **Mapping.TimeFormat**(유닉스 초 또는 밀리초)에 따라 변환됩니다. 텍스트 필드는 CSV 텍스트처럼 파싱됩니다. X 또는 Y가 null인 행은 점을 생성하지 않고, open, high, low, close 중 하나라도 null인 행은 캔들을 생성하지 않으며, volume이 null이면 0입니다.

속성

속성	형식	설명
DataSet	TDataSet	행을 로드할 데이터셋입니다.
XField	string	X 값을 담고 있는 필드입니다. 비어 있으면 0부터 시작하는 레코드 인덱스를 의미합니다.
YField	string	Y 값을 담고 있는 필드입니다.
OpenField, HighField, LowField, CloseField	string	OHLC 값을 담고 있는 필드입니다. 행이 캔들을 생성하려면 네 필드 모두 설정되어야 합니다.
VolumeField	string	거래량을 담고 있는 필드입니다. 선택 사항이며, null이거나 설정되지 않은 필드는 0입니다.
LabelField	string	바인딩된 카테고리 시리즈(원형, 레이더, 깔때기, 트리맵)의 항목 텍스트 필드입니다. 각 행은 **SetValue**(text, Y)를 호출하며, 텍스트가 null이거나 비어 있는 행은 항목을 쓰지 않습니다. **LabelField**가 없으면 각 행은 자신의 X 위치의 항목을 설정합니다(**XField**가 비어 있으면 레코드 인덱스). 카테고리 시리즈와 히트맵을 참조하십시오.

메서드

메서드	설명
LoadFromDataSet	열려 있는 **DataSet**의 모든 행을 추가하며, 컨트롤을 비활성화한 뒤 작업 후 현재 레코드를 복원하고 마지막에 하나의 알림을 보냅니다. **DataSet**이 nil이거나 닫혀 있으면 아무 동작도 하지 않습니다.
Refresh	버퍼를 지우고 **LoadFromDataSet**을 다시 호출합니다. **DataSet**이 nil이거나 닫혀 있으면 아무 동작도 하지 않습니다.

예제: 데이터셋에서 버퍼로 로드

참고 항목

- TsgcChartDataSource
- TsgcChartBuffer

TsgcChartDataSource_JSON

Unit: sgcCharts_DataSource_JSON
JSON 문서에서 차트 데이터를 로드합니다. 이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **OHLCBuffer**, **Mapping**, **Active**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다.
ArrayPath는 레코드를 선택합니다. **ArrayPath**가 비어 있으면 문서 루트입니다. 객체 또는 배열의 배열은 요소마다 하나의 레코드를 생성하고, 객체는 하나의 레코드를 생성합니다. 각 레코드는 TsgcChartDataSource에서 설명한 것과 정확히 같은 방식으로 **Mapping**을 통해 읽힙니다. 로드 작업은 버퍼에 추가되므로, 데이터를 교체하려면 추가하는 대신 먼저 **Clear**를 호출하십시오. 마지막에는 하나의 알림을 보냅니다. 잘못된 형식의 JSON이거나 **ArrayPath**가 없으면 `EsgcChartJSONError`가 발생합니다.

속성

속성	형식	설명
ArrayPath	string	문서 안에서 레코드의 배열(또는 객체)에 대한 경로입니다. 비어 있으면 문서 루트를 의미합니다.

메서드

메서드	설명
LoadFromString(const aJSON: string)	JSON 문자열에서 로드합니다.
LoadFromStream(aStream: TStream)	UTF-8 텍스트 스트림에서 로드합니다. BOM은 선택 사항입니다.
LoadFromFile(const aFileName: string)	UTF-8 파일에서 로드합니다. BOM은 선택 사항입니다.

예제: JSON 배열에서 캔들 로드

참고 항목

- TsgcChartDataSource
- TsgcChartDataSource_Manual (AppendJSON)
- TsgcChartBuffer

TsgcChartDataSource_CSV

Unit: sgcCharts_DataSource_CSV
CSV 텍스트(RFC 4180)에서 차트 데이터를 로드합니다. 이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **OHLCBuffer**, **Mapping**, **Active**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다.
필드는 **Delimiter**로 구분됩니다. 필드는 "로 따옴표 처리될 수 있으며, 따옴표로 묶인 필드는 구분 기호, 줄바꿈, 이중 따옴표를 포함할 수 있습니다. 줄은 CRLF, LF 또는 CR로 끝나며, 빈 줄은 건너뜁니다. **HasHeader**가 True이면 첫 번째 레코드가 열 이름을 지정합니다.
Mapping 경로는 열을 선택합니다. 헤더 이름(대소문자 구분 안 함) 또는 0부터 시작하는 열 인덱스입니다. 숫자는 로케일에 관계없이 **DecimalSeparator**(기본값 .)를 사용합니다. 매핑된 값이 없거나 숫자가 아닌 행은 건너뜁니다. 로드 작업은 버퍼에 추가되므로(데이터를 교체하려면 먼저 **Clear**를 호출), 마지막에 하나의 알림을 보냅니다.

속성

속성	형식	기본값	설명
Delimiter	Char	,	필드 구분 기호입니다.
HasHeader	Boolean	True	True이면 첫 번째 레코드가 열 이름을 지정하며, Mapping 경로는 열 인덱스 대신 그 이름을 사용할 수 있습니다.
DecimalSeparator	Char	.	로케일에 관계없이 숫자 필드를 파싱하는 데 사용되는 소수점 구분 기호입니다.

메서드

메서드	설명
LoadFromString(const aText: string)	CSV 문자열에서 로드합니다.
LoadFromStream(aStream: TStream)	UTF-8 텍스트 스트림에서 로드합니다. BOM은 선택 사항입니다.
LoadFromFile(const aFileName: string)	UTF-8 파일에서 로드합니다. BOM은 선택 사항입니다.

예제: CSV 파일에서 점 로드

참고 항목

- TsgcChartDataSource
- TsgcChartBuffer

TsgcChartDataSource_Table

Unit: sgcCharts_DataSource_Table
sgcUICore의 `IsgcUITable` 인터페이스(unit `sgcUICore_Table`)를 구현하는 모든 테이블, 예를 들어 sgcGrid 그리드의 데이터 모델에서 차트 데이터를 불러옵니다. 두 제품 사이에 종속성은 없습니다. 이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **Series**, **Active**, **Throttle**과 추가 전 이벤트 및 데이터 변경 이벤트를 상속합니다. 열을 직접 지정하므로 상속된 **Mapping** 경로는 사용하지 않습니다.
XColumn과 **YColumn**은 열을 지정하며, 테이블의 열 이름과 대소문자를 구분하지 않고 비교됩니다(sgcGrid에서는 열의 **FieldName**). **XColumn**이 비어 있으면 X는 0부터 시작하는 행 인덱스가 됩니다. 날짜/시간 값은 해당 `TDateTime`을 X로 사용하고, 숫자 값은 **Mapping.TimeFormat**으로 변환되며, 텍스트 값은 CSV 텍스트처럼 구문 분석됩니다. X 또는 Y가 비어 있거나 변환할 수 없는 행은 점을 만들지 않습니다. 행은 테이블의 순서대로 읽으며, sgcGrid에서는 정렬 및 필터가 적용된 보기의 순서입니다.
소스는 테이블에 리스너를 등록합니다. **Active**가 True이고 마지막 로드 이후 테이블 버전이 바뀌었으면, 테이블이 변경될 때마다 전체 테이블을 다시 불러옵니다. 다시 불러오기는 테이블을 변경한 스레드에서 실행되며, 차트 알림은 그 후 다른 데이터 소스와 같은 방식으로 전달됩니다. **SetTable**로 바인딩한 테이블은 해제하기 전에 `SetTable(nil)`로 바인딩을 해제해야 하며, **Table** 컴포넌트는 자동으로 해제됩니다.

속성

속성	형식	설명
Table	TComponent	`IsgcUITable` 을 구현하는 컴포넌트. 이를 구현하지 않은 컴포넌트를 할당하면 예외가 발생합니다.
XColumn	string	X를 담은 열. 비어 있으면 0부터 시작하는 행 인덱스입니다.
YColumn	string	Y를 담은 열.
LabelColumn	string	바인딩된 카테고리 시리즈의 항목 텍스트 열이며, `;`로 구분한 여러 열도 가능합니다. 여러 열의 텍스트는 트리맵이면 **PathDelimiter**로(Region, Country, City 열이 있는 테이블이 3단계 트리맵에 데이터를 공급합니다), 그렇지 않으면 `/` 로 연결되며, 뒤쪽의 빈 부분은 버려집니다. 히트맵에서는 라인 레이블입니다.

메서드

메서드	설명
SetTable	`IsgcUITable` 을 구현하는 모든 객체, 예를 들어 sgcGrid 그리드의 **Data** 모델을 바인딩합니다. nil은 바인딩을 해제합니다.
GetTable	바인딩된 테이블을 반환하며, 없으면 nil입니다.
LoadFromTable	테이블의 모든 행을 추가하고, 읽는 동안 테이블 잠금을 유지하며, 끝에 알림을 한 번 보냅니다.
Refresh	버퍼를 지우고 테이블을 다시 불러옵니다. 바인딩된 테이블이 없으면 아무 일도 하지 않습니다.

카테고리 시리즈와 히트맵

카테고리 시리즈에서는 각 행이 **SetValue**(label, Y)를 호출하며, 레이블이 비어 있는 행은 항목을 쓰지 않습니다. **LabelColumn**이 없으면 각 행은 자신의 X 위치의 항목을 설정합니다(**XColumn**이 비어 있으면 행 인덱스). 다시 로드는 시리즈의 **BeginReload** / **EndReload** 안에서 실행되므로 항목은 색상과 플래그를 유지하고 제거된 행의 항목은 사라집니다. 히트맵에서는 로드할 때마다 행렬이 행당 라인 하나로 교체됩니다(키는 **XColumn**이 설정되었으면 X이고, 아니면 행 인덱스입니다). 셀은 ;로 구분한 **YColumn** 열이며, **YColumn**이 비어 있으면 **XColumn**과 레이블 열을 제외한 모든 숫자 열입니다. 셀 레이블은 열 이름입니다. 피벗 결과나 그리드를 위한 경로입니다.

예제: 테이블에서 데이터를 공급받는 원형 차트

참고 항목

- TsgcChartDataSource
- TsgcChartSeries_Pie
- TsgcChartSeries_Heatmap

브리지 개요

sgcChartsWS 패키지는 실시간 sgcWebSockets 연결로 공급되는 차트 데이터 소스를 추가합니다. 이는 핵심 sgcCharts 패키지와는 별도의 패키지이며 컴파일하려면 sgcWebSockets가 필요합니다. 핵심 데이터 소스(TsgcChartDataSource 및 그 수동, 데이터셋, JSON, CSV 파생 클래스)만 사용하는 프로젝트는 sgcWebSockets가 전혀 필요하지 않습니다.

구성 요소

- TsgcChartDataSource_MQTT: TsgcWSPClient_MQTT가 수신한 MQTT 발행 메시지를 차트로 표시합니다.
- TsgcChartDataSource_WebSocket: TsgcWebSocketClient가 수신한 JSON 메시지를 차트로 표시합니다.
- TsgcChartDataSource_SSE: TsgcHTTP1Client가 수신한 Server-Sent Events를 차트로 표시합니다.
- TsgcChartDataSource_Binance: TsgcWSAPI_Binance를 통해 Binance 공개 시장 데이터 스트림을 구독하고 차트로 표시합니다.

핸들러 체이닝

각 브리지 데이터 소스는 동일한 방식으로 동작합니다. **Active**가 설정되어 있고 클라이언트 속성이 할당되어 있는 동안, 클라이언트에 자체 이벤트 핸들러(OnMQTTPublish, OnMessage, OnSSEMessage, 또는 Binance 클라이언트의 메시지 이벤트)를 설치하고 이전에 있던 핸들러를 그대로 유지합니다. 수신되는 모든 메시지는 먼저 차트로 표시된 다음 이전 핸들러로 전달되므로, 애플리케이션 자체의 핸들러는 차트 데이터 소스가 없는 것처럼 그대로 계속 실행됩니다.
Active를 False로 설정하거나, 클라이언트 속성을 지우거나, 데이터 소스를 파괴하면 그 사이에 애플리케이션이 핸들러를 다시 교체하지 않은 한 이전에 있던 핸들러가 복원됩니다. 이 후킹은 런타임에서만 발생하며 디자인 타임에는 발생하지 않으므로, 데이터 소스가 연결된 후 자체 핸들러를 할당하는 애플리케이션은 두 핸들러를 모두 유지하려면 클라이언트 속성을 설정하기 전(또는 **Active**를 전환하기 전)에 핸들러를 할당해야 합니다.
메시지는 기본 클라이언트가 알림을 보내는 스레드에서 도착합니다. 추가 작업은 기본 TsgcChartDataSource의 잠금 및 제한된(throttled) 알림을 거치므로, 어떤 알림 모드를 사용하더라도 브리지 데이터 소스와 함께 안전하게 사용할 수 있습니다.

TsgcChartDataSource_MQTT

Unit: sgcChartsWS_DataSource_MQTT
sgcWebSockets MQTT 클라이언트로부터 공급되는 차트 데이터 소스입니다. 이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **OHLCBuffer**, **Mapping**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다. 핸들러 체이닝이 동작하는 방식은 브리지 개요를 참조하십시오.
TopicFilter는 MQTT 와일드카드 `+`(한 레벨)와 `#`(이 레벨과 그 아래의 모든 레벨)를 사용하여 차트로 표시할 발행 메시지를 선택합니다. **TopicFilter**가 비어 있으면 모든 발행 메시지를 차트로 표시합니다.
PayloadFormat이 `cpfJSON`이면 **Mapping**을 통해 페이로드를 읽습니다(JSON 객체 또는 레코드 배열). `cpfNumber`는 페이로드를 숫자로 파싱하고(로케일에 관계없이 소수점 구분 기호로 `.` 사용) X로 `Now`를 사용합니다.

속성

속성	형식	기본값	설명
Active	Boolean	True	**Client**에 훅을 설치하거나 제거합니다.
Client	TsgcWSPClient_MQTT	nil	`OnMQTTPublish` 가 훅 처리되는 MQTT 클라이언트입니다.
PayloadFormat	TsgcChartMQTTPayloadFormat	cpfJSON	`cpfJSON` 또는 `cpfNumber` . 위 내용을 참조하십시오.
TopicFilter	string	''	MQTT 토픽 필터(`+` , `#`)입니다. 비어 있으면 모든 발행 메시지를 차트로 표시합니다.

메서드

메서드	반환 값	설명
ProcessPublish(const aTopic, aText: string)	Boolean	**Client**에서 수신된 것처럼 발행 메시지 하나를 차트로 표시합니다. 토픽이 **TopicFilter**와 일치하지 않거나 페이로드를 읽을 수 없는 경우 False를 반환합니다.

예제: MQTT로부터 센서 판독값 차트 표시

Client 속성은 `TsgcWSPClient_MQTT` 프로토콜 컴포넌트를 받으며, 이 컴포넌트는 다른 sgcWebSockets MQTT 클라이언트와 마찬가지로 자신의 **Client** 속성을 통해 전송 클라이언트(여기서는 `TsgcWebSocketClient`)에 연결됩니다.

참고 항목

- 브리지 개요
- TsgcChartDataSource

TsgcChartDataSource_WebSocket

Unit: sgcChartsWS_DataSource_WebSocket
sgcWebSockets WebSocket 클라이언트로부터 공급되는 차트 데이터 소스입니다. 이 컴포넌트는 TsgcChartDataSource 에서 파생되며 **Buffer**, **OHLCBuffer**, **Mapping**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다. 핸들러 체이닝이 동작하는 방식은 브리지 개요를 참조하십시오.
클라이언트가 수신하는 모든 텍스트 메시지는 JSON으로 취급되어 **Mapping**을 통해 읽힙니다. **MessageFilter**는 차트로 표시되기 위해 메시지 안에 존재해야 하는 선택적 경로입니다. 비워 두면 모든 메시지를 차트로 표시합니다.

속성

속성	형식	기본값	설명
Active	Boolean	True	**Client**에 훅을 설치하거나 제거합니다.
Client	TsgcWebSocketClient	nil	`OnMessage` 가 훅 처리되는 WebSocket 클라이언트입니다.
MessageFilter	string	''	차트로 표시되기 위해 메시지 안에 존재해야 하는 경로입니다. 비워 두면 모든 메시지를 차트로 표시합니다.

메서드

메서드	반환 값	설명
ProcessMessage(const aText: string)	Boolean	**Client**에서 수신된 것처럼 메시지 하나를 차트로 표시합니다. 메시지가 유효한 JSON이 아니거나 **MessageFilter**가 그 안에 존재하지 않는 경우 False 를 반환합니다.

예제: WebSocket을 통한 JSON 피드 차트 표시

참고 항목

- 브리지 개요
- TsgcChartDataSource

TsgcChartDataSource_SSE

Unit: sgcChartsWS_DataSource_SSE
sgcWebSockets HTTP 클라이언트의 Server-Sent Events 스트림으로부터 공급되는 차트 데이터 소스입니다. 이 컴포넌트는 TsgcChartDataSource에서 파생되며 **Buffer**, **OHLCBuffer**, **Mapping**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다. 핸들러 체이닝이 동작하는 방식은 브리지 개요를 참조하십시오.
각 SSE 이벤트의 데이터는 **Mapping**을 통해 JSON으로 읽힙니다. 하나 이상의 스트림 청크에 걸쳐 분할된 이벤트는 완료될 때까지 내부적으로 버퍼링됩니다.

속성

속성	형식	기본값	설명
Active	Boolean	True	**Client**에 훅을 설치하거나 제거합니다.
Client	TsgcHTTP1Client	nil	`OnSSEMessage` 가 훅 처리되는 HTTP 클라이언트입니다.

메서드

메서드	반환 값	설명
ProcessChunk(const aText: string)	Integer	**Client**에서 수신된 것처럼 하나의 스트림 청크에서 완료된 이벤트들을 차트로 표시합니다. 완료되지 않은 이벤트는 다음 청크를 기다립니다. 페이로드가 유효한 JSON이었던 이벤트의 개수를 반환합니다.
ResetStream	—	버퍼링된 미완료 이벤트를 폐기합니다.

예제: SSE 피드 차트 표시

참고 항목

- 브리지 개요
- TsgcChartDataSource

TsgcChartDataSource_Binance

Unit: sgcChartsWS_DataSource_Binance
sgcWebSockets Binance API 클라이언트로부터 공급되는 차트 데이터 소스입니다. 이 컴포넌트는 TsgcChartDataSource 에서 파생되며 **Buffer**, **OHLCBuffer**, **Throttle** 및 추가 전 이벤트와 데이터 변경 이벤트를 상속합니다(Binance 메시지의 형식은 스트림별로 고정되어 있으므로 상속된 **Mapping**은 사용하지 않습니다). 핸들러 체이닝이 동작하는 방식은 브리지 개요 를 참조하십시오.
Stream은 차트로 표시할 대상을 선택합니다. `cbsKLine`은 **Symbol**과 **Interval**에 대한 kline/캔들스틱 이벤트로부터 구성된 캔들을 (**OHLCBuffer**에) 차트로 표시합니다. `cbsTicker`와 `cbsTrade`는 **Symbol**에 대응하는 티커 또는 체결 스트림의 값을 (**Buffer**에) 차트로 표시합니다. `cbsKLine`을 사용할 때 **ClosedOnly**는 진행 중인 모든 업데이트가 아니라 캔들이 마감된 kline 만 차트로 표시합니다.
`cbsTrade`를 사용하면 **OnTrade**가 차트로 표시되기 전에 **Symbol**의 모든 체결을 시간, 가격, 수량(`q` 필드)과 함께 보고합니다. Financial edition에서는 **Aggregator**에 TsgcChartAggregator를 할당하여 체결로부터 캔들을 구성할 수 있습니다. 이 경우 각 체결은 **Buffer** 대신 `Aggregator.AddTrade(time, price, quantity)`로 전달되므로, 애그리게이터 시리즈의 캔들이 체결 거래량을 포함하게 되고 체결 자체는 선으로 표시되지 않으며 `OnBeforeAppend`도 해당 체결에 대해서는 발생하지 않습니다. **Aggregator**가 nil이면 체결은 이전과 같이 **Buffer**에 차트로 표시됩니다.

속성

속성	형식	기본값	설명
Active	Boolean	True	**API**의 클라이언트에 훅을 설치하거나 제거합니다.
Aggregator	TsgcChartAggregator	nil	Financial edition. `Stream = cbsTrade` 일 때, 모든 체결은 **Buffer**에 추가되는 대신 시간, 가격, 수량과 함께 `Aggregator.AddTrade` 로 전달됩니다. 애그리게이터가 해제되면 지워집니다.
API	TsgcWSAPI_Binance	nil	클라이언트 메시지가 훅 처리되는 Binance API 컴포넌트입니다.
ClosedOnly	Boolean	False	`Stream = cbsKLine` 일 때, 마감된 캔들만 차트로 표시합니다.
Interval	TsgcWSBinanceChartIntervals	bci1m	`Stream = cbsKLine` 일 때 사용되는 kline 간격입니다.
Stream	TsgcChartBinanceStream	cbsKLine	`cbsKLine`, `cbsTicker` 또는 `cbsTrade`.
Symbol	string	''	차트로 표시할 Binance 심볼입니다. 예: `BTCUSDT`.

메서드

메서드	반환 값	설명
Subscribe	Integer	**Symbol**(klines의 경우 **Interval**도 포함)에 대해 **Stream**을 구독합니다. API가 반환하는 id를 반환합니다.
Unsubscribe	Integer	동일한 스트림 구독을 취소합니다. API가 반환하는 id를 반환합니다.
ProcessMessage(const aText: string)	Boolean	**API.Client**에서 수신된 것처럼 Binance 메시지 하나를 차트로 표시합니다. 메시지가 **Stream/Symbol/Interval**에 해당하지 않거나 아무것도 차트로 표시되지 않은 경우 False를 반환합니다.

이벤트

이벤트	설명
OnTrade(Sender: TObject; aTime: TDateTime; aPrice, aQty: Double)	Public. `Stream = cbsTrade` 일 때, 데이터 소스가 활성 상태인 동안 체결이 추가되거나 집계되기 전에 **Symbol**의 모든 체결에 대해 발생합니다. aTime은 체결 시간(UTC), aPrice는 가격, aQty는 수량입니다(메시지에 `q` 가 없으면 0).

예제: 실시간 Binance kline 차트 표시

참고 항목

- 브리지 개요
- TsgcChartDataSource

TsgcChartSeries_Indicator

Unit: sgcCharts_Series_Financial
Registered 및 Trial 에디션에서만 지원됩니다(Community 에디션에는 포함되지 않습니다. 에디션 참조).
TsgcChartSeries_Indicator는 모든 금융 지표 시리즈의 추상 기반 클래스입니다. 다른 시리즈, 즉 자신의 **Source**인 캔들스틱 또는 OHLC 시리즈나 시가, 고가, 저가, 종가를 거래량 없이 Y로 대신하는 X/Y 시리즈의 바를 읽는 TsgcChartSeries의 하위 클래스이며, 원본 데이터 자체를 그리는 대신 자체적으로 계산된 출력을 내부 배열에 보관합니다. 구체적인 지표인 TsgcChartSeries_SMA, TsgcChartSeries_EMA, TsgcChartSeries_WMA, TsgcChartSeries_Bollinger, TsgcChartSeries_VWAP는 가격과 같은 축에 그려지므로, 일반적으로 캔들스틱 시리즈 위에 추가됩니다. 오실레이터 계열(TsgcChartSeries_Oscillator 와 그 하위 클래스인 RSI, Stochastic), MACD, ATR, TsgcChartSeries_Volume 도 TsgcChartSeries_Indicator에서 파생되지만, 일반적으로 가격 아래의 별도 패널에 그려지므로 각자의 주제에서 문서화되어 있습니다.

속성

속성	형식	설명
Source	TsgcChartSeries	이 지표가 추적하는 시리즈입니다. 캔들스틱/OHLC 시리즈이거나, X/Y 시리즈(그 Y가 거래량 없이 시가, 고가, 저가, 종가로 사용됨)입니다. 이를 변경하면 출력이 지워지고 다음 **Sync**에서 전체 계산이 강제로 수행됩니다.
LineWidth	Double	스케일 1에서의 디바이스 픽셀입니다. 기본값은 1.5입니다.
OutputCount	Integer(읽기 전용)	지표가 바마다 계산하는 출력 라인의 개수입니다. SMA와 같은 단일 라인 지표는 1이고, Bollinger와 같은 다중 라인 지표는 더 많습니다.
PointCount	Integer(읽기 전용)	처리된 바의 개수이며, 0이 가장 오래된 바입니다.
XValues[aIndex: Integer]	Double(읽기 전용)	aIndex에 있는 처리된 바의 X(보통 바 시간)입니다.
Values[aLine, aIndex: Integer]	Double(읽기 전용)	aIndex에 있는 처리된 바의 출력 aLine(0부터 시작, 최대 **OutputCount** - 1)입니다. 지표가 값을 산출할 만큼 충분한 바를 아직 보지 못한 워밍업 기간 동안에는 NaN입니다.

메서드

메서드	반환 값	설명
Sync		출력을 **Source**와 일치시킵니다. 원본과 동기화하기를 참조하십시오.
Recalculate		예를 들어 **Source**의 오래된 바를 제자리에서 다시 작성한 후처럼, 다음 **Sync**에서 전체 계산을 강제로 수행합니다.
GetDataRange(var aXMin, aXMax, aYMin, aYMax: Double)	Boolean	먼저 **Sync**를 호출한 다음 출력의 X와 Y 범위를 반환합니다. 출력이 없으면 False입니다.
Draw(aCanvas: TsgcUICanvas; aXAxis, aYAxis: TsgcChartAxis; const aPlot: TsgcUIRect; aScale: Double)		모든 출력을 **EffectiveColor**의 선으로 그립니다. Bollinger와 같은 하위 클래스는 이를 재정의하여 대신 밴드나 막대를 그립니다.

DrawLegendSymbol(aCanvas: TsgcUICanvas; const aRect: TsgcUIRect; aScale: Double)		이 지표의 범례 기호를 그립니다.

원본과 동기화하기

지표는 **Source**의 데이터를 절대 복사하지 않습니다. 그 바를 읽고 자신만의 출력만 보관합니다. **Sync**는 그 출력을 다시 **Source**와 일치시키는 역할을 하며, **GetDataRange**(따라서 렌더링할 때마다)와 히트 테스트에 의해 자동으로 호출되고, 직접 호출할 수도 있습니다. 수행하는 작업은 마지막 호출 이후 무엇이 변경되었는지에 따라 달라집니다:

- 아직 출력이 없거나, (**Period** 설정과 같은) 매개변수 변경이 있었거나, 원본이 지워지거나 다시 로드된 경우(이미 처리된 바보다 개수가 적거나, 최신 처리된 바의 X가 달라진 경우): 전체 계산을 수행하여 지표를 재설정하고 모든 바를 다시 공급하며, 처음부터 계산한 것과 같은 결과를 냅니다.
- 마지막 Sync 이후 **Source**에 추가된 바: 새 바만 공급됩니다.
- 최신 처리된 바가 제자리에서 변경된 경우, 즉 실시간 캔들의 경우 (**TsgcChartSeries_OHLCBase.UpdateLastOHLC** 참조): 마지막 출력만 다시 계산됩니다.
- 가득 찬 링 버퍼의 앞부분에서 바가 제거된 경우(원본의 **BufferMode** = `cbmOverwrite`): 제거된 바의 출력은 폐기되고 계산은 이전 상태에서 이어지므로, 남아 있는 출력은 계산에 사용된 이력을 그대로 유지합니다.

Sync는 **Source**의 바 개수, 가장 오래된 바와 최신 처리된 바의 X, 그리고 그 최신 바의 OHLCV 값을 비교하여 어느 경우에 해당하는지 감지합니다. **Source**의 오래된 바를 제자리에서 다시 작성한 후에는 **Recalculate**를 호출하십시오. Sync는 그 경우를 스스로 감지할 수 없기 때문입니다. 지표가 값을 산출할 만큼 충분한 바를 아직 보지 못한 워밍업 기간의 출력은 **Values**를 통해 NaN으로 읽히며 그려지지 않습니다. 모든 작업은 메인 스레드에서 실행되며, 원본 버퍼는 자신의 잠금 아래에서 읽힙니다.

TsgcChartSeries_SMA

종가의 단순 이동평균입니다. 출력 라인은 하나이며 인덱스는 0입니다.

속성	형식	설명
Period	Integer	평균을 계산할 바의 개수입니다. 기본값은 20입니다.

TsgcChartSeries_EMA

종가의 지수 이동평균이며, 단순 이동평균보다 최근 바에 더 큰 가중치를 둡니다. 출력 라인은 하나이며 인덱스는 0입니다.

속성	형식	설명
Period	Integer	평활 계수(smoothing factor)를 도출하는 바의 개수입니다. 기본값은 20입니다.

TsgcChartSeries_WMA

종가의 선형 가중 이동평균이며, 기간 내 가장 최근 바에 가장 큰 가중치를, 가장 오래된 바에 가장 작은 가중치를 둡니다. 출력 라인은 하나이며 인덱스는 0입니다.

속성	형식	설명
Period	Integer	가중치를 부여할 바의 개수입니다. 기본값은 20입니다.

TsgcChartSeries_Bollinger

볼린저 밴드입니다. 종가의 이동평균과, 그로부터 선택한 표준편차 개수만큼 떨어진 상단 밴드 및 하단 밴드로 구성됩니다. 두 밴드 모두 원본과 같은 가격 스케일을 따르므로, Bollinger는 이동평균과 마찬가지로 자신만의 패널이 아니라 가격 위에 그려집니다. 출력 라인은 세 개입니다: 0은 중간 밴드, 1은 상단 밴드, 2는 하단 밴드입니다.

속성	형식	설명
Period	Integer	중간 밴드와 표준편차를 계산할 바의 개수입니다. 기본값은 20입니다.
Deviations	Double	상단 및 하단 밴드가 중간 밴드로부터 떨어진 표준편차의 개수입니다. 기본값은 2입니다.
BandFill	Boolean	상단 밴드와 하단 밴드 사이의 영역을 채웁니다. 기본값은 True입니다.

TsgcChartSeries_VWAP

거래량 가중 평균 가격입니다. 0이 아닌 거래량과 함께 **AddOHLC**로 공급되는 OHLC 또는 캔들스틱 시리즈처럼 거래량을 가진 원본이 필요합니다. X/Y 원본은 거래량이 없으므로 모든 바가 동일한 가중치를 갖는 것처럼 평균을 계산합니다. 원본의 X는 바 시간으로 취급됩니다. 출력 라인은 하나이며 인덱스는 0입니다.

속성	형식	설명
SessionStart	TDateTime	세션이 시작되는 하루 중 비율(day fraction)이며, 0(기본값)은 자정을 의미합니다. 계산은 각 세션 경계에서 재설정됩니다.

예제: 캔들스틱 시리즈 위의 SMA, EMA, 볼린저 밴드

참고 항목

- TsgcChartSeries_Oscillator
- TsgcChartSeries_Volume
- TsgcChartAggregator
- TsgcChartDrawing
- 패널
- TsgcChartSeries_Candlestick
- TsgcChartSeries
- 에디션

TsgcChartSeries_Oscillator

Unit: sgcCharts_Series_Financial
Registered 및 Trial 에디션에서만 지원됩니다(Community 에디션에는 포함되지 않습니다. 에디션 참조).
TsgcChartSeries_Oscillator는 0..100 스타일 지표의 기반 클래스입니다. 데이터와 무관하게 Y 범위를 0..100으로 고정하고 과매수/과매도 가이드 라인 한 쌍을 점선으로 그릴 수 있는 TsgcChartSeries_Indicator의 하위 클래스입니다. TsgcChartSeries_RSI 와 TsgcChartSeries_Stochastic 이 여기서 파생됩니다. **TsgcChartSeries_MACD** 와 **TsgcChartSeries_ATR**도 마찬가지로 일반적으로 자신만의 패널에 그려지므로 이 페이지의 아래쪽에서 함께 문서화되어 있지만, TsgcChartSeries_Oscillator가 아니라 TsgcChartSeries_Indicator에서 직접 파생됩니다. 이들의 출력은 가격에서 파생된 값으로 자연스러운 0..100 상한이 없으므로, 고정된 범위 대신 유동적인 Y 범위를 유지합니다. 이 네 가지 지표는 모두 스케일이 가격과 맞지 않으므로 일반적으로 가격 패널 아래의 자신만의 패널에 배치됩니다.

속성

속성	형식	설명
GuideLines	Boolean	과매수 및 과매도 가이드 라인을 점선 수평선으로 그립니다. 기본값은 True입니다.

Overbought와 Oversold는 기반 클래스 자체에는 공개(published)되어 있지 않습니다. RSI와 Stochastic이 그 수준에 대해 서로 다른 기본값을 가지므로, 각 구체적인 오실레이터가 자신만의 기본 수준으로 이를 공개합니다. 아래의 TsgcChartSeries_RSI와 TsgcChartSeries_Stochastic을 참조하십시오.
TsgcChartSeries_Indicator의 모든 멤버, 즉 **Source**, 구체적인 클래스의 **Period** 계열 속성, **Sync**, **PointCount**, **Values**가 여기에도 그대로 적용됩니다. 지표가 자신의 **Source** 시리즈와 어떻게 일치 상태를 유지하는지는 원본과 동기화하기를 참조하십시오.

TsgcChartSeries_RSI

종가의 상대강도지수(RSI)입니다. 출력 라인은 하나이며 인덱스는 0입니다. 가이드 라인의 기본값은 30과 70입니다.

속성	형식	설명
Period	Integer	평균 상승폭과 하락폭을 계산할 바의 개수입니다. 기본값은 14입니다.
Overbought	Double	가이드 라인 수준입니다. 기본값은 70입니다.
Oversold	Double	가이드 라인 수준입니다. 기본값은 30입니다.

TsgcChartSeries_Stochastic

스토캐스틱 오실레이터입니다. 출력 라인은 두 개입니다: 0은 %K, 1은 %D(%K의 이동평균)입니다. 가이드 라인의 기본값은 20과 80입니다.

속성	형식	설명
KPeriod	Integer	%K의 룩백 기간입니다. 기본값은 14입니다.
KSmoothing	Integer	%K에 적용되는 평활 기간입니다. 기본값은 3입니다.
DPeriod	Integer	%D를 산출하는 %K 이동평균의 기간입니다. 기본값은 3입니다.
DColor	TsgcUIColor	%D 선의 색상입니다. %K는 단일 라인 지표와 마찬가지로 **EffectiveColor**로 그려집니다. 기본값은 주황색 강조 색상입니다.

Overbought	Double	가이드 라인 수준입니다. 기본값은 80입니다.
Oversold	Double	가이드 라인 수준입니다. 기본값은 20입니다.

TsgcChartSeries_MACD

종가의 이동평균 수렴/발산(MACD)입니다. TsgcChartSeries_Indicator에서 직접 파생됩니다. 그 스케일은 자신이 만들어진 가격 차이를 따르므로 고정된 0..100 범위나 과매수/과매도 가이드 라인이 없습니다. 다만 히스토그램이 걸쳐 있는 0 기준선이 항상 보이도록 Y 범위를 재정의합니다. 출력 라인은 세 개입니다: 0은 MACD 선, 1은 시그널 선, 2는 히스토그램(MACD 선에서 시그널 선을 뺀 값)입니다.

속성	형식	설명
FastPeriod	Integer	빠른 EMA의 기간입니다. 기본값은 12입니다.
SlowPeriod	Integer	느린 EMA의 기간입니다. 기본값은 26입니다.
SignalPeriod	Integer	시그널 선을 산출하는 MACD 선의 EMA 기간입니다. 기본값은 9입니다.
SignalColor	TsgcUIColor	시그널 선의 색상입니다. MACD 선은 **EffectiveColor**를 사용하며 히스토그램은 상승/하락 막대로 그려집니다. 기본값은 주황색 강조 색상입니다.

TsgcChartSeries_ATR

평균 실질 변동폭(ATR)이며, 변동성을 측정하는 지표입니다. MACD와 같은 이유로 TsgcChartSeries_Indicator에서 직접 파생됩니다. 그 값은 0..100 비율이 아니라 상한이 없는 양의 가격 범위이므로, 유동적인 Y 범위를 유지하며 가이드 라인이 없습니다. 출력 라인은 하나이며 인덱스는 0입니다.

속성	형식	설명
Period	Integer	평균을 계산할 바의 개수입니다. 기본값은 14입니다.

예제: 각자의 패널에 있는 RSI와 MACD

참고 항목

- TsgcChartSeries_Indicator
- TsgcChartSeries_Volume
- 패널
- TsgcChartSeries_Candlestick
- TsgcChartSeries
- 에디션

TsgcChartSeries_Volume

단위: sgcCharts_Series_Financial
Registered 및 Trial 에디션 전용(Community 에디션에서는 제외됩니다. Editions 참조).
TsgcChartSeries_Volume은 OHLC **Source**의 거래량 막대를, 해당 바가 상승 마감했는지 하락 마감했는지에 따라 색을 입혀 그립니다. 이는 TsgcChartSeries_Oscillator가 아니라 TsgcChartSeries_Indicator에서 직접 파생됩니다. 거래량에는 고정된 상한이 없으므로 0..100 척도 대신 기준선이 0에 고정된 유동적인 Y 범위를 유지합니다. 바마다 두 개의 출력을 계산합니다. 0은 거래량이고 1은 방향(종가 >= 시가이면 1, 그렇지 않으면 0)입니다. 다만 출력 0만 Y 범위 계산에 반영되므로 방향 플래그가 축을 늘리는 일은 없습니다. RSI, Stochastic, MACD, ATR과 마찬가지로 척도가 가격과 맞지 않으므로 보통 가격 아래의 별도 패널에 배치됩니다. 출력을 **Source**와 일치시키는 방법은 Synchronizing with the source를 참조하십시오. 동일한 규칙이 여기에도 적용됩니다.

속성

속성	형식	설명
UpColor	TsgcUIColor	종가가 시가보다 크거나 같은 바의 색상입니다. 기본값은 차트의 상승 색상에 투명도를 낮춘 것입니다.
DownColor	TsgcUIColor	종가가 시가보다 낮은 바의 색상입니다. 기본값은 차트의 하락 색상에 투명도를 낮춘 것입니다.
BarWidthPercent	Integer	바 하나가 X 슬롯에서 차지하는 비율이며, 1에서 100입니다. 기본값은 70입니다.

TsgcChartSeries_Indicator의 모든 멤버도 그대로 적용됩니다: **Source**, **Sync**, **Recalculate**, **PointCount**, **XValues**, **Values**입니다. 처리된 바(aIndex)의 거래량은 `Values[0, aIndex]`로, 상승/하락 플래그는 `Values[1, aIndex]`로 읽습니다.

메서드

메서드	반환값	설명
Draw(aCanvas: TsgcUICanvas; aXAxis, aYAxis: TsgcChartAxis; const aPlot: TsgcUIRect; aScale: Double)		TsgcChartSeries_Indicator가 기본으로 그리는 선 대신, **UpColor**의 상승 바와 **DownColor**의 하락 바로 거래량 막대를 그리도록 재정의되었습니다.
DrawLegendSymbol(aCanvas: TsgcUICanvas; const aRect: TsgcUIRect; aScale: Double)		선 기호 대신 막대 범례 기호와 일치하도록 재정의되었습니다.

예제: 캔들스틱 시리즈 아래의 거래량 막대

참고 항목

- TsgcChartSeries_Indicator
- TsgcChartSeries_Oscillator
- Panes
- TsgcChartSeries_Candlestick
- TsgcChartSeries
- Editions

패널

Unit: sgcCharts_Chart
Registered 및 Trial 에디션에서만 지원됩니다(Community 에디션에는 포함되지 않습니다. 에디션 참조).
TsgcChart는 일반적으로 하나의 플롯 영역을 가집니다. **Panes**에 하나 이상의 항목이 있으면, 그 플롯 영역은 위에서 아래로 쌓인 수평 대역으로 분할되며, 각 대역은 자신만의 수직 축과 자신만의 시리즈를 가지지만 모든 패널은 여전히 차트의 수평 축을 공유합니다. 이는 아래쪽에 RSI, MACD 또는 거래량을 함께 표시하는 가격 차트에 필요한 레이아웃입니다. 하나의 패널에는 가격 캔들이, 그 아래 다른 패널에는 오실레이터가 표시되며, 둘 다 같은 아래쪽 축을 공유하므로 함께 스크롤되고 확대/축소됩니다. **TsgcChartPane**은 컬렉션의 대역 하나이고, **TsgcChartPanes**는 컬렉션 자체이며, 축은 **TsgcChartAxis.Pane**을 통해 패널에 연결됩니다.

TsgcChartPane 속성

속성	형식	설명
Name	string	**TsgcChartAxis.Pane**과 **TsgcChartPanes.Find**가 조회하는 대상입니다(대소문자 구분 안 함).
HeightPercent	Double	패널의 가중치입니다. 표시되는 패널들은 이 값에 비례하여 플롯 높이를 나누어 가지며, 합이 100이 되어야 하는 실제 백분율은 아닙니다. 기본값은 100입니다.
Visible	Boolean	숨겨진 패널은 플롯 높이를 차지하지 않으며 그 축과 시리즈는 그려지지 않습니다. 기본값은 True입니다.

TsgcChartPanes 메서드

메서드	반환 값	설명
Add	TsgcChartPane	스택의 맨 아래에 패널을 추가합니다.
Find(const aName: string)	TsgcChartPane	이 Name을 가진 첫 번째 패널입니다(대소문자 구분 안 함). 없으면 nil입니다.

TsgcChart 속성과 메서드

멤버	형식	설명
Panes	TsgcChartPanes(속성)	비어 있으면(기본값) 하나의 플롯 영역을 의미합니다. 비어 있지 않은 컬렉션은 컬렉션 순서대로 위에서 아래로 표시되는 패널들 사이에 플롯을 분할합니다.
HasPanes	Boolean(메서드)	마지막 **Render**가 플롯을 패널로 분할했으면 True입니다.
PaneRect(aIndex: Integer)	TsgcUIRect(메서드)	마지막 **Render**에서 이 컬렉션 인덱스에 있는 패널의 사각형입니다. 패널이 숨겨져 있으면 비어 있습니다.
AxisPlotRect(aAxis: TsgcChartAxis)	TsgcUIRect(메서드)	마지막 **Render**에서 aAxis가 사용한 플롯 사각형입니다. 차트에 패널이 있으면 수직 축의 패널이고(패널이 숨겨져 있으면 비어 있음), 그렇지 않으면 전체 **PlotRect**입니다.
SeriesPlotRect(aSeries: TsgcChartSeries)	TsgcUIRect(메서드)	aSeries가 속한 패널, 즉 그 수직 축의 패널에 대한 플롯 사각형입니다.

SplitterAt(X, Y: Double)	Integer(메서드)	X, Y(차트 픽셀, 표시되는 두 패널 사이의 간격) 아래에 있는 분할선 위쪽 패널의 인덱스입니다. 지점이 분할선 위에 있지 않으면 -1입니다.
MoveSplitter(aIndex: Integer; Y: Double)	(메서드)	aIndex에 있는 패널 아래의 분할선을 Y(차트 픽셀)로 이동합니다. 해당 패널과 다음으로 표시되는 패널의 **HeightPercent**가 변경되며, 그 합은 그대로 유지됩니다.

축과 시리즈를 패널에 바인딩하기

TsgcChartAxis.Pane(문자열, 기본값은 비어 있음)은 수직 축이 속한 패널의 이름을 지정합니다. **Pane**이 비어 있으면 첫 번째로 표시되는 패널로 결정됩니다. 수평 축은 모든 패널이 동일한 수평 축을 공유하므로 **Pane**을 완전히 무시합니다. 시리즈는 자신만의 **Pane** 속성을 가지고 있지 않습니다. 시리즈는 자신의 **YAxis**가 속한 패널에 그려지므로, 시리즈를 어느 패널에 배치할지는 전적으로 그 **AxisY**가 지정하는 수직 축과 그 축 자체의 **Pane**에 달려 있습니다. 가격 패널과 그 아래에 RSI 패널이 있는 차트는 일반적으로 세 개의 축, 즉 공유되는 하나의 아래쪽 축과 패널당 하나의 왼쪽 축을 가지며, 각 왼쪽 축의 **Pane**은 해당 패널의 **Name**으로 설정됩니다.

분할선 드래그하기

패널은 자동으로 크기가 조정되지 않습니다. 사용자가 두 패널 사이의 간격을 드래그할 수 있도록 하는 호스트는 마우스 버튼을 누를 때 **SplitterAt**을 호출하여 포인터가 분할선 위에 있는지(그리고 어느 패널이 그 위에 있는지) 확인한 다음, 드래그하는 동안 마우스가 움직일 때마다 그 패널 인덱스로 **MoveSplitter**를 호출하며 새 Y값을 차트 픽셀 단위로 전달합니다. **MoveSplitter**는 분할선 양쪽에 있는 두 패널의 결합된 **HeightPercent**를 일정하게 유지하므로, 스택의 더 아래쪽에 있는 패널은 영향을 받지 않습니다. 이는 TsgcChartInteraction이 VCL과 FMX의 마우스 이벤트로부터 분할선 드래그를 처리하는 방식과 정확히 같습니다. 뷰 없이 렌더링하고 마우스 입력을 전혀 받지 않는 코드는 이 두 메서드 중 어느 것도 호출할 필요가 없습니다.

예제: 아래쪽 축을 공유하는 가격 패널과 RSI 패널

참고 항목

- TsgcChartSeries_Indicator
- TsgcChartSeries_Oscillator
- TsgcChartSeries_Volume
- TsgcChartInteraction
- 축
- TsgcChart
- 에디션

TsgcChartAggregator

Unit: sgcCharts_Aggregator
Registered 및 Trial 에디션에서만 지원됩니다(Community 에디션에는 포함되지 않습니다. 에디션 참조).
TsgcChartAggregator는 체결(trade) 스트림으로부터 **Series**(캔들스틱 또는 OHLC 시리즈)의 캔들을 구성하는 `TComponent`입니다. 모든 체결에 대해 **AddTrade(aTime, aPrice, aQty)**를 호출하며, aTime은 UTC `TDateTime`입니다. 각 캔들은 하나의 **Interval** 버킷을 포함하며 그 X는 그 버킷의 시작(오픈 시간, Binance kline과 동일한 관례)입니다. 형성 중인 캔들의 버킷 안에 있는 체결은 그 캔들의 고가, 저가, 종가, 거래량을 그 자리에서 갱신합니다. 이후 버킷에 있는 체결은 그 캔들을 마감하고 자신의 버킷 시작 시점에 새 캔들을 열어 추가합니다. 이때 **FillGaps**가 설정되어 있으면 그 사이의 모든 빈 버킷에 대해 평평한 캔들을 선택적으로 삽입합니다. 형성 중인 캔들보다 오래된 체결은 무시되며 **DroppedCount**에 집계됩니다. 생성 직후 또는 **Reset** 이후의 첫 번째 체결은, 그 캔들이 같은 버킷에 속하면(실시간 체결이 시작되기 전에 이미 이력이 로드된 경우) **Series**의 최신 캔들을 이어가고, 최신 캔들이 이후 버킷에 속하면 버려집니다.
버킷은 UTC를 기준으로 정렬된 정수 밀리초 단위로 계산됩니다. 초, 분, 시는 시계에 맞춰 정렬되고, 일은 00:00에, 주는 월요일 00:00에, 월은 달력상 첫날에 정렬됩니다. **SessionAlign**은 UTC 00:00에 시작하지 않는 세션을 위해 모든 경계를 일정 분만큼 이동시킵니다.

속성

속성	형식	설명
Series	TsgcChartSeries_OHLCBase	캔들을 받는 캔들 시리즈입니다. 이를 변경하거나 시리즈가 해제되면 **Reset**과 마찬가지로 형성 중인 캔들을 잊어버립니다.
Interval	TsgcChartAggregatorInterval	버킷 크기입니다: `cai1s`, `cai5s`, `cai15s`, `cai30s`, `cai1m`, `cai3m`, `cai5m`, `cai15m`, `cai30m`, `cai1h`, `cai2h`, `cai4h`, `cai6h`, `cai12h`, `cai1d`, `cai1w` 또는 `cai1Month`. 기본값은 `cai1m` 입니다. 변경하면 형성 중인 캔들을 잊어버립니다.
SessionAlign	Integer	모든 UTC 경계에 더해지는 분 단위 값이며, 0(기본값)은 순수한 UTC 경계를 의미합니다. 변경하면 형성 중인 캔들을 잊어버립니다.
FillGaps	Boolean	두 체결 사이의 모든 빈 버킷에 대해 평평한 캔들(open = high = low = close = 이전 종가, 거래량 0)을 삽입합니다. 기본값은 False입니다.
DroppedCount	Integer(읽기 전용)	형성 중인 캔들의 버킷보다 오래되어 무시된 체결의 개수입니다.

메서드

메서드	반환 값	설명
AddTrade(aTime: TDateTime; aPrice, aQty: Double)		체결 하나를 공급합니다. **AddOHLC**와 **UpdateLastOHLC**와 같은 방식으로 **Series**를 갱신하므로, 그것들과 동일한 스레드에서 호출해야 합니다.
Reset		형성 중인 캔들을 잊고 **DroppedCount**를 지웁니다. **Series** 자체는 변경되지 않습니다.
BucketStart(aTime: TDateTime)	TDateTime	**Interval**과 **SessionAlign**을 반영하여, aTime이 속하는 버킷의 시작 시각입니다.

애그리게이터에 데이터 공급하기

Binance 체결은 TsgcChartDataSource_Binance에 의해 네 단계로 집계됩니다. 차트에 캔들스틱 또는 OHLC 시리즈를 배치합니다(자신만의 패널에 있는 TsgcChartSeries_Volume은 캔들의 거래량을 표시할 수 있습니다). 그 시리즈를 **Series**에 할당하고 **Interval**을 선택합니다. 데이터 소스의 `Stream`을 `cbsTrade`로 설정합니다. 애그리게이터를 그 `Aggregator` 속성에 할당합니다. 그러면 모든 체결이 시간, 가격, 수량과 함께 **AddTrade**에 전달되어, 캔들이 체결 거래량을 포함하게 되고 체결은 선으로 표시되지 않습니다. 다른 어떤 피드라도, **Series**가 그 밖의 방식으로 갱신되는 스레드이기만 하다면, 체결을 생성하는 스레드에서 직접 **AddTrade**를 호출할 수 있습니다.

예제: 체결을 1분 캔들로 집계

참고 항목

- TsgcChartSeries_Indicator
- TsgcChartDrawing
- 패널
- 브리지 개요
- TsgcChartDataSource_Binance
- TsgcChartSeries_Candlestick
- 에디션

TsgcChartDrawing

Unit: sgcCharts_Drawing
Registered 및 Trial 에디션에서만 지원됩니다(Community 에디션에는 포함되지 않습니다. 에디션 참조).
TsgcChart.Drawings는 **TsgcChartDrawing** 항목들, 즉 추세선, 수평선과 수직선, 사각형, 피보나치 되돌림을 보유하며, 이들은 **AxisX**와 **AxisY**로 지정된 축의 데이터 값(**X1**, **Y1**, **X2**, **Y2**)에 고정되므로, 드로잉은 고정된 픽셀 위치에 머무르는 대신 확대/축소, 이동, 실시간 데이터를 따라갑니다. **TsgcChart.RenderContent**는 시리즈와 축 다음에 드로잉을 그리며, 해당 수직 축의 패널로 클리핑됩니다. 드로잉과 함께, 시리즈는 자신만의 가격 마커인 **TsgcChartSeries.LastValueMarker**와 **TsgcChartSeries.BidAskBand**를 가지며, 둘 다 `TPersistent`이고 이 페이지에서 함께 문서화되어 있습니다.

TsgcChartDrawing 속성

속성	형식	설명
Kind	TsgcChartDrawingKind	`cdkTrendLine` (두 앵커 사이의 선분), `cdkHorizontalLine` (Y1 위치에서 패널을 가로지르는 선), `cdkVerticalLine` (X1 위치에서 패널을 가로지르는 선), `cdkRectangle` (두 앵커로 이루어진 상자, 외곽선과 옅은 채우기) 또는 `cdkFibonacci` (X1과 X2 사이의 되돌림 레벨). 기본값은 `cdkTrendLine` 입니다.
X1, Y1, X2, Y2	Double	**AxisX**(X)와 **AxisY**(Y)의 데이터 값으로서의 앵커입니다. `cdkFibonacci` 의 경우 0%는 Y2에, 100%는 Y1에 위치합니다(스윙의 시작에서 끝까지 드래그). X1이 X2와 같으면 레벨은 패널 전체 너비에 걸쳐 표시됩니다.
AxisX	string	비어 있으면 시리즈와 동일한 관례로 첫 번째 아래쪽 축을 의미합니다.
AxisY	string	비어 있으면 첫 번째 왼쪽 축을 의미하며, 드로잉은 이 축의 패널에 그려집니다. 패널을 참조하십시오.
Color	TsgcUIColor	0(기본값)은 차트의 첫 번째 팔레트 색상을 의미합니다.
LineWidth	Double	스케일 1에서의 디바이스 픽셀입니다. 기본값은 1.5입니다.
Dash	TsgcChartDrawingDash	`cddSolid` , `cddDash` 또는 `cddDot` . 기본값은 `cddSolid` 입니다.
Visible	Boolean	기본값은 True입니다.
Locked	Boolean	잠긴 드로잉은 선택은 가능하지만 대화형으로 이동하거나 삭제할 수 없습니다. 기본값은 False입니다.

TsgcChartDrawing 메서드

메서드	반환 값	설명
SetAnchors(const aX1, aY1, aX2, aY2: Double)		속성마다 하나씩이 아니라 하나의 변경 알림으로 네 개의 앵커를 모두 설정합니다.

TsgcChartDrawings (TsgcChart.Drawings)

TsgcChartDrawings는 **TsgcChart.Drawings**의 배경이 되는 `TOwnedCollection`입니다.

멤버	반환 값	설명
Add	TsgcChartDrawing	드로잉을 추가합니다.
AnchorPixels(aIndex: Integer; out aX1, aY1, aX2, aY2: Double; out aPlot: TsgcUIRect)	Boolean	마지막 **Render** 기준으로 aIndex에 있는 드로잉의 앵커를 차트 픽셀 단위로 반환합니다. 드로잉이 그려진 패널 사각형도 함께 반환합니다.
PixelToValue(aIndex: Integer; X, Y: Double; out aX, aY: Double)	Boolean	마지막 **Render** 기준으로, aIndex에 있는 드로잉의 축 위에서 차트 픽셀 X, Y에 해당하는 데이터 값입니다.
HitTest(X, Y: Double; out aHandle: Integer)	Integer	X, Y(차트 픽셀) 아래에 있는 드로잉의 인덱스이며, 맨 위에 있는 것부터 우선합니다. 없으면 -1입니다. aHandle은 그 지점 아래에 있는 앵커로 0 또는 1이며, 본체에 히트한 경우 -1입니다.
DeleteSelected	Boolean	**Locked**가 아닌 한 **Selected**를 제거합니다. 제거되었으면 True입니다.
Draw(aCanvas: TsgcUICanvas; aScale: Double)		표시되는 모든 드로잉을 그립니다. **TsgcChart.RenderContent**가 호출합니다.
DrawSelection(aCanvas: TsgcUICanvas; aScale: Double)		**Selected**의 핸들을 그립니다. 상호작용 오버레이의 일부입니다.
SaveToJSON	string	"drawings" 배열을 가진 객체이며, 드로잉마다 하나의 객체를 가집니다: "kind"(trendline, hline, vline, rectangle 또는 fibonacci), "x1", "y1", "x2", "y2", "axisX", "axisY", "color"(부호 없는 숫자로 표현된 ARGB), "lineWidth", "dash"(solid, dash 또는 dot), "visible", "locked"입니다. 숫자는 로케일에 관계없이 '.'을 사용하며 double의 모든 자릿수를 유지합니다.
LoadFromJSON(const aJSON: string)		드로잉을 교체합니다(루트 배열도 허용됩니다). 알 수 없는 kind의 항목은 건너뛰며, 잘못된 형식의 JSON은 **EsgcChartJSONError**를 발생시킵니다.
Selected	Integer(속성)	선택된 드로잉의 인덱스입니다. 없으면 -1입니다.

대화형 드로잉과 선택

TsgcChartInteraction.DrawingTool은 마우스로 드로잉을 만듭니다. 누르면 첫 번째 앵커가 설정되고, 드래그하면 두 번째 앵커가 설정되며, 놓으면 완료되고 도구는 `cdtNone`으로 돌아갑니다. 활성화된 도구가 없을 때, 클릭하면 포인터 아래의 드로잉이 선택되고(**Drawings.Selected**), 본체를 드래그하면 이동하며, 핸들을 드래그하면 해당 앵커 하나만 이동합니다. VCL과 FMX 뷰에서 Delete 키에 바인딩된 **TsgcChartInteraction.DeleteSelectedDrawing**은 **DeleteSelected**를 통해 이를 제거합니다. 마우스와 키보드 연결에 대해서는 TsgcChartInteraction을 참조하십시오. 뷰가 없는 코드는 대신 **Add**, **HitTest**, **Selected**를 직접 호출합니다.

TsgcChartLastValueMarker

시리즈의 마지막 값 위치에서 패널을 가로지르는 점선이며, 그 값에 대한 레이블 상자가 플롯 쪽의 수직 축 위에 표시됩니다.

속성	형식	설명
Visible	Boolean	기본값은 False입니다.
Color	TsgcUIColor	0(기본값)은 시리즈 색상을 의미하며, 캔들 시리즈의 경우 마지막 캔들의 상승 또는 하락 색상입니다.

TsgcChartBidAskBand

코드에서 설정하는 Bid와 Ask이며, 두 가격 사이의 옅은 밴드로 그려지고, 각 가격에 선이, 수직 축 위에 각각의 레이블 상자가 표시됩니다.

속성	형식	설명
Visible	Boolean	기본값은 False입니다.
Bid, Ask	Double	**SetPrices**를 사용하여 함께 설정하면 하나의 변경 알림만 발생시킵니다.
BidColor, AskColor	TsgcUIColor	기본값은 각각 녹색(bid)과 빨간색(ask) 강조 색상입니다.

SetPrices(const aBid, aAsk: Double)는 두 번이 아니라 하나의 알림으로 두 가격을 모두 설정합니다.
두 마커 모두 드로잉과 마찬가지로 X/Y 시리즈(그 마지막 Y)와 캔들 시리즈(그 마지막 종가)에 대해 그려집니다. 시리즈는 같은 차트의 다른 시리즈와 무관하게 둘 중 하나, 둘 다, 또는 어느 것도 표시하지 않을 수 있습니다.

예제: 추세선, 마지막 값 마커, Bid/Ask 밴드

참고 항목

- TsgcChartSeries_Indicator
- TsgcChartAggregator
- 패널
- TsgcChartInteraction
- TsgcChartSeries_Candlestick
- TsgcChartSeries
- 에디션

TsgcChartBuffer and TsgcChartOHLCBuffer

단위: sgcCharts_Buffer
차트 시리즈를 위한 링 버퍼입니다. **TsgcChartBuffer**는 X/Y 점을 보관하고, **TsgcChartOHLCBuffer**는 X, Open, High, Low, Close, Volume을 보관합니다. X는 `Double`입니다(`TDateTime`은 `Double`로 저장됩니다). Append는 O(1)입니다. `cbmOverwrite` 모드에서는 버퍼가 가득 차면 다음 추가 시 가장 오래된 점을 삭제하고, `cbmGrow` 모드에서는 용량이 두 배로 늘어납니다.
모든 공개 메서드는 버퍼의 잠금을 사용하므로, 피드 스레드가 데이터를 추가하는 동안 렌더링 스레드가 읽을 수 있습니다. 렌더링 스레드는 **Snapshot**이나 **SnapshotRange**를 호출해야 합니다. 이들은 잠금 하에서 데이터를 호출자가 소유한 레코드로 복사한 뒤 즉시 잠금을 해제하며, 이후 그 복사본으로부터 디시메이션과 그리기를 수행합니다. Decimation을 참조하십시오.
X는 단조 비감소(monotonic non-decreasing)로 예상됩니다. Append는 **Monotonic** 플래그를 유지합니다. 이전보다 낮은 X를 가진 점이 도착하는 즉시 False가 되며, **Clear**만이 이를 리셋합니다. **Monotonic**이 True이면, **SnapshotRange**는 이진 탐색으로 표시 범위를 찾고 각 가장자리 바깥의 가장 가까운 점을 추가하므로, 플롯을 벗어나는 선도 그려집니다. False이면 **SnapshotRange**는 범위 안의 점만 복사하는 선형 스캔으로 대체되며, 디시메이션 함수는 데이터를 디시메이션 없이 반환합니다.

공유 속성 및 메서드(TsgcChartBufferBase)

멤버	설명
Capacity: Integer	점 또는 캔들 단위의 현재 용량입니다. 늘리면 재할당되고, **Count** 미만으로 줄이면 가장 오래된 데이터가 버려집니다.
Count: Integer	현재 보유하고 있는 점 또는 캔들 수입니다.
Mode: TsgcChartBufferMode	`cbmOverwrite` (가득 차면 가장 오래된 점 삭제) 또는 `cbmGrow` (가득 차면 용량을 두 배로 늘림)입니다.
Monotonic: Boolean	추가된 모든 X가 이전 X 이상이었던 동안 True입니다. 위를 참조하십시오.
Lock / Unlock	호출이 하나 이상 원자적이어야 하는 호출자를 위해 버퍼 자체의 잠금을 획득하거나 해제합니다.
Clear	버퍼를 비우고 **Monotonic**을 True로 리셋합니다.

TsgcChartBuffer

멤버	설명
Create(aCapacity: Integer = 1024; aMode: TsgcChartBufferMode = cbmOverwrite)	생성자입니다.
Append(const aX, aY: Double)	점 하나를 추가합니다.
AppendBatch(const aX, aY: array of Double)	하나의 잠긴 호출로 여러 점을 추가합니다.
First(out aX, aY: Double): Boolean	가장 오래된 점입니다. 버퍼가 비어 있으면 False입니다.
Last(out aX, aY: Double): Boolean	가장 최근 점입니다. 버퍼가 비어 있으면 False입니다.

Snapshot(var aSnap: TsgcChartBufferSnapshot)	모든 점을 `aSnap`으로 복사합니다. 배열은 호출 사이에 재사용되며 `aSnap.Count`보다 길 수 있습니다.
SnapshotRange(const aXMin, aXMax: Double; var aSnap: TsgcChartBufferSnapshot)	[aXMin, aXMax]에서 보이는 점을 복사하며, **Monotonic**인 경우 위에서 설명한 가장자리 점도 함께 복사합니다.
X[aIndex], Y[aIndex]: Double	논리 순서입니다: 0이 가장 오래된 점이고, `Count - 1`이 가장 최근 점입니다.

TsgcChartOHLCBuffer

멤버	설명
Create(aCapacity: Integer = 1024; aMode: TsgcChartBufferMode = cbmOverwrite)	생성자입니다.
Append(const aX, aOpen, aHigh, aLow, aClose: Double; const aVolume: Double = 0)	캔들 하나를 추가합니다.
AppendBatch(const aX, aOpen, aHigh, aLow, aClose, aVolume: array of Double)	하나의 잠긴 호출로 여러 캔들을 추가합니다. `aVolume`은 비어 있을 수 있으며, 그 경우 거래량은 0입니다.
First / Last(out aX, aOpen, aHigh, aLow, aClose, aVolume: Double): Boolean	가장 오래된 / 가장 최근 캔들입니다. 버퍼가 비어 있으면 False입니다.
UpdateLast(const aX, aOpen, aHigh, aLow, aClose, aVolume: Double): Boolean	아직 형성 중인 실시간 캔들을 위해, X가 `aX`와 같을 때 가장 최근 캔들의 값을 교체합니다. 버퍼가 비어 있거나 가장 최근 캔들이 다른 X를 가지면 False를 반환하고 아무것도 변경하지 않습니다.
Snapshot(var aSnap: TsgcChartOHLCSnapshot)	모든 캔들을 `aSnap`으로 복사합니다.
SnapshotRange(const aXMin, aXMax: Double; var aSnap: TsgcChartOHLCSnapshot)	[aXMin, aXMax]에서 보이는 캔들을 복사합니다.
X[aIndex], Open[aIndex], High[aIndex], Low[aIndex], Close[aIndex], Volume[aIndex]: Double	논리 순서입니다: 0이 가장 오래된 캔들이고, `Count - 1`이 가장 최근 캔들입니다.

예제: 추가하고 스냅숏 가져오기

참고 항목

- Decimation
- TsgcChartDataSource

Decimation

단위: sgcCharts_Buffer
버퍼 스냅숏을 `[aXMin, aXMax]`를 표시하는 `aPixelWidth` 픽셀 너비의 플롯에 대해 픽셀 열마다 대략 한 세트의 점으로 줄이는, 프레임워크에 의존하지 않는 함수들입니다. 비단조 스냅숏, 빈 범위, 또는 `aPixelWidth < 1`인 경우에는 아래의 모든 경우에서 디시메이션 없이 그대로 복사됩니다.

값 / 함수	설명
TsgcChartDecimation = (dmNone, dmMinMax, dmLTTB)	`dmNone` 은 모든 점을 복사합니다. `dmMinMax` 는 X를 픽셀 열로 나누고, 각 열마다 X 순서로 첫 번째, 최소, 최대, 마지막 점(최대 4개)을 유지하여 꺾은선이 시각적으로 정확하게 유지되도록 합니다. `aXMin` 왼쪽의 점들은 하나의 추가 열을 공유하고, `aXMax` 오른쪽의 점들은 또 다른 추가 열을 공유합니다. `dmLTTB` 는 정적 시리즈를 위해 Largest Triangle Three Buckets로 선택된 `2 × aPixelWidth` 개의 점을 반환합니다.
sgcChartDecimate(const aSnap: TsgcChartBufferSnapshot; const aXMin, aXMax: Double; aPixelWidth: Integer; aMode: TsgcChartDecimation; var aOutX, aOutY: TsgcChartDoubleArray; out aOutCount: Integer)	`aMode` 를 사용하여 점 스냅숏을 디시메이션합니다. `aOutX` 와 `aOutY` 는 너무 짧을 때만 늘어나며, 줄어들지는 않습니다.
sgcChartLTTB(const aX, aY: TsgcChartDoubleArray; aCount, aThreshold: Integer; var aOutX, aOutY: TsgcChartDoubleArray; out aOutCount: Integer)	X/Y 배열 쌍에 직접 적용하는 Largest Triangle Three Buckets입니다. 정확히 `Min(aThreshold, aCount)` 개의 점을 반환하며, 첫 번째와 마지막 점은 항상 포함됩니다.
sgcChartDecimateOHLC(const aSnap: TsgcChartOHLCSnapshot; const aXMin, aXMax: Double; aPixelWidth: Integer; var aOut: TsgcChartOHLCSnapshot)	같은 픽셀 열에 속하는 캔들들을 병합합니다: 첫 캔들의 X와 시가, 가장 높은 고가, 가장 낮은 저가, 마지막 캔들의 종가, 거래량의 합입니다. `aOut.Count` 가 그 결과를 담습니다.

예제: 800px 플롯을 위한 스냅숏 디시메이션

참고 항목

- TsgcChartBuffer and TsgcChartOHLCBuffer

TsgcUICanvas

단 위: sgcUICore_Canvas, 두 개 의 백 엔 드 와 함 께: sgcUICore_Canvas_Raster(**TsgcUICanvas_Raster**)와 sgcUICore_Canvas_SVG(**TsgcUICanvas_SVG**)
TsgcUICanvas는 TsgcChart.**Render**가 그리는 추상 드로잉 표면입니다. 핵심 패키지는 두 개의 구체적인 백엔드를 제공합니다: **TsgcUICanvas_Raster**는 메모리 내 ARGB 버퍼에 그려 PNG로 저장하고, **TsgcUICanvas_SVG**는 SVG 문서를 만듭니다. 둘 다 동일한 경로, 페인트, 텍스트 API를 공유하므로 차트는 둘 중 어느 쪽으로 렌더링해도 동일하게 표시됩니다. 애플리케이션은 보통 캔버스를 만들고, **Render**를 호출하고, 결과를 저장하는 것만 합니다. 아래의 그리기 기본 요소들은 사용자 정의 오버레이를 추가할 때 사용할 수 있도록 공개되어 있습니다.

형식

형식	설명
TsgcUIColor = Cardinal	직선(사전 곱셈되지 않은) `$AARRGGBB` 입니다.
TsgcUIRect	`Left`, `Top`, `Right`, `Bottom` : Double, 장치 픽셀입니다. Right / Bottom은 바깥쪽 가장자리입니다.
TsgcUIFontStyle	`Size` : Double, `Bold` : Boolean, `Color` : TsgcUIColor입니다.
TsgcUILineJoin, TsgcUILineCap	`(ljMiter, ljRound, ljBevel)`, `(lcButt, lcRound, lcSquare)` 입니다.
TsgcUITextHAlign, TsgcUITextVAlign	`(thaLeft, thaCenter, thaRight)`, `(tvaTop, tvaMiddle, tvaBaseline, tvaBottom)` 입니다.
TsgcUIFillRule	`frNonZero`, `frEvenOdd` 입니다.

메서드

메서드	설명
Clear(aColor: TsgcUIColor)	표면 전체를 채웁니다.
BeginPath, MoveTo, LineTo, QuadTo, CubicTo, Arc, RoundRect, Rectangle, ClosePath	경로 구성입니다. **Arc**는 라디안 단위의 각도를 받으며, +X에서 0이고 화면상 시계 방향이 양의 스윕입니다.
FillPath(aColor: TsgcUIColor; aRule: TsgcUIFillRule = frNonZero)	현재 경로를 채웁니다. 다음 **BeginPath**까지 유지됩니다.
FillPathLinearGradient(aX1, aY1, aX2, aY2: Double; const aStops: array of TsgcUIGradientStop; aRule: TsgcUIFillRule = frNonZero)	현재 경로를 선형 그라디언트로 채웁니다.
StrokePath(aColor, aWidth[, aJoin, aCap[, aDash, aDashOffset]])	두 개의 오버로드: 대시 패턴이 있는 것과 없는 것입니다.
DrawText, MeasureText, TextWidth, LineHeight, FitText	**TsgcUIFontStyle**을 사용한 텍스트 그리기와 측정입니다. Fonts를 참조하십시오.
DrawImage(const aImage: TsgcUIImage; aX, aY: Integer; aOpacity: Double = 1)	ARGB 이미지를 블릿합니다.

PushClip(aLeft, aTop, aRight, aBottom: Integer), PopClip	장치 픽셀 단위의 클립 사각형이며, Right / Bottom은 제외됩니다. 중첩된 클립은 서로 교차됩니다.
SetTransform(aA, aB, aC, aD, aE, aF: Double), ResetTransform	아핀 변환입니다: `x' = A*x + C*y + E`, `y' = B*x + D*y + F`.
SetFontFile(const aRegular, aBold: string), GetFont(aBold: Boolean)	텍스트를 측정하고 그리는 데 사용되는 글꼴입니다. 빈 파일 이름은 내장된 Roboto를 선택합니다. Fonts를 참조하십시오.
Width, Height	장치 픽셀 단위의 표면 크기입니다.

TsgcUICanvas_Raster 추가 사항

멤버	설명
Create(aWidth, aHeight: Integer)	자체 래스터라이저를 할당하는 오버로드입니다.
Create(aRasterizer: TsgcUIRasterizer)	기존 래스터라이저에 그리는 오버로드입니다.
SetSize(aWidth, aHeight: Integer)	표면 크기를 조정합니다.
SaveToPNGStream(aStream: TStream; aCompress: Boolean = True)	표면을 PNG로 인코딩합니다.
SaveToPNGFile(const aFileName: string; aCompress: Boolean = True)	PNG를 인코딩하여 디스크에 씁니다.
Rasterizer	TsgcUIRasterizer(읽기 전용), 기반이 되는 래스터라이저입니다.

TsgcUICanvas_SVG 추가 사항

멤버	설명
Create(aWidth, aHeight: Integer)	지정된 크기의 빈 문서를 만듭니다.
SetSize(aWidth, aHeight: Integer)	지정된 크기의 새 빈 문서입니다.
SaveToString	string, XML 선언을 포함한 전체 문서입니다. 열려 있는 클립은 출력에서 닫힙니다.
SaveToStream(aStream: TStream), SaveToFile(const aFileName: string)	문서를 씁니다.

차트를 PNG나 SVG로 렌더링하기

목표 크기로 캔버스를 만들고, 이를 덮는 **TsgcUIRect**로 **TsgcChart.Render**를 호출한 뒤 저장합니다. 이것이 VCL이나 FireMonkey 컨트롤이 전혀 관여하지 않는, 헤드리스 보고서나 예약된 내보내기에 필요한 전체 코드 경로입니다.

예제: PNG와 SVG로 각각 렌더링된 동일한 차트

참고 항목

- TsgcChart
- Fonts

TsgcUIFont

단위: sgcUICore_Font
TsgcUIFont는 모든 TsgcUICanvas가 텍스트를 측정하고 그리는 데 사용하는 순수 Pascal TrueType 텍스트 엔진입니다. 운영체제의 텍스트 스택에 의존하지 않고 글꼴 파일 자체를 직접 파싱합니다. 모든 캔버스는 **SetFontFile** / **GetFont**를 통해 자체 정체와 굵게 **TsgcUIFont**를 만들므로, 캔버스 밖에서 텍스트를 측정하는 경우가 아니라면 애플리케이션이 이를 직접 만드는 일은 드뭅니다.

속성

속성	형식	설명
Loaded	Boolean(읽기 전용)	글꼴 파일이나 스트림이 파싱된 뒤 True가 됩니다.
UnitsPerEm, Ascender, Descender, LineGap	Integer(읽기 전용)	글꼴 단위의 글꼴 메트릭입니다.
GlyphCount	Integer(읽기 전용)	
CmapFormat	Integer(읽기 전용)	파일에서 발견된 문자-글리프 테이블 형식입니다.
HasKerning	Boolean(읽기 전용)	불러온 글꼴 파일 자체에 kern 테이블이 있을 때만 True입니다.
Kerning	Boolean	기본값은 True입니다. 글꼴에 kern 테이블이 있으면 측정과 그리기 중에 이를 적용합니다. 테이블을 가진 글꼴이라도 커닝을 끄려면 False로 설정하십시오.
CacheCapacity, CacheCount	Integer	내부 글리프 윤곽선 캐시의 크기와 현재 항목 수입니다.

메서드

메서드	반환값	설명
LoadFromFile(const aFileName: string)		TrueType(glyf 윤곽선) 글꼴 파일을 파싱합니다. CFF(PostScript 윤곽선) 글꼴이나 사용할 수 없는 다른 글꼴에 대해서는 **EsgcUIFont**를 발생시킵니다.
LoadFromStream(aStream: TStream)		동일하지만 스트림에서 읽습니다.
LoadDefault(aBold: Boolean = False)		sgcCharts에 내장된 글꼴(Roboto Regular 또는 Bold)을 불러옵니다. 항상 성공하며 파일이 필요하지 않습니다.
LoadPlatformDefault(aBold: Boolean = False)		현재 플랫폼에 대해 최선을 다해 찾는 시스템 UI 글꼴입니다(Windows에서는 Segoe UI, Android에서는 Roboto, Linux에서는 DejaVu Sans나 Liberation Sans, macOS/iOS에서는 Helvetica나 Arial). 이 파일들 중 어느 것도 찾지 못하면

		LoadDefault로 대체되므로 절대 예외를 발생시키지 않습니다.
CharToGlyph(aCode: Cardinal)	Integer	유니코드 코드 포인트를 글리프 id로 변환합니다. 매핑되지 않으면 0(.notdef)입니다.
GlyphAdvance(aGlyph: Integer)	Integer	글꼴 단위의 전진 폭입니다.
GetKerning(aLeft, aRight: Integer)	Integer	글리프 쌍에 대한 글꼴 단위의 kern 테이블 보정값입니다. 쌍이 없으면 0입니다.
IsCompositeGlyph(aGlyph: Integer)	Boolean	
GetGlyphOutline(aGlyph: Integer)	TsgcUIGlyphOutline	글꼴 단위의 캐시된 윤곽선이며, 글꼴이 소유하고 이를 무효화할 수 있는 다음 호출까지 유효합니다. 잘못된 글리프 id에는 nil입니다.
MeasureText(const aText: string; aSize: Double; out aWidth, aAscent, aDescent: Double)		
TextWidth(const aText: string; aSize: Double)	Double	
LineHeight(aSize: Double)	Double	
FitText(const aText: string; aSize, aMaxWidth: Double)	string	**aText**를 **aMaxWidth**에 맞게 잘라냅니다.
DrawText(aRaster: TsgcUIRasterizer; const aText: string; aX, aY: Double; aSize: Double; aColor: Cardinal; aAngle: Double = 0)		**TsgcUICanvas** 수준보다 아래에서 래스터라이저에 직접 그립니다.
TextToCodePoints(const aText: string) (class function)	TsgcUICodePointArray	

내장 글꼴과 플랫폼 글꼴

sgcCharts는 모든 차트가 외부 파일 없이 텍스트를 그릴 수 있도록 Roboto의 TrueType 빌드를 내장합니다. **LoadDefault**는 이를 직접 불러오며, 이는 **TsgcUICanvas.SetFontFile**에 빈 파일 이름을 전달했을 때 선택되는 글꼴이기도 합니다. **LoadPlatformDefault**는 먼저 플랫폼 자체의 UI 글꼴을 시도하고, 사용 가능한 파일을 찾거나 파싱할 수 없을 때만 내장 글꼴로 대체하므로, 호스트 운영체제와 일치시키고자 하는 애플리케이션에 더 안전한 기본값입니다. 내장된 Roboto 빌드에는 kern 테이블이 없으므로, **Kerning**은 True로 유지되더라도 **HasKerning**은 이에 대해 False를 보고합니다. 자체 kern 테이블을 가지고 불러온 사용자 정의 TrueType 글꼴은 자동으로 커닝됩니다.

TrueType 윤곽선만 지원됩니다

TsgcUIFont는 `glyf` 윤곽선 테이블을 직접 읽으며, CFF(PostScript 윤곽선) 파서는 없습니다. macOS의 Helvetica나 SF처럼 일반 TrueType 파일처럼 보이는 많은 시스템 글꼴이 실제로는 CFF이며, 그런 경우 **LoadFromFile** / **LoadFromStream**은 **EsgcUIFont**를 발생시킵니다. 플랫폼 기본 글꼴이 CFF로 밝혀지면 **LoadPlatformDefault**가 이를 포착하여 다음 후보로 넘어가며, 사용 가능한 것이 없으면 결국 내장 글꼴에 도달합니다.

예제: 내장 글꼴, 플랫폼 글꼴, 텍스트 측정

참고 항목

- Canvas

지표

단위: sgcCharts_Indicators *(재무 에디션 전용, `SGC_CHARTS_FINANCIAL` 아래에서 컴파일됨. 그렇지 않으면 빈 껍데기)*
프레임워크에 의존하지 않는 점진적 기술 지표 계산기입니다: **TsgcChartIndicatorSMA**, **EMA**, **WMA**, **Bollinger**, **RSI**, **MACD**, **Stochastic**, **ATR**, **VWAP**이며, 모두 **TsgcChartIndicatorCalc**에서 파생됩니다. TsgcChart에 대한 의존성이 전혀 없으며, 추가 시리즈, 신호 엔진, 또는 단순 보고 코드에 데이터를 공급할 수 있습니다.

공통 패턴

멤버	설명
Add(...)	새 바 하나를 공급하고 최신 출력을 반환합니다. Stochastic을 제외하면 O(1) 상각이며, Stochastic은 고가/저가 윈도우를 스캔합니다(O(KPeriod)).
UpdateLast(...)	아직 형성 중인 실시간 캔들을 위해, 가장 최근에 추가된 바의 입력을 대체합니다. 마지막 **Add** 이전에 저장된 상태가 복원되고 새 값이 다시 추가되므로, 결과는 수정된 바로 처음부터 다시 계산한 것과 일치합니다. 계산기가 비어 있으면 **Add**처럼 동작합니다.
Calculate(const aInput: array of Double; ...)	계산기를 리셋하고 배열 전체를 **Add**로 공급하므로, 배치 결과와 점진적 결과가 일치합니다. 계산기는 이후에도 상태를 유지하므로 이력을 불러온 뒤 실시간 바를 추가할 수 있습니다.
Reset	모든 상태를 지웁니다. **Count**는 0으로 돌아갑니다.
Count	Integer(읽기 전용). 마지막 **Reset** 이후 추가된 바 수입니다.

워밍업

계산기의 워밍업 기간이 끝날 때까지 모든 출력은 `NaN`입니다(`Math.NaN`, `Math.IsNan`으로 테스트). `NaN` 입력은 무시되므로 윈도우에 들어가지 않으며, 해당 바의 출력도 `NaN`입니다. 아래의 바 인덱스는 0부터 시작합니다.

정의

계산기	생성자	정의
TsgcChartIndicatorSMA	Create(aPeriod: Integer = 20)	마지막 **Period**개 값의 산술 평균입니다. 첫 값은 바 Period-1에서 나타납니다.
TsgcChartIndicatorEMA	Create(aPeriod: Integer = 20)	alpha = 2/(Period+1), 첫 Period개 값의 SMA로 바 Period-1에서 시드된 뒤, Value := X * alpha + Prev * (1 - alpha)입니다.
TsgcChartIndicatorWMA	Create(aPeriod: Integer = 20)	선형 가중 이동 평균입니다. 가중치는 1..Period(가장 최근 값에 Period)이며 Period(Period+1)/2로 나눕니다. 첫 값은 바 Period-1에서 나타납니다.
TsgcChartIndicatorBollinger	Create(aPeriod: Integer = 20; const aDeviations: Double = 2)	Middle = SMA(Period), 모집단 표준편차(Period로 나눔), Upper / Lower = Middle +/- Deviations * StdDev입니다. 첫 값은 바 Period-1에서 나타납니다.
TsgcChartIndicatorRSI	Create(aPeriod: Integer = 14)	Wilder 평활화입니다. Change(i) = Close(i) - Close(i-1)입니다. 첫 평균 상승/하락은 첫 Period개 변화량의 평균(바 Period)이며, 이후 Avg :=

		(Avg * (Period-1) + Current) / Period입니다. RSI = 100 - 100 / (1 + AvgGain / AvgLoss)입니다. AvgLoss가 0이면 100, 둘 다 0이면 50입니다.
TsgcChartIndicatorMACD	Create(aFastPeriod: Integer = 12; aSlowPeriod: Integer = 26; aSignalPeriod: Integer = 9)	Line = EMA(Fast) - EMA(Slow)이며, 첫 Line은 바 Slow-1에서 나타납니다. Signal은 Line의 EMA(Signal)이며, 첫 Signal개 Line 값의 SMA로 시드되고 첫 값은 바 Slow+Signal-2에서 나타납니다. Histogram = Line - Signal입니다.
TsgcChartIndicatorStochastic	Create(aKPeriod: Integer = 14; aKSmoothing: Integer = 3; aDPeriod: Integer = 3)	Raw = 100 * (Close - LowestLow) / (HighestHigh - LowestLow)이며, 최근 KPeriod개의 고가/저가에 대해 계산됩니다(HighestHigh = LowestLow이면 50). 첫 값은 바 KPeriod-1에서 나타납니다. K는 Raw의 SMA(KSmoothing) (KSmoothing = 1이면 빠른 스토캐스틱), D는 K의 SMA(DPeriod)입니다.
TsgcChartIndicatorATR	Create(aPeriod: Integer = 14)	Wilder 방식입니다. TR(i) = Max(High-Low, |High-PrevClose|, |Low-PrevClose|)이며 바 1부터 계산됩니다. 첫 ATR은 첫 Period개 TR의 평균(바 Period)이며, 이후 ATR := (ATR * (Period-1) + TR) / Period입니다.
TsgcChartIndicatorVWAP	Create(const aSessionStart: TDateTime = 0)	누적 Sum(TypicalPrice * Volume) / Sum(Volume)이며, TypicalPrice = (High+Low+Close)/3입니다. 합계는 매 세션의 첫 바에서 다시 시작됩니다. 바의 세션은 Floor(Time - SessionStart)이므로, SessionStart = 0이면 매 달력일마다, 9.5/24이면 09:30마다 리셋됩니다. 워밍업이 없으며, 세션 거래량이 0인 동안은 NaN입니다.

예제: SMA와 RSI, 스트리밍 및 배치

참고 항목

- TsgcChartSeries

.NET Overview

sgcCharts .NET은 Delphi sgcCharts 라이브러리를 감싼 래퍼가 아니라 관리형 C# 포트입니다. 각 Delphi 유닛은 **esegece.sgcCharts** 네임스페이스 아래에 동일한 **Tsgc*** 클래스 이름을 사용하는 C# 대응 클래스를 가지므로, Delphi API를 대상으로 작성된 코드가 C#에서도 같은 방식으로 읽힙니다. 이 포트는 프로젝트가 대상으로 하는 플랫폼만 가져오도록 여섯 개의 NuGet 패키지로 나뉘어 있습니다.

패키지

PackageId	포함 내용	대상 프레임워크
esegece.sgcCharts	핵심 차트 모델: **TsgcChart**, 축, 패널, 시리즈, 데이터 소스, 버퍼, 드로잉, 수학/서식 및 JSON 헬퍼, SVG 캔버스입니다. 네이티브 의존성이 없습니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcCharts.Rendering	Windows, Linux, macOS에서 헤드리스 PNG 출력에 사용되는 SkiaSharp 래스터 캔버스인 **TsgcUICanvas_Raster**입니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcCharts.WinForms	Windows Forms 뷰인 **TsgcWinFormsChart**입니다.	net472, net8.0-windows, net9.0-windows
esegece.sgcCharts.Wpf	WPF 뷰인 **TsgcWpfChart**입니다.	net472, net8.0-windows, net9.0-windows
esegece.sgcCharts.Avalonia	크로스 플랫폼 뷰(Windows, Linux, macOS)인 **TsgcAvaloniaChart**입니다.	net8.0, net9.0
esegece.sgcCharts.sgcWebSockets	**sgcWebSockets** 연결(WebSocket 메시지, MQTT 발행, server-sent events, Binance 시장 스트림)에서 데이터를 공급받는 차트 데이터 소스인 실시간 브리지입니다. **esegece.sgcWebSockets** NuGet 패키지에 의존합니다.	netstandard2.0, net8.0, net9.0

Community 및 Registered 에디션

위 각 패키지의 기본 빌드(스위치 없음)는 유료 에디션인 **Registered**입니다: `SGC_COMMUNITY` 상수도, 내장 로고도, 시작 알림도 없습니다. `-p:SgcCommunity=true`로 빌드하거나 패키징하면 대신 **Community**(무료) 에디션이 만들어지며, 브리지(**esegece.sgcCharts.sgcWebSockets**)를 제외한 모든 패키지의 PackageId에 **.Community** 접미사가 붙습니다. 브리지는 Community 에디션에서는 전혀 제공되지 않으며, 이는 Delphi Community 에디션이 자체 브리지 유닛에 적용하는 것과 같은 규칙입니다. Community 빌드는 여덟 개 시리즈 전부(캔들스틱과 OHLC 포함)와 모든 뷰를 그대로 유지하며, 재무 팩(패널, 지표 및 거래량 시리즈, 애그리게이터, 그리기 도구, 최근값 마커, 매수/매도 밴드)만 제외합니다.
Community는 소규모 팀의 상업적, 비상업적 사용 모두에 대해 무료입니다: 연간 총매출 100만 유로 미만이고 개발자 5명 이하인 개인 및 단체가 대상입니다. 둘 중 하나라도 그 기준 이상인 단체는 상업용(Registered) 라이선스가 필요합니다.

API는 Delphi를 그대로 반영합니다

뷰 클래스는 Delphi의 명명 방식을 유지합니다: **TsgcWinFormsChart**와 **TsgcWpfChart**는 Delphi의 **TsgcVCLChart**에 대응하는 .NET 클래스이고, **TsgcAvaloniaChart**는 Delphi의 **TsgcFMXChart**에 대응합니다.
TsgcChart.OnChange, **TsgcChartDataSource.OnDataChanged**, **TsgcChartDataSource.OnBeforeAppend**, **TsgcChart.OnCrosshairMove**, **OnSeriesClick**, **OnZoom**과 같은 핵심 모델의 알림들은 C# `event` 멤버가 아니라 단순한

`get; set;` 델리게이트 속성입니다. 새 핸들러를 지정하면 Delphi의 단일 메서드 포인터와 정확히 동일하게 이전 핸들러를 대체합니다. 실제 C# 멀티캐스트 이벤트(`+=` / `-=`)가 관여하는 유일한 곳은 **esegece.sgcCharts.sgcWebSockets** 브리지가 .NET sgcWebSockets 포트의 클라이언트를 구독하는 부분입니다. Bridge Overview와 .NET Differences from Delphi를 참조하십시오.
대부분의 Delphi 인덱싱된 배열 속성은 실제 C# 인덱서가 됩니다: **TsgcChartAxes**, **TsgcChartPanes**, **TsgcChartDrawings**, **TsgcChartPalette**는 모두 `this[int Index]`를 노출하며, **TsgcChartJSONNode**는 배열 요소에 대해 `this[int aIndex]`를 노출합니다. 시리즈 뒤의 점 버퍼는 예외입니다: **TsgcChartBuffer.X(Index)** / **Y(Index)**, **TsgcChartOHLCBuffer.X(Index)** / **Open(Index)** / **High(Index)** / **Low(Index)** / **Close(Index)** / **Volume(Index)**, 그리고 재무 시리즈의 **XValues(Index)** / **Values(Line, Index)**는 일반 메서드입니다. C#은 클래스당 인덱서 시그니처를 하나만 허용하는데, 이 클래스들 각각은 같은 클래스에서 독립적으로 이름 붙여진, 또는 다중 매개변수를 가진 여러 인덱스 접근자가 필요하기 때문입니다. .NET Differences from Delphi를 참조하십시오.

폐기 가능한 뷰와 캔버스

컨트롤 크기에 맞춰진 영속적인 네이티브 Skia 서피스나 GDI 비트맵을 보유하는 클래스는 `IDisposable`을 구현하며 반드시 폐기해야 합니다. 그렇지 않으면 네이티브 메모리가 누수됩니다:

- **TsgcUICanvas_Raster**(esegece.sgcCharts.Rendering): SkiaSharp 비트맵, 캔버스, 페인트, 서체를 소유합니다.
- **TsgcChartRenderer**: 자신의 래스터 캔버스를 소유합니다.
- **TsgcWinFormsChart**: 재정의된 `Dispose(bool)`에서 래스터 캔버스, Skia 비트맵, GDI+ 비트맵, 그리고 제한 타이머를 폐기합니다.
- **TsgcWpfChart**: 래스터 캔버스와 Skia 비트맵을 폐기합니다.
- **TsgcAvaloniaChart**의 내부 프레임(**TsgcAvaloniaChartFrame**): 전면 및 후면 Skia 이미지와 래스터 캔버스를 폐기합니다.

핵심 차트 모델 자체(**TsgcChart**, 시리즈, 데이터 소스)는 네이티브 리소스를 전혀 보유하지 않으므로 그 이유로 폐기가 필요하지는 않습니다. 다만 시리즈를 소스에서, 또는 데이터 소스를 클라이언트에서 분리하는 등 Delphi 방식의 해제 알림을 실행하기 위해 일부 위치에서 여전히 `IDisposable`을 구현합니다.

Delphi와 동일한 텍스트 측정값

모든 캔버스는 Delphi `sgcUICore_Font.pas`의 METRICS 부분을 포팅한 TrueType 리더인 **TsgcUIFont**로 텍스트를 측정합니다. 이는 문자-글리프 조회를 위해 `cmap` 포맷 4 또는 12 테이블을, 전진 폭을 위해 `hmtx`를, 커닝 쌍을 위해 `kern` 포맷 0 서브테이블을, 그리고 어센더, 디센더, 라인 갭을 위해 `hhea`를 읽습니다. Delphi 라이브러리가 내장하는 것과 동일한 **Roboto-Regular.ttf**와 **Roboto-Bold.ttf** 파일(Apache License 2.0)을 불러오므로, .NET에서 계산된 레이아웃은 마지막 비트까지 Delphi 레이아웃과 일치합니다. SkiaSharp은 측정 리더가 이미 배치한 글리프 윤곽선을 래스터화하는 데에만 사용되며, 문자당 하나의 배치된 글리프를 안티앨리어싱과 서브픽셀 위치 지정으로(힌팅 없이) 그립니다. SkiaSharp 자체는 텍스트를 측정하지 않으며, 이 때문에 래스터와 SVG를 포함한 모든 백엔드가 Delphi와 일치된 상태를 유지합니다.

참고 항목

- .NET Differences from Delphi
- TsgcChart
- TsgcVCLChart
- Bridge Overview

Quick Start: WinForms

TsgcWinFormsChart는 TsgcChart 모델의 Windows Forms 뷰입니다. 자신의 **Chart** 속성에 지정된 차트를 그리는 `Control`의 하위 클래스입니다. `net472`, `net8.0-windows`, `net9.0-windows`를 대상으로 하는 **esegece.sgcCharts.WinForms** 패키지로 제공되며, **esegece.sgcCharts.Rendering**에 대한 `ProjectReference`(NuGet에서 사용할 때는 `PackageReference`)를 전이적으로 포함하므로, WinForms 프로젝트는 **esegece.sgcCharts.WinForms** 자체만 참조하면 됩니다. 이 컨트롤은 Delphi의 TsgcVCLChart와 동일한 Skia 래스터 캔버스로, 영속적인 GDI DIB 섹션에 직접 그려지며, **Chart.MaxFPS**로 제한됩니다. 백그라운드 스레드에서 데이터를 공급해도 안전합니다.

속성

속성	형식	설명
Chart	TsgcChart	이 컨트롤이 그리는 차트 모델입니다. 이 값을 지정하면 컨트롤이 리스너로 등록됩니다. 값을 지우거나 차트를 폐기하면 컨트롤이 더 이상 아무것도 그리지 않습니다.
FrameCount	uint(읽기 전용)	컨트롤이 생성된 이후 렌더링된 전체 프레임 수입니다.
OverlayFrameCount	uint(읽기 전용)	컨트롤이 생성된 이후 렌더링된 오버레이 전용 프레임 수입니다. 크로스헤어, 툴팁, 또는 러버 밴드 선택만 이동시키는 변경에 대한 것입니다.

일반적인 `Control` 속성들도 변경 없이 공개되어 있습니다: **Dock**, **Anchor**, **BackColor**, **Enabled**, **Visible**, **TabIndex**, **TabStop**, **ContextMenuStrip**, 그리고 마우스, 드래그, 크기 조정 이벤트입니다.

메서드

메서드	반환 값	설명
SaveToPNG(string aFileName)	void	컨트롤의 현재 클라이언트 크기와 DPI로 **Chart**를 렌더링하여 새 PNG 파일로 저장합니다. **Chart**가 null이면 아무 동작도 하지 않습니다.
SaveToSVG(string aFileName)	void	컨트롤의 현재 클라이언트 크기와 DPI로 **Chart**를 렌더링하여 SVG 파일로 저장합니다. **Chart**가 null이면 아무 동작도 하지 않습니다.

예제: WinForms 폼에 차트 그리기

```
using System;
using System.Windows.Forms;
using esegece.sgcCharts;

public class Form1 : Form
{
    private readonly TsgcChart FChart;
    private readonly TsgcWinFormsChart FView;
    private readonly TsgcChartSeries_Line FSeries;

    public Form1()
    {
        Text = "sgcCharts WinForms Sample";
        Width = 820;
        Height = 480;
```

```
        FChart = new TsgcChart();
        TsgcChartAxis oBottom = FChart.Axes.Add();
        oBottom.Name = "bottom";
        oBottom.Position = TsgcChartAxisPosition.capBottom;
        TsgcChartAxis oLeft = FChart.Axes.Add();
        oLeft.Name = "left";
        oLeft.Position = TsgcChartAxisPosition.capLeft;

        FSeries = new TsgcChartSeries_Line();
        FSeries.Chart = FChart;
        FSeries.AxisX = "bottom";
        FSeries.AxisY = "left";
        for (int i = 0; i <= 99; i++)
            FSeries.AddXY(i, Math.Sin(i / 10.0) * 50 + 50);

        FView = new TsgcWinFormsChart();
        FView.Dock = DockStyle.Fill;
        FView.Chart = FChart;
        Controls.Add(FView);
    }

    // elsewhere, e.g. from a menu action or before closing:
    private void SaveChart()
    {
        FView.SaveToPNG("chart.png");
    }
}
```

참고 항목

- .NET Overview
- Quick Start: WPF
- TsgcVCLChart
- TsgcChartSeries_Line

Quick Start: WPF

TsgcWpfChart는 TsgcChart 모델의 WPF 뷰입니다. 자신의 **Chart** 종속성 속성에 지정된 차트를 그리는 `Control`의 하위 클래스입니다. `net472`, `net8.0-windows`, `net9.0-windows`를 대상으로 하는 **esegece.sgcCharts.Wpf** 패키지로 제공되며, **esegece.sgcCharts.Rendering**에 대한 `ProjectReference`(NuGet에서 사용할 때는 `PackageReference`)를 전이적으로 포함하므로, WPF 프로젝트는 **esegece.sgcCharts.Wpf** 자체만 참조하면 됩니다. 이 컨트롤은 Delphi의 TsgcVCLChart와 동일한 Skia 래스터 캔버스로, 영속적인 `WriteableBitmap`의 백 버퍼에 직접 그려지며, **Chart.MaxFPS**로 제한됩니다. 백그라운드 스레드에서 데이터를 공급해도 안전합니다.

속성

속성	형식	설명
Chart	TsgcChart	이 컨트롤이 그리는 차트 모델이며, **ChartProperty** 종속성 속성이 뒷받침합니다. 이 값을 지정하면 컨트롤이 로드되어 있는 동안 리스너로 등록됩니다. 값을 지우거나 차트를 폐기하면 컨트롤이 더 이상 아무것도 그리지 않습니다.
FrameCount	long(읽기 전용)	컨트롤이 생성된 이후 렌더링된 전체 프레임 수입니다.
OverlayFrameCount	long(읽기 전용)	컨트롤이 생성된 이후 렌더링된 오버레이 전용 프레임 수입니다. 크로스헤어, 툴팁, 또는 러버 밴드 선택만 이동시키는 변경에 대한 것입니다.

일반적인 `Control` 속성들도 변경 없이 공개되어 있습니다: **Width**, **Height**, **HorizontalAlignment**, **VerticalAlignment**, **Background**, **IsEnabled**, **Visibility**, **Focusable**, 그리고 마우스와 키보드 라우팅 이벤트입니다.

메서드

메서드	반환 값	설명
SaveToPNG(string aFileName)	void	컨트롤의 현재 렌더링 크기와 DPI로 **Chart**를 렌더링하여 새 PNG 파일로 저장합니다. **Chart**가 null이면 아무 동작도 하지 않습니다.
SaveToSVG(string aFileName)	void	컨트롤의 현재 렌더링 크기와 DPI로 **Chart**를 렌더링하여 SVG 파일로 저장합니다. **Chart**가 null이면 아무 동작도 하지 않습니다.

예제: WPF 창에 차트 그리기

```
using System;
using System.Windows;
using System.Windows.Controls;
using esegece.sgcCharts;

public class MainWindow : Window
{
    private readonly TsgcChart FChart;
    private readonly TsgcWpfChart FView;
    private readonly TsgcChartSeries_Line FSeries;

    public MainWindow()
    {
        Title = "sgcCharts WPF Sample";
        Width = 820;
        Height = 480;
```

```
        FChart = new TsgcChart();
        TsgcChartAxis oBottom = FChart.Axes.Add();
        oBottom.Name = "bottom";
        oBottom.Position = TsgcChartAxisPosition.capBottom;
        TsgcChartAxis oLeft = FChart.Axes.Add();
        oLeft.Name = "left";
        oLeft.Position = TsgcChartAxisPosition.capLeft;

        FSeries = new TsgcChartSeries_Line();
        FSeries.Chart = FChart;
        FSeries.AxisX = "bottom";
        FSeries.AxisY = "left";
        for (int i = 0; i <= 99; i++)
            FSeries.AddXY(i, Math.Sin(i / 10.0) * 50 + 50);

        FView = new TsgcWpfChart();
        FView.Chart = FChart;

        Content = new Grid { Children = { FView } };
    }

    // elsewhere, e.g. from a menu action or before closing:
    private void SaveChart()
    {
        FView.SaveToPNG("chart.png");
    }
}
```

참고 항목

- .NET Overview
- Quick Start: WinForms
- TsgcVCLChart
- TsgcChartSeries_Line

Quick Start: Avalonia

패키지: esegece.sgcCharts.Avalonia(대상: net8.0, net9.0, **esegece.sgcCharts.Rendering**에 의존)
TsgcAvaloniaChart는 **TsgcChart** 모델의 Avalonia 뷰입니다. Windows, Linux, macOS에서 자신의 **Chart** 속성에 지정된 차트를 그리는 Avalonia `Control`의 하위 클래스입니다. Delphi의 TsgcFMXChart에 대응하는 .NET 클래스입니다. 컨트롤은 시각적 트리에 연결되어 있는 동안 차트에 리스너를 등록하고 모델이 변경될 때마다 스스로 다시 그리며, SkiaSharp으로 영속적인 프레임 비트맵에 그린 뒤 Avalonia 렌더러의 Skia 캔버스 리스를 통해 표시합니다. 모델 자체는 그리기 코드를 전혀 가지고 있지 않으므로, 같은 차트를 동시에 WinForms 뷰나 WPF 뷰로 표시하거나 헤드리스로 렌더링할 수도 있습니다.

속성

속성	형식	설명
Chart	TsgcChart	이 컨트롤이 그리는 차트 모델입니다. 이 값을 지정하면 컨트롤이 창에 연결되어 있는 동안 리스너로 등록됩니다. 값을 지우거나 차트를 폐기하면 컨트롤이 더 이상 아무것도 그리지 않습니다.
FrameCount	uint(읽기 전용)	컨트롤이 생성된 이후 렌더링된 전체 프레임 수입니다.
OverlayFrameCount	uint(읽기 전용)	컨트롤이 생성된 이후 렌더링된 오버레이 전용 프레임 수입니다(전체 다시 그리기 대신 마지막 캐시된 내용을 재사용하는 크로스헤어, 툴팁 또는 러버 밴드 변경).

일반적인 Avalonia `Control` 속성들도 변경 없이 사용할 수 있습니다: **Width**, **Height**, **Margin**, **HorizontalAlignment**, **VerticalAlignment**, **Focusable**, 그리고 포인터, 휠, 키 이벤트입니다.

메서드

메서드	반환 값	설명
SaveToPNG(string aFileName)		컨트롤의 현재 크기와 DPI로 **Chart**를 렌더링하여 새 PNG 파일로 저장합니다. **Chart**가 null이면 아무 동작도 하지 않습니다.
SaveToSVG(string aFileName)		컨트롤의 현재 크기와 DPI로 **Chart**를 렌더링하여 SVG 파일로 저장합니다. **Chart**가 null이면 아무 동작도 하지 않습니다.

프레임 배분은 다른 시각적 뷰와 동일하게 **Chart.MaxFPS**를 따릅니다: 변경 알림은 컨트롤을 더티로 표시하고 다시 그리기를 예약할 뿐이며, 초당 최대 **Chart.MaxFPS** 회로 제한됩니다. 따라서 백그라운드 스레드가 UI 스레드를 그리기 요청으로 넘치게 하지 않고 차트의 시리즈에 데이터를 공급할 수 있습니다.

예제

하나의 라인 시리즈를 가진 **TsgcChart**에 바인딩된 **TsgcAvaloniaChart**를 보여 주는 최소한의 Avalonia 애플리케이션이며, 전부 코드로만 작성됩니다(XAML 마크업 불필요). **esegece.sgcCharts.Avalonia** 외에, 다른 Avalonia 데스크톱 애플리케이션과 마찬가지로 **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** NuGet 패키지도 필요합니다.

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;
using esegece.sgcCharts;
```

```
namespace QuickStartAvalonia
{
    internal class TFRMMain : Window
    {
        public TFRMMain()
        {
            Title = "sgcCharts .NET Avalonia Quick Start";
            Width = 800;
            Height = 500;

            TsgcChart oChart = new TsgcChart();
            oChart.Title.Text = "Avalonia Quick Start";

            TsgcChartAxis oAxisX = oChart.Axes.Add();
            oAxisX.Name = "bottom";
            oAxisX.Position = TsgcChartAxisPosition.capBottom;

            TsgcChartAxis oAxisY = oChart.Axes.Add();
            oAxisY.Name = "left";
            oAxisY.Position = TsgcChartAxisPosition.capLeft;

            TsgcChartSeries_Line oSeries = new TsgcChartSeries_Line();
            oSeries.Chart = oChart;
            oSeries.AxisX = "bottom";
            oSeries.AxisY = "left";
            for (int i = 0; i <= 99; i++)
                oSeries.AddXY(i, Math.Sin(i / 10.0) * 50 + 50);

            TsgcAvaloniaChart oView = new TsgcAvaloniaChart();
            oView.Chart = oChart;
            Content = oView;
        }
    }

    internal class TQuickStartApp : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new TFRMMain();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main(string[] args)
        {
            BuildAvaloniaApp().StartWithClassicDesktopLifetime(args);
        }

        public static AppBuilder BuildAvaloniaApp()
        {
            return AppBuilder.Configure<TQuickStartApp>().UsePlatformDetect().LogToTrace();
        }
    }
}
```

참고 항목

- .NET Overview
- Quick Start: WinForms
- Quick Start: WPF
- TsgcFMXChart (Delphi)

헤드리스 렌더링

패키지: esegece.sgcCharts.Rendering(대상: netstandard2.0, net472, net6.0, net8.0, net9.0)
TsgcChartRenderer는 **TsgcChart** 모델의 헤드리스 뷰입니다. UI 프레임워크에 의존하지 않고 자체 창도 없는 단순한 클래스로서, 콘솔 프로그램, 서비스 또는 ASP.NET 애플리케이션에서 차트를 PNG 파일이나 스트림, SVG 문서 또는 직선 ARGB 픽셀 버퍼로 렌더링합니다. 디스플레이가 연결되지 않은 Linux 컨테이너나 CI 작업을 포함하여 Windows, Linux, macOS에서 동작하며, Delphi의 TsgcChartRenderer에 대응합니다. 시각적 표면이 없으므로 **Chart.MaxFPS**의 영향을 받지 않습니다. 프레임 루프나 타이머가 없으며, 각 렌더링 호출은 호출되는 즉시 동기적으로 차트를 그립니다.

속성

속성	형식	설명
Chart	TsgcChart	이 렌더러가 그리는 차트 모델입니다. 이 값이 null이면 모든 Render* 메서드가 **EsgcChartRendererError**를 발생시킵니다.
Width	int	크기를 지정하지 않는 오버로드에서 사용되는 기본 논리 너비(픽셀)입니다. 기본값은 800입니다.
Height	int	크기를 지정하지 않는 오버로드에서 사용되는 기본 논리 높이(픽셀)입니다. 기본값은 500입니다.
Scale	double	논리 픽셀당 장치 픽셀 수입니다. 기본값은 1입니다. 출력 크기는 Round(Width * Scale) x Round(Height * Scale) 장치 픽셀이 됩니다. **Chart.Scale**은 렌더링 동안 이 값으로 설정되었다가 이후 이전 값으로 복원됩니다.
CompressPNG	bool	Skia PNG 인코더를 통해 PNG 출력을 deflate로 압축할지 여부입니다. 기본값은 true입니다.

메서드

메서드	반환 값	설명
RenderToPNGFile(string aFileName) RenderToPNGFile(string aFileName, int aWidth, int aHeight)		**Width**/**Height** 크기로, 또는 **Width**와 **Height**를 변경하지 않고 지정된 논리 크기로 PNG 파일에 렌더링합니다.
RenderToPNGStream(Stream aStream) RenderToPNGStream(Stream aStream, int aWidth, int aHeight)		PNG(RGBA, 8비트)를 스트림에 렌더링합니다.
RenderToSVG() RenderToSVG(int aWidth, int aHeight)	string	렌더링하여 SVG 1.1 문서(순수 ASCII)를 문자열로 반환합니다.
RenderToSVGFile(string aFileName) RenderToSVGFile(string aFileName, int aWidth, int aHeight)		SVG를 파일에 렌더링합니다.
RenderToSVGStream(Stream aStream) RenderToSVGStream(Stream aStream, int aWidth, int aHeight)		SVG를 스트림에 렌더링합니다.
RenderToBuffer(ref uint[] aBuffer, out int aPixelWidth, out int aPixelHeight) RenderToBuffer(ref uint[] aBuffer, out int		직선(사전 곱셈되지 않은) 0xAARRGGBB 픽셀로, 위에서 아래 순서로, **aPixelWidth**와 같은 행 스트라이드로 렌

aPixelWidth, out int aPixelHeight, int aWidth, int aHeight)		더링합니다. **aPixelWidth**와 **aPixelHeight**는 장치 크기로 반환됩니다.

스레딩

모든 렌더링 메서드는 어느 스레드에서나 호출할 수 있습니다. 각 **TsgcChartRenderer**는 래스터용과 SVG용 각각 하나씩 자체 캔버스를 가지며, 이는 해당 렌더러 인스턴스 전용 잠금으로 보호됩니다. 이에 더해, **TsgcChart.Render**가 실행되는 동안 레이아웃 상태를 차트 자체에 보관하기 때문에, 프로세스 내 모든 **TsgcChartRenderer**의 모든 렌더링은 프로세스 전체가 공유하는 하나의 추가 잠금을 거칩니다. 렌더링이 진행되는 동안에는 차트를 구조적으로 변경해서는 안 됩니다. 즉 시리즈나 축을 추가, 제거, 폐기하거나 레이아웃에 영향을 주는 속성을 변경해서는 안 됩니다. 시리즈 버퍼를 통해 데이터를 추가하는 것은 안전합니다. 시리즈는 버퍼 자체의 잠금 하에 만들어진 스냅숏에서 그려지기 때문입니다. 시각적 뷰(**TsgcAvaloniaChart**, WinForms 뷰 또는 WPF 뷰)는 공유 렌더링 잠금을 사용하지 않으므로, 시각적 뷰가 같은 차트를 그리고 있는 순간에 다른 스레드에서 렌더러로 같은 차트를 렌더링하지 마십시오.

예제: Linux에서 PNG로 헤드리스 렌더링

UI 프레임워크 참조가 전혀 없는 콘솔 애플리케이션으로, Linux에서 빌드 및 실행할 수 있으며 차트를 PNG 파일로 렌더링합니다.

```
using System;
using esegece.sgcCharts;

namespace QuickStartHeadless
{
    internal static class Program
    {
        private static void Main(string[] args)
        {
            TsgcChart oChart = new TsgcChart();
            oChart.Title.Text = "Headless Render";

            TsgcChartAxis oAxisX = oChart.Axes.Add();
            oAxisX.Name = "bottom";
            oAxisX.Position = TsgcChartAxisPosition.capBottom;

            TsgcChartAxis oAxisY = oChart.Axes.Add();
            oAxisY.Name = "left";
            oAxisY.Position = TsgcChartAxisPosition.capLeft;

            TsgcChartSeries_Line oSeries = new TsgcChartSeries_Line();
            oSeries.Chart = oChart;
            oSeries.AxisX = "bottom";
            oSeries.AxisY = "left";
            for (int i = 0; i <= 99; i++)
                oSeries.AddXY(i, Math.Sin(i / 10.0) * 100);

            using (TsgcChartRenderer oRenderer = new TsgcChartRenderer())
            {
                oRenderer.Chart = oChart;
                oRenderer.Width = 800;
                oRenderer.Height = 400;
                oRenderer.RenderToPNGFile("chart.png");
            }

            oSeries.Dispose();
            oChart.Dispose();
        }
    }
}
```

이는 Linux 컨테이너나 CI 작업에서도 변경 없이 그대로 실행됩니다. **esegece.sgcCharts.Rendering**은 Linux용 SkiaSharp 네이티브 자산을 자체적으로 포함하므로, X 서버나 GPU, fontconfig 설치가 필요하지 않습니다.

예제: pie와 treemap의 헤드리스 렌더링

pie, radar, heatmap, treemap, funnel 시리즈(Series Gallery 참조)도 다른 시리즈와 마찬가지로 헤드리스로 렌더링됩니다. 렌더러는 마우스 오버 오버레이를 그리지 않으므로 출력에는 모델이 가진 상태만 표시됩니다. 뷰에서 클릭했을 때 남는 상태처럼 코드에서 설정하십시오. pie 조각에는 **Exploded**, treemap에는 **DrillPath**를 사용합니다. **Chart**가 바뀌면 하나의 렌더러가 여러 차트를 차례로 그립니다.

```
using System;
using esegece.sgcCharts;

namespace HeadlessSeries
{
    internal static class Program
    {
        private static void Main(string[] args)
        {
            // a pie with its largest slice exploded, as a click would leave it
            TsgcChart oPieChart = new TsgcChart();
            oPieChart.Title.Text = "Sales by region";
            TsgcChartSeries_Pie oPie = new TsgcChartSeries_Pie();
            oPie.Title = "Sales";
            oPie.Chart = oPieChart;
            oPie.AddItem("Europe", 342);
            oPie.AddItem("North America", 296);
            oPie.AddItem("Asia", 254);
            oPie.AddItem("Africa", 18);
            oPie.AddItem("Oceania", 14);
            oPie.OthersThreshold = 4;
            ((TsgcChartPieSlice)oPie.Slices[0]).Exploded = true;

            // a treemap drilled into one branch, as a click on a tile would leave it
            TsgcChart oTreeChart = new TsgcChart();
            oTreeChart.Title.Text = "Sales by country and city";
            oTreeChart.Legend.Visible = false;
            TsgcChartSeries_Treemap oTree = new TsgcChartSeries_Treemap();
            oTree.Title = "Sales";
            oTree.Chart = oTreeChart;
            oTree.AddItem("Europe/Spain/Madrid", 120);
            oTree.AddItem("Europe/Spain/Barcelona", 95);
            oTree.AddItem("Europe/France/Paris", 140);
            oTree.AddItem("Europe/France/Lyon", 55);
            oTree.AddItem("America/USA/New York", 210);
            oTree.AddItem("America/Canada/Toronto", 75);
            oTree.DrillPath = "Europe";

            using (TsgcChartRenderer oRenderer = new TsgcChartRenderer())
            {
                oRenderer.Width = 800;
                oRenderer.Height = 500;
                oRenderer.Chart = oPieChart;
                oRenderer.RenderToPNGFile("pie.png");
                oRenderer.RenderToSVGFile("pie.svg");
                oRenderer.Chart = oTreeChart;
                oRenderer.RenderToPNGFile("treemap.png");
                oRenderer.RenderToSVGFile("treemap.svg");
            }
            Console.WriteLine("pie.png, pie.svg, treemap.png and treemap.svg written");

            oPieChart.Dispose();
            oTreeChart.Dispose();
        }
    }
}
```

이 시리즈의 SVG는 같은 차트와 크기에 대해 Delphi **TsgcChartRenderer**가 쓰는 SVG와 바이트 단위로 동일합니다. **ImagePath**가 true인 heatmap은 셀을 하나의 이미지로 채우며, SVG는 이를 PNG로 포함합니다. `demos\Charts`의

SeriesHeadless 데모는 이 방식으로 시리즈 갤러리의 모든 차트를 렌더링합니다: `SeriesHeadless [folder] [-dark] [-width N] [-height N] [-scale N].`

참고 항목

- .NET Overview
- Quick Start: WinForms
- Series Gallery
- TsgcChartRenderer (Delphi)

Series Gallery

패키지: esegece.sgcCharts(대상: netstandard2.0, net472, net6.0, net8.0, net9.0)
pie, radar, heatmap, treemap, funnel 시리즈는 같은 이름의 Delphi 시리즈에 대응하는 .NET 버전입니다. 이들은 핵심 **esegece.sgcCharts** 패키지에 들어 있으므로 모든 뷰(**TsgcWinFormsChart**, **TsgcWpfChart**, **TsgcAvaloniaChart**)에서 그려지고, **TsgcChartRenderer**를 통한 헤드리스 렌더링도 지원합니다(Headless Rendering 참조). 속성, 기본값, 레이아웃은 Delphi와 같습니다. 아래에 링크된 각 시리즈의 Delphi 항목이 모든 속성을 설명합니다. 이 페이지는 C#에서 사용하는 방법을 보여 주고 API의 .NET 형태가 다른 부분을 정리합니다.

시리즈

클래스	플롯 모드	설명
TsgcChartSeries_Pie	cpmCell	원형 차트이며, **InnerRadiusPercent**가 0보다 크면 도넛이 됩니다. **OthersThreshold** 퍼센트 미만의 조각은 하나의 Others 조각으로 병합되고, **ExplodeOnClick**은 클릭 시 조각을 바깥으로 꺼내며, 레이블은 지시선과 함께 안쪽 또는 바깥쪽에 놓입니다. 항목: **Slices**, 각각 하나의 **TsgcChartPieSlice**.
TsgcChartSeries_Radar	cpmPolar	차트의 모든 radar 시리즈가 공유하는 축 위에 시리즈마다 하나의 다각형을 그립니다. **Filled**로 채우거나 윤곽선만 그리며, 격자는 다각형 또는 원형(**GridShape**)입니다. 항목: **Points**, 축마다 하나.
TsgcChartSeries_Heatmap	cpmXY	색상 스케일로 색을 입힌 셀의 격자입니다. **AppendLine**으로 한 줄씩 공급하는 실시간 스펙트로그램(**Capacity**가 최신 줄을 유지)이거나, 범주 축에 레이블이 있는 행과 열로 이루어진 고정 행렬입니다. 데이터: **Data**, 하나의 **TsgcChartHeatmapData**.
TsgcChartSeries_Treemap	cpmCell	`Europe/Spain/Madrid` 같은 경로(**PathDelimiter**)로 만드는 중첩 사각형이며, 그룹마다 헤더(**Levels**), 클릭 시 드릴다운(**DrillOnClick**), 위로 돌아가는 이동 경로(breadcrumb)를 제공합니다. 항목: **Nodes**, 잎마다 하나.
TsgcChartSeries_Funnel	cpmCell	컬렉션 순서대로 항목마다 하나의 단계가 있으며 목 부분(**NeckWidthPercent**, **NeckHeightPercent**)을 향해 좁아집니다. 단계의 높이는 같거나 **ProportionalHeight**로 값에 비례합니다. 항목: **Stages**.

플롯 모드에 따라 어떤 시리즈가 하나의 차트를 공유할 수 있는지가 정해집니다. Plot Areas를 참조하십시오. 한 차트에 pie와 funnel을 나란히 둘 수 있으며 각각 자신의 셀 영역을 사용하는 반면, heatmap은 XY 플롯과 그 축을 다른 XY 시리즈와 공유합니다.

항목과 실시간 값

멤버	설명
AddItem(string aText, double aValue, uint aColor = 0)	Pie, radar, treemap, funnel. 항목을 추가하고 그 인덱스를 반환합니다. 색이 0이면 항목 인덱스의 팔레트 색을 사용하므로 값이 바뀌어도 항목은 색을 유지합니다.
SetValueAt(int aIndex, double aValue) DeleteItem(int aIndex)	항목 하나를 변경하거나 제거합니다. 항목의 순서는 바뀌지 않습니다. 범위를 벗어난 인덱스는 `ArgumentOutOfRangeException`을 던집니다.
ItemCount()	항목 수이며, 포팅된 모든 매개변수 없는 Delphi 함수처럼 메서드입니다.

Slices, Points, Nodes, Stages	항목 컬렉션입니다. 인덱서는 **TsgcChartCategoryItem**(**Text**, **Value**, **Color**)을 반환하며, **Exploded**에 접근하려면 **TsgcChartPieSlice**로 캐스팅하십시오.
BeginUpdate() EndUpdate()	많은 변경을 한 번의 다시 그리기와 한 번의 레이아웃으로 묶습니다.
AppendLine(double[] aValues) AppendLine(double aKey, double[] aValues)	Heatmap. 셀 한 줄을 추가하며, 키는 **aKey** 또는 저장소의 실행 중인 줄 카운터입니다. 값은 배열이며 `params` 목록이 아닙니다.
SetLineLabels(string[] aLabels) SetCellLabels(string[] aLabels)	Heatmap. 범주 축에서 줄과 셀의 레이블입니다. 한 오버로드는 Delphi `TStrings` 의 .NET 형태인 **TsgcChartStrings**를 받으므로 단순한 `null` 은 모호합니다. 캐스팅하십시오.
Cells(int aLine, int aCell) SetCells(int aLine, int aCell, double aValue)	Heatmap. 셀 하나를 읽고 씁니다. Delphi 인덱스 속성 `Cells[Line, Cell]` 의 메서드 형태입니다.

상호 작용

뷰는 마우스를 **TsgcChart.Interaction**으로 전달하므로 마우스 오버 툴팁, explode, drill이 WinForms, WPF, Avalonia에서 동일하게 동작합니다. 시리즈를 클릭하면 먼저 **TsgcChart.OnSeriesClick**이 호출되고, 그 다음에 시리즈가 처리합니다.

멤버	설명
TsgcChartSeries_Pie.ExplodeOnClick	기본값은 false입니다. true이면 조각을 클릭할 때 해당 **Exploded** 플래그가 전환되며, Others 조각은 **OthersExploded**가 전환됩니다.
TsgcChartSeries_Treemap.DrillOnClick	기본값은 true입니다. 타일을 클릭하면 클릭한 위치의 그룹으로 한 단계 드릴다운하고, 이동 경로 항목을 클릭하면 그곳까지 다시 올라갑니다. **DrillDown(string aPath)**, **DrillUp()**, **DrillReset()**, **DrillPath**로 코드에서도 같은 동작을 할 수 있습니다.
TsgcChartSeries_Treemap.OnDrill	**DrillPath**가 바뀔 때 호출되는 **TsgcChartNotifyEvent**(`object Sender`)입니다. 델리게이트 속성이므로 핸들러를 할당하면 Delphi처럼 이전 핸들러가 대체됩니다.
TsgcChart.OnSeriesClick	**TsgcChartHitEvent**(`object Sender, TsgcChartHitInfo aHit`) 입니다. 이 시리즈에서 **aHit.SubIndex**는 항목 인덱스(pie의 Others 조각은 -2), **aHit.Text**는 조각 또는 단계의 텍스트나 treemap 경로, **aHit.Percent**는 pie, funnel, treemap에서 항목이 차지하는 비율입니다.

마우스 오버 오버레이(강조 표시와 툴팁)는 화면에만 그려집니다. **TsgcChartRenderer**, **SaveToPNG**, **SaveToSVG**는 모델만 렌더링하므로 꺼낸 조각이나 드릴다운한 treemap은 출력에 나타나지만 마우스를 올린 조각은 나타나지 않습니다.

예제: 다섯 시리즈 만들기

시리즈마다 차트 하나를 만드는 도우미 클래스이며, **SeriesGallery** 데모와 같은 차트를 더 적은 데이터로 만듭니다.

```
using esegece.sgcCharts;

namespace SeriesGallerySample
{
    public static class SeriesCharts
    {
        // pie: the regions under 4% are merged into Others, a click explodes a slice
        public static TsgcChart CreatePie(out TsgcChartSeries_Pie aPie)
        {
            TsgcChart vChart = new TsgcChart();
            vChart.Title.Text = "Sales by region";
```

```
    vChart.Legend.Position = TsgcChartLegendPosition.clpRight;
    aPie = new TsgcChartSeries_Pie();
    aPie.Title = "Sales";
    aPie.Chart = vChart;
    aPie.AddItem("Europe", 342);
    aPie.AddItem("North America", 296);
    aPie.AddItem("Asia", 254);
    aPie.AddItem("South America", 61);
    aPie.AddItem("Africa", 18);
    aPie.AddItem("Oceania", 14);
    aPie.OthersThreshold = 4;
    aPie.ExplodeOnClick = true;
    aPie.Labels = TsgcChartSliceLabels.cslOutside;
    aPie.InnerRadiusPercent = 55; // a donut, 0 draws a full pie
    return vChart;
}

// radar: two products scored on six criteria, one series per product
public static TsgcChart CreateRadar()
{
    string[] vCriteria = { "Price", "Quality", "Support", "Speed", "Security", "Docs" };
    double[][] vScores = { new double[] { 8, 6, 7, 9, 6, 5 }, new double[] { 6, 9, 8, 5, 8, 8 } };
    TsgcChart vChart = new TsgcChart();
    vChart.Title.Text = "Product comparison";
    vChart.Legend.Position = TsgcChartLegendPosition.clpBottom;
    for (int i = 0; i < vScores.Length; i++)
    {
        TsgcChartSeries_Radar oRadar = new TsgcChartSeries_Radar();
        oRadar.Title = i == 0 ? "Product A" : "Product B";
        oRadar.Chart = vChart;
        oRadar.Filled = true;
        oRadar.GridShape = TsgcChartRadarGrid.crgCircle;
        for (int j = 0; j < vCriteria.Length; j++)
            oRadar.AddItem(vCriteria[j], vScores[i][j]);
    }
    return vChart;
}

// heatmap: units by region (rows) and month (columns), values in the cells
public static TsgcChart CreateMatrix()
{
    string[] vRegions = { "North", "South", "East", "West" };
    string[] vMonths = { "Jan", "Feb", "Mar", "Apr", "May", "Jun" };
    TsgcChart vChart = new TsgcChart();
    vChart.Title.Text = "Units by region and month";
    vChart.Legend.Position = TsgcChartLegendPosition.clpRight;
    vChart.Axes[0].AxisType = TsgcChartAxisType.catCategory;
    vChart.Axes[1].AxisType = TsgcChartAxisType.catCategory;
    TsgcChartSeries_Heatmap oHeat = new TsgcChartSeries_Heatmap();
    oHeat.Title = "Units";
    oHeat.Chart = vChart;
    oHeat.Orientation = TsgcChartHeatmapOrientation.chmRows;
    oHeat.ShowValues = true;
    oHeat.BeginUpdate();
    try
    {
        for (int i = 0; i < vRegions.Length; i++)
        {
            double[] vLine = new double[vMonths.Length];
            for (int j = 0; j < vMonths.Length; j++)
                vLine[j] = 40 + (i * 37 + j * 23) % 60;
            oHeat.AppendLine(vLine);
        }
        oHeat.SetLineLabels(vRegions);
        oHeat.SetCellLabels(vMonths);
    }
    finally
```

```
            {
                oHeat.EndUpdate();
            }
            return vChart;
        }

        // treemap: the path levels are split on PathDelimiter ('/'); a click
        // on a tile drills down, a click on the breadcrumb drills back up
        public static TsgcChart CreateTreemap(out TsgcChartSeries_Treemap aTree)
        {
            TsgcChart vChart = new TsgcChart();
            vChart.Title.Text = "Sales by continent, country and city";
            vChart.Legend.Visible = false;
            aTree = new TsgcChartSeries_Treemap();
            aTree.Title = "Sales";
            aTree.Chart = vChart;
            aTree.BeginUpdate();
            try
            {
                aTree.AddItem("Europe/Spain/Madrid", 120);
                aTree.AddItem("Europe/Spain/Barcelona", 95);
                aTree.AddItem("Europe/France/Paris", 140);
                aTree.AddItem("Europe/France/Lyon", 55);
                aTree.AddItem("Europe/Germany/Berlin", 110);
                aTree.AddItem("America/USA/New York", 210);
                aTree.AddItem("America/USA/Chicago", 90);
                aTree.AddItem("America/Canada/Toronto", 75);
                aTree.AddItem("Asia/Japan/Tokyo", 160);
                aTree.AddItem("Asia/China/Shanghai", 130);
            }
            finally
            {
                aTree.EndUpdate();
            }
            return vChart;
        }

        // funnel: one stage per item, from the widest to the narrowest
        public static TsgcChart CreateFunnel(out TsgcChartSeries_Funnel aFunnel)
        {
            string[] vStages = { "Visitors", "Leads", "Qualified", "Proposals", "Won" };
            double[] vValues = { 12000, 5200, 2600, 1100, 310 };
            TsgcChart vChart = new TsgcChart();
            vChart.Title.Text = "Sales funnel";
            vChart.Legend.Position = TsgcChartLegendPosition.clpRight;
            aFunnel = new TsgcChartSeries_Funnel();
            aFunnel.Title = "Deals";
            aFunnel.Chart = vChart;
            for (int i = 0; i < vStages.Length; i++)
                aFunnel.AddItem(vStages[i], vValues[i]);
            return vChart;
        }
    }
}
```

예제: WinForms 폼에 차트 호스팅하기

탭마다 하나의 **TsgcWinFormsChart**, 클릭과 드릴 경로를 따라가는 캡션, 그리고 실시간 값입니다. WPF에서는 **TsgcWpfChart**, Avalonia에서는 **TsgcAvaloniaChart**를 사용하며 차트 코드는 바뀌지 않습니다.

```
using System;
using System.Windows.Forms;
using esegece.sgcCharts;

namespace SeriesGallerySample
```

```
{
    public class GalleryForm : Form
    {
        private readonly TsgcChart[] FCharts;
        private readonly TsgcWinFormsChart[] FViews;
        private readonly TsgcChartSeries_Pie FPie;
        private readonly TsgcChartSeries_Treemap FTree;
        private readonly TsgcChartSeries_Funnel FFunnel;
        private readonly Timer FTimer;
        private int FTicks;

        public GalleryForm()
        {
            Text = "sgcCharts series gallery";
            Width = 900;
            Height = 600;
            FCharts = new TsgcChart[]
            {
                SeriesCharts.CreatePie(out FPie), SeriesCharts.CreateRadar(), SeriesCharts.CreateMatrix(),
                SeriesCharts.CreateTreemap(out FTree), SeriesCharts.CreateFunnel(out FFunnel)
            };
            string[] vNames = { "Pie", "Radar", "Heatmap", "Treemap", "Funnel" };

            // a click calls OnSeriesClick first, then the series handles it
            // (ExplodeOnClick, DrillOnClick); both are delegate properties
            FCharts[0].OnSeriesClick = (Sender, aHit) => Text = "Clicked: " + aHit.Text;
            FTree.OnDrill = Sender => Text = "Treemap: " + (FTree.DrillPath == "" ? "All" : FTree.DrillPath);

            TabControl oTabs = new TabControl();
            oTabs.Dock = DockStyle.Fill;
            FViews = new TsgcWinFormsChart[FCharts.Length];
            for (int i = 0; i < FCharts.Length; i++)
            {
                FViews[i] = new TsgcWinFormsChart();
                FViews[i].Dock = DockStyle.Fill;
                FViews[i].Chart = FCharts[i];
                TabPage oPage = new TabPage(vNames[i]);
                oPage.Controls.Add(FViews[i]);
                oTabs.TabPages.Add(oPage);
            }
            Controls.Add(oTabs);

            Button oUp = new Button();
            oUp.Text = "Treemap up";
            oUp.Dock = DockStyle.Bottom;
            oUp.Click += (s, e) => FTree.DrillUp();
            Controls.Add(oUp);

            // live values: SetValueAt keeps the order and the colour of the items
            FTimer = new Timer();
            FTimer.Interval = 500;
            FTimer.Tick += (s, e) =>
            {
                FTicks++;
                FFunnel.SetValueAt(0, 12000 + 600 * Math.Sin(FTicks * 0.4));
                FPie.SetValueAt(2, 254 + 40 * Math.Cos(FTicks * 0.3));
            };
            FTimer.Start();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                FTimer.Dispose();
                foreach (TsgcWinFormsChart oView in FViews)
                    oView.Chart = null;
```

```
                foreach (TsgcChart oChart in FCharts)
                    oChart.Dispose();
            }
            base.Dispose(disposing);
        }

        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new GalleryForm());
        }
    }
}
```

데모

데모	프레임워크	설명
SeriesGallery	WinForms, net8.0-windows	pie, radar, 초당 30줄 스펙트로그램, 행렬 heatmap, treemap, funnel을 탭마다 하나의 **TsgcWinFormsChart**로 보여 줍니다. 도넛 전환, 채움 또는 원형 radar 옵션, treemap용 Up 및 Top 버튼이 있습니다.
SeriesGalleryAvalonia	Avalonia, net8.0	같은 갤러리를 **TsgcAvaloniaChart** 컨트롤로 구현한 것으로 Windows, Linux, macOS에서 동작합니다.
SeriesHeadless	Console, net8.0	갤러리 차트를 **TsgcChartRenderer**로 창 없이 PNG 및 SVG 파일로 렌더링합니다: `SeriesHeadless [folder] [-dark] [-width N] [-height N] [-scale N]` .

데모는 `demos\Charts`에 있습니다. 두 갤러리 모두 자체 테스트로 `-seconds N`을 받습니다. 모든 탭을 표시하고, 차트 상호 작용을 통해 pie 조각과 treemap 타일을 클릭한 뒤, N초 후에 닫히며 모든 검사를 통과하면 종료 코드 0을 반환합니다.

참고 항목

- .NET Overview
- .NET Differences from Delphi
- Headless Rendering
- Quick Start: WinForms
- Plot Areas
- Colour Scale

브리지

esegece.sgcCharts.sgcWebSockets 패키지(`netstandard2.0`, `net8.0`, `net9.0` 대상)는 sgcCharts 데이터 소스를 실시간 sgcWebSockets 연결에 연결합니다: WebSocket 메시지, MQTT 발행, Server-Sent Events, Binance 시장 스트림입니다. 이는 핵심 **esegece.sgcCharts** 패키지와 별도의 패키지이며 컴파일하려면 sgcWebSockets가 필요합니다. 핵심 데이터 소스만 사용하는 프로젝트는 이 패키지도 sgcWebSockets도 전혀 필요하지 않습니다.
기본적으로 이 패키지는 sgcWebSockets .NET 포트 프로젝트를 직접 참조합니다. 해당 소스 트리가 없는 빌드는 대신 **esegece.sgcWebSockets** NuGet 패키지를 참조할 수 있습니다. 이 브리지는 Community 에디션에는 포함되지 않습니다.

MQTT: TsgcChartDataSource_MQTT

TsgcWSProtocol_MQTT_Client가 수신한 MQTT 발행을 차트로 표시합니다. 소스가 **Active**이고 **Client**가 지정되어 있는 동안, 소스는 자신의 핸들러를 **Client.OnMQTTPublish**에 구독합니다.

속성	형식	설명
Client	TsgcWSProtocol_MQTT_Client	차트로 표시할 발행 메시지를 가진 MQTT 클라이언트입니다.
TopicFilter	string	와일드카드 `+` (한 단계)와 `#` (이 단계 및 그 아래 모든 단계)를 사용하는 MQTT 토픽 필터입니다. 필터가 비어 있으면 모든 발행이 차트로 표시됩니다.
PayloadFormat	TsgcChartMQTTPayloadFormat (cpfJSON, cpfNumber)	**cpfJSON**은 **Mapping**을 통해 페이로드를 읽습니다 (JSON 객체 또는 레코드 배열). **cpfNumber**는 페이로드를 Y 값의 일반 숫자로 파싱하고 X에는 현재 로컬 시각을 사용합니다.
Active	bool(기본 속성을 재정의)	이 소스는 **Active**를 다시 선언하여, 이를 설정하면 **Client**를 지정할 때와 동일하게 **Client**의 핸들러도 갱신되도록 합니다.

WebSocket: TsgcChartDataSource_WebSocket

TsgcWebSocketClient가 수신한 JSON 메시지를 차트로 표시합니다. 소스가 **Active**이고 **Client**가 지정되어 있는 동안, 소스는 자신의 핸들러를 **Client.OnMessage**에 구독합니다. 모든 텍스트 메시지는 **Mapping**을 통해 JSON으로 읽힙니다.

속성	형식	설명
Client	TsgcWebSocketClient	차트로 표시할 메시지를 가진 WebSocket 클라이언트입니다.
MessageFilter	string	메시지가 차트로 표시되려면 존재해야 하는 JSON 경로입니다. 예를 들어 Binance kline 메시지만 유지하려면 `"k"`를 지정합니다. 필터가 비어 있으면 JSON으로 파싱되는 모든 메시지가 차트로 표시됩니다.
Active	bool(기본 속성을 재정의)	**TsgcChartDataSource_MQTT**와 동일하게 재선언되어, 후크가 값을 따라갑니다.

Server-Sent Events: TsgcChartDataSource_SSE

서로 독립적인 두 소스로부터 Server-Sent Events를 차트로 표시하며, 이 둘은 동일한 데이터 소스에서 함께 사용할 수 있습니다.

- **Client**는 Delphi와 마찬가지로 **TsgcHTTP1Client**입니다. 애플리케이션은 보통 워커 스레드에서 클라이언트의 블로킹 `GetSSE(URL)`로 스트림을 시작합니다. 소스가 **Active**이고 **Client**가 지정되어 있는 동안, 소스는 자신의 핸들러를

Client.OnSSEMessage에 구독하며, 이는 도착하는 원시 스트림 텍스트를 그대로 전달합니다. 소스는 청크를 버퍼링하고, 빈 줄마다 이벤트로 분리하고, 한 이벤트의 `data:` 줄들을 결합한 뒤 그 결과를 **Mapping**을 통해 JSON으로 읽습니다.

- **SSEClient**는 관리형 EventSource 방식 클라이언트인 **TsgcSSEClient**입니다. 이 브리지에는 Delphi에 대응되는 것이 없습니다. **OnEvent**는 이미 파싱된 이벤트를 전달하므로, 소스는 청크 파싱 없이 이벤트의 **Data**를 **Mapping**을 통해 직접 JSON으로 읽습니다.

속성	형식	설명
Client	TsgcHTTP1Client	블로킹 SSE 클라이언트입니다. 위를 참조하십시오.
SSEClient	TsgcSSEClient	관리형 EventSource 클라이언트로, .NET에서 추가된 것입니다. 위를 참조하십시오.
Active	bool(기본 속성을 재정의)	**TsgcChartDataSource_MQTT**와 동일하게 재선언됩니다. **Client**와 **SSEClient** 모두 속성이 변경될 때 독립적으로 후크되거나 해제됩니다.

ResetStream 메서드는 **Client**의 청크 파서에 버퍼링되어 아직 완료되지 않은 이벤트를 폐기합니다. 예를 들어 재연결 이후에 사용합니다.

Binance: TsgcChartDataSource_Binance

TsgcWSAPI_Binance를 통해 Binance 공개 시장 데이터 스트림을 구독하고 차트로 표시합니다. Binance 시장 스트림에는 형식화된 이벤트가 없습니다. API가 연결된 WebSocket 클라이언트의 **OnMessage** 이벤트(**API.Client**)로 원시 JSON이 도착합니다. 소스가 **Active**이고 **API**가 지정되어 있는 동안, 소스는 그곳에 자신의 핸들러를 구독합니다. **API** 또는 **Active**가 설정될 때, 그리고 **Subscribe** 시점에도 후크는 **API.Client**를 따라갑니다.

속성	형식	설명
API	TsgcWSAPI_Binance	스트림을 차트로 표시할 Binance API 컴포넌트입니다.
Symbol	string	Binance 심볼입니다. 예를 들어 `"BTCUSDT"` 입니다. Binance 스트림 이름은 소문자만 사용하므로 구독 시 자동으로 소문자로 변환됩니다. 다른 심볼의 메시지는 무시됩니다.
Stream	TsgcChartBinanceStream (cbsKLine, cbsTicker, cbsTrade)	차트로 표시할 Binance 이벤트입니다: kline(캔들), 24시간 티커(최근 가격), 또는 개별 거래입니다.
Interval	TsgcWSBinanceChartIntervals	kline 간격입니다. 예를 들어 `bci1m` 입니다. **cbsKLine**에서만 사용되며, 다른 간격의 메시지는 무시됩니다.
ClosedOnly	bool	**cbsKLine** 전용입니다. true이면 마감된 캔들만 차트로 표시되며, 그렇지 않으면 아직 형성 중인 캔들이 메시지마다 그 자리에서 갱신됩니다.
Aggregator	TsgcChartAggregator (financial edition)	**cbsTrade** 전용입니다. 지정되어 있으면 모든 거래는 시리즈 버퍼 대신 **Aggregator.AddTrade**로 전달됩니다. 거래 자체는 표시되지 않고, 애그리게이터의 OHLC 시리즈가 캔들(거래량 포함)을 그립니다. 애그리게이터를 폐기해도 이 속성은 지워지지 않으며, 폐기된 애그리게이터는 계속 수신되는 거래를 무시합니다.
Active	bool(기본 속성을 재정의)	**TsgcChartDataSource_MQTT**와 동일하게 재선언됩니다.

이벤트

이벤트	형식	설명

OnTrade	TsgcChartBinanceTradeEvent(object Sender, double aTime, double aPrice, double aQty)	**cbsTrade** 전용입니다. **Symbol**의 모든 거래에 대해, 거래가 버퍼에 추가되거나 **Aggregator**로 전달되기 전에 UTC 시각, 가격, 수량과 함께 발생합니다.

메서드

메서드	반환 값	설명
Subscribe()	int	**API**의 `SubscribeKLine`, `SubscribeTicker` 또는 `SubscribeTrades`를 호출하여 **Symbol**(kline의 경우 **Interval**도 포함)에 대해 **Stream**을 구독합니다. API가 반환한 id를 반환합니다.
Unsubscribe()	int	이에 대응하는 구독 해지 호출입니다.

멀티캐스트 이벤트

모든 브리지 데이터 소스는 Delphi 라이브러리에서는 단순한 Delphi 메서드 포인터(**OnMQTTPublish**, **OnMessage**, **OnSSEMessage**, **OnEvent**)인 클라이언트 이벤트를 후크하지만, 여기서는 .NET 멀티캐스트 이벤트입니다. Delphi 소스는 이미 지정되어 있던 핸들러를 저장하고 자신의 핸들러를 설치한 뒤, 분리될 때 저장해 둔 핸들러를 복원합니다. 반면 .NET 데이터 소스는 연결될 때 +=로 자신의 핸들러를 호출 목록에 추가하고, 분리될 때 -=로 그 동일한 핸들러만 제거합니다. 이미 구독되어 있던, 또는 이후에 구독된 애플리케이션 핸들러는 계속 실행되며 소스에 의해 전혀 건드려지지 않습니다.
실질적인 차이는 존재 여부가 아니라 구독 순서입니다. 핸들러는 추가된 순서대로 실행되므로, 데이터 소스가 연결되기 전에 클라이언트에 구독된 애플리케이션 핸들러는 데이터 소스가 메시지를 차트로 표시하기 전에 실행되는 반면, Delphi는 항상 먼저 차트로 표시합니다. 후크는 런타임에만 발생하며 디자인 타임에는 발생하지 않습니다. .NET에는 **FreeNotification**이 없으므로, 기반 클라이언트를 폐기해도 데이터 소스의 **Client**(또는 **API**, **SSEClient**) 속성은 지워지지 않습니다. 대신 null로 설정하거나 데이터 소스를 폐기하십시오. **TsgcChartDataSource_SSE**에서는 **OnSSEMessage**의 애플리케이션 핸들러가 여전히 **Cancel**을 설정할 수 있는데, 이는 전체 호출 목록이 공유하므로 어느 핸들러든 true로 설정하면 다른 모든 핸들러에 대해서도 스트림이 중지됩니다.
메시지, 발행, 이벤트는 모두 기반 클라이언트가 알림을 보내는 스레드에서 도착합니다. 추가 작업은 기본 **TsgcChartDataSource**의 잠금과 제한된 알림을 거치므로, 어떤 알림 모드를 사용하든 브리지 데이터 소스와 함께 안전하게 사용할 수 있습니다.

예제: Binance 실시간 차트

```
using System;
using esegece.sgcCharts;
using esegece.sgcWebSockets;

namespace QuickStartBridge
{
    internal static class Program
    {
        private static void Main()
        {
            // ... chart: X is the trade time (UTC), Y is the trade price
            TsgcChart oChart = new TsgcChart();
            TsgcChartAxis oAxisX = oChart.Axes.Add();
            oAxisX.Name = "X";
            oAxisX.Position = TsgcChartAxisPosition.capBottom;
            oAxisX.AxisType = TsgcChartAxisType.catDateTime;
            TsgcChartAxis oAxisY = oChart.Axes.Add();
            oAxisY.Name = "Y";
            oAxisY.Position = TsgcChartAxisPosition.capLeft;

            TsgcChartSeries_Line oSeries = new TsgcChartSeries_Line();
            oSeries.Chart = oChart;
            oSeries.AxisX = "X";
            oSeries.AxisY = "Y";
```

```
            // ... sgcWebSockets: the Binance API on a WebSocket client
            TsgcWebSocketClient oWSClient = new TsgcWebSocketClient();
            TsgcWSAPI_Binance oBinance = new TsgcWSAPI_Binance();
            oBinance.Client = oWSClient;

            // ... the bridge data source: individual trades of BTCUSDT
            TsgcChartDataSource_Binance oDataSource = new TsgcChartDataSource_Binance();
            oDataSource.Series = oSeries;
            oDataSource.API = oBinance;
            oDataSource.Symbol = "BTCUSDT";
            oDataSource.Stream = TsgcChartBinanceStream.cbsTrade;

            // ... OnTrade fires for every trade before it is appended to the
            // ... chart; here it also keeps a running total of the traded volume
            double vVolume = 0;
            oDataSource.OnTrade += (Sender, aTime, aPrice, aQty) =>
            {
                vVolume += aQty;
                Console.WriteLine("trade price=" + aPrice.ToString("0.00") +
                    " qty=" + aQty.ToString("0.0000") +
                    " total volume=" + vVolume.ToString("0.0000"));
            };

            oDataSource.Active = true;
            oWSClient.Active = true;
            oDataSource.Subscribe();

            Console.WriteLine("Press Enter to exit.");
            Console.ReadLine();

            oDataSource.Active = false;
            oWSClient.Active = false;
            oDataSource.Dispose();
            oBinance.Dispose();
            oWSClient.Dispose();
            oChart.Dispose();
        }
    }
}
```

참고 항목

- Bridge Overview(Delphi)
- .NET Overview

Delphi와의 .NET 차이점

sgcCharts .NET은 두 언어가 허용하는 한 Delphi의 클래스 이름, 속성, 동작을 그대로 유지합니다(.NET Overview 참조). 이 페이지는 이름이나 형식이 자연스러운 C# 대응물로 바뀐 곳이 아니라, .NET 포트가 Delphi 라이브러리와 실제로 다르게 동작하는 부분을 나열합니다.

이벤트

영역	Delphi	.NET
실시간 브리지 데이터 소스 (**esegece.sgcCharts.sgcWebSockets**): **OnMQTTPublish**, **OnMessage**(WebSocket 클라이언트 및 Binance 클라이언트), **OnSSEMessage** / **OnEvent**	단일 메서드 포인터입니다. 데이터 소스는 클라이언트의 현재 핸들러를 저장하고 자신의 핸들러를 설치한 뒤, 중지될 때 저장해 둔 핸들러를 복원합니다.	실제 C# 멀티캐스트 `event` 입니다(`+=` / `-=`). 데이터 소스는 핸들러를 저장하고 복원하는 대신 자신의 핸들러를 구독하고 구독 해지합니다. 전체 메커니즘은 Bridge Overview를 참조하십시오.
핵심 차트 모델: **TsgcChart.OnChange**, **TsgcChartDataSource.OnDataChanged** / **OnBeforeAppend**, **TsgcChart.OnCrosshairMove** / **OnSeriesClick** / **OnZoom**, 그리고 축, 시리즈, 드로잉의 유사한 멤버들	단일 메서드 포인터이며, 대입으로 교체됩니다.	차이 없음: 이들은 C# `event` 가 아니라 여전히 단순한 `get; set;` 델리게이트 속성으로 남아 있으므로, 새 핸들러를 지정하면 Delphi의 단일 메서드 포인터와 정확히 동일하게 이전 핸들러를 대체합니다. 둘 다 **OnX**라는 이름을 사용하므로, 위의 브리지 행과 대비하기 위해서만 여기에 나열했습니다.

인덱싱된 속성

영역	Delphi	.NET
TsgcChartAxes, **TsgcChartPanes**, **TsgcChartDrawings**, **TsgcChartPalette**, **TsgcChartJSONNode**	인덱싱된 배열 속성입니다. 예: `Axes[Index]` .	실제 C# 인덱서입니다: `this[int Index]` .
TsgcChartBuffer(`X` , `Y`), **TsgcChartOHLCBuffer**(`X` , `Open` , `High` , `Low` , `Close` , `Volume`), 그리고 재무 시리즈의 `XValues` 와 `Values`	열마다 인덱싱된 배열 속성입니다. 예: `Open[Index]` , 그리고 2차원 속성인 `Values[Line, Index]` 입니다.	인덱스를 매개변수로 받는 일반 메서드입니다: `Open(Index)` , `Values(Line, Index)` . C# 클래스는 인덱서 시그니처(`this[...]`)를 하나만 선언할 수 있는데, 이 클래스들은 같은 클래스에서 독립적으로 이름 붙여진, 또는 다중 매개변수를 가진 여러 인덱스 접근자가 필요하므로 대신 메서드가 됩니다.
pie, radar, heatmap, treemap, funnel 시리즈와 그 도우미: **TsgcChartSeries_Radar.SpokeText**, **TsgcChartSeries_Funnel.DrawnStages**, **TsgcChartSeries_Heatmap.Cells**, **TsgcChartHeatmapData.Cells**, **TsgcChartColorScale.LUT**, **TsgcChartPieLayout.Slices**,	인덱스 배열 속성. 예: `SpokeText[Index]` , `DrawnStages[Index]` , 그리고 읽기와 쓰기가 가능한 `Cells[Line, Cell]` .	메서드: `SpokeText(Index)` , `DrawnStages(Index)` (단계 레코드의 복사본), 읽기용 `Cells(Line, Cell)` , 쓰기용 `SetCells(Line, Cell, Value)` . `DrawnStages` 는 Delphi가 `EListError` 를 발생시키는 곳에서 `ArgumentOutOfRangeException`

TsgcChartTreemapLayout.Nodes, **Crumbs**, **DrawnNodes**		을 던집니다. 레이아웃 메서드 `Slices`, `Nodes`, `Crumbs`, `DrawnNodes` 도 `0..Count - 1` 밖의 인덱스에 대해 이를 던지며, Delphi는 내부 배열을 검사 없이 읽습니다.

pie, radar, heatmap, treemap, funnel 시리즈

영역	Delphi	.NET
pie, radar, treemap, funnel 시리즈의 **SetValueAt**과 **DeleteItem**	범위를 벗어난 인덱스는 `ERangeError` 를 발생시킵니다.	범위를 벗어난 인덱스는 `ArgumentOutOfRangeException` 을 던집니다.
TsgcChartSeries_Heatmap: **AppendLine**, **SetLineLabels**, **SetCellLabels**	오픈 배열 매개변수 (`AppendLine([1, 2, 3])`)와 레이블용 `TStrings` 오버로드.	`double[]` 또는 `string[]` 매개변수 (`params` 목록이 아님)와 레이블용 **TsgcChartStrings** 오버로드. 단순한 `null` 은 두 레이블 오버로드 사이에서 모호하므로 캐스팅하십시오.
항목(**Slices**, **Points**, **Nodes**, **Stages**)과 treemap의 **Levels**	오브젝트 인스펙터에서 편집하고 폼(DFM)에 저장됩니다.	폼 디자이너 스트리밍이 없습니다. 항목은 **AddItem**으로, 레벨은 **Levels.Add()**로 코드에서 추가하십시오. Series Gallery를 참조하십시오.
래스터 출력(**RenderToPNGFile**, **RenderToBuffer**, **SaveToPNG**)	sgcUICore 래스터 캔버스가 그립니다.	SkiaSharp가 그리므로 안티앨리어싱된 가장자리 픽셀이 Delphi와 약간 다를 수 있습니다. 레이아웃은 같고 SVG 출력은 Delphi SVG와 바이트 단위로 같습니다.

DataSet 또는 DataReader에서 불러오기

TsgcChartDataSource_DataSet 은 Delphi `TDataSet` 에 대응하는 .NET 인 `System.Data.DataTable` 또는 임의의 `IDataReader`에서 데이터를 불러옵니다. `DataTable`에는 커서가 없으므로 불러온 뒤 되돌릴 현재 레코드가 없습니다. **Refresh** 실행 전에 있던 레코드로 원본 `TDataSet`을 되돌리는 Delphi 데이터 소스의 북마크 처리에는 .NET 대응물이 없습니다.

에디션

Delphi sgcCharts 는 Community, Trial, Registered 에디션으로 제공됩니다. sgcCharts .NET 은 **Community**(`-p:SgcCommunity=true`로 빌드)와 **Registered**(기본 빌드)만 제공합니다. .NET 트라이얼 빌드나 트라이얼 상수는 없습니다.

참고 항목

- .NET Overview
- Series Gallery
- Bridge Overview
- TsgcChartDataSource_DataSet

문제 해결

현재까지 알려진 문제는 없습니다.

소개

sgcMaps는 인터랙티브 지도를 위한 Delphi 구성 요소 라이브러리입니다. 지도 컨트롤은 브라우저 컨트롤이나 JavaScript 없이 래스터 타일과 벡터 타일을 네이티브로 그리고, 그 위에 마커, 경로, 도형, 레이블, 히트맵을 표시합니다. 비시각 구성 요소는 지오코딩, 길 찾기, 장소 검색, 고도, 시간대, 기기 위치를 추가하며, 오프라인 라우터와 오프라인 지오코더는 네트워크 연결이 전혀 없어도 응답합니다.
VCL 컨트롤, FireMonkey 컨트롤, Lazarus(LCL) 컨트롤은 하나의 지도 모델을 공유하므로, 레이어, 마커, 경로를 추가하는 코드는 모든 프레임워크에서 동일합니다. Lazarus 컨트롤은 타일을 다운로드할 수 없으므로 Lazarus에서는 폴더의 타일만 표시합니다. 지도 컨트롤은 창 없이 렌더링할 수도 있어, 콘솔 애플리케이션이나 서비스를 정적 지도 렌더러로 만들 수 있습니다.

- 개요: sgcMaps에 포함된 내용과 각 기능에 필요한 Delphi 버전입니다.
- 설치: sgcUI 설치 프로그램, 패키지, 요구 사항입니다.
- 빠른 시작 VCL과 빠른 시작 FMX: 몇 줄로 만드는 동작하는 지도입니다.
- 지도 컨트롤: TsgcVCLMap과 TsgcFMXMap, 그리고 그 레이어와 오버레이입니다.
- 온라인 서비스: 지오코딩, 길 찾기, 장소, 고도, 시간대입니다.
- 라우팅: 대화형 경로 편집과 오프라인 라우터입니다.
- 오프라인 지오코딩: 로컬 GeoNames 인덱스를 이용한 장소 검색입니다.
- 위치: 실제 또는 시뮬레이션된 기기의 위치입니다.
- .NET: .NET용 sgcMaps로, Windows Forms 컨트롤, 헤드리스 렌더링, 오프라인 서비스를 갖춘 C# 포트입니다.
- 데모, 도구, Lazarus, 문제 해결: 샘플 프로젝트, 데이터 도구, LCL 컨트롤, 흔히 겪는 문제입니다.

개요

sgcMaps는 **지도 모델**을 중심으로 만들어졌습니다. 지도가 무엇을 보여 주는지 설명하는 컬렉션들의 집합입니다. Layers는 타일 소스를 담고, Markers, Polylines, Polygons, Rectangles, Circles, Labels, Heatmaps는 오버레이를 담습니다. Options와 MarkerOptions는 보기의 동작을 설정합니다. 모델 클래스는 **sgcMaps_Classes**에 있으며 TsgcVCLMap과 TsgcFMXMap에서 동일합니다.
비시각 구성 요소는 특정 컨트롤이 아니라 이 컬렉션들과 함께 동작합니다. TsgcMapRouteCalculator는 경유지를 지도의 Markers에, 경로를 Polylines에 보관하며, TsgcMapLocation는 위치 마커와 정확도 원을 최신 상태로 유지합니다. 따라서 같은 구성 요소를 VCL 폼과 FireMonkey 폼에서 모두 사용할 수 있습니다.

기능

타일과 레이어

- 내장 공급자의 온라인 래스터 타일: OpenStreetMap, OpenTopoMap, Esri World Imagery, Esri World Street, CARTO Light 및 Dark, MapTiler Streets 및 Satellite, Thunderforest Cycle 및 Transport, Stadia Alidade Smooth 및 Smooth Dark.
- {z}, {x}, {y}, 하위 도메인, 레티나, API 키 자리 표시자가 있는 URL 템플릿을 통한 그 밖의 모든 타일 서버.
- 레이어별 디스크 캐시. 이미 본 타일은 연결 없이 그려집니다.
- z/x/y 폴더 또는 MBTiles 파일의 오프라인 타일.
- 서버 또는 로컬 MBTiles 파일의 벡터 타일(Mapbox Vector Tiles). MapLibre 또는 Mapbox GL 스타일로 그립니다.
- 불투명도, 줌 범위, 기본 또는 오버레이 역할, 지도의 선택적 레이어 전환기를 갖춘 여러 레이어.
- 영역과 줌 범위의 모든 타일로 폴더를 채우는 타일 다운로더. 오프라인 지도 팩용입니다.

오버레이

- 팝업과 툴팁 텍스트, 색상, 사용자 지정 아이콘 비트맵, 앵커, 드래그를 지원하는 마커.
- 마커 클러스터링. 수천 개의 마커도 읽기 쉽고 빠르게 유지됩니다.
- 폴리라인(실선, 점선, 측지선), 구멍이 있는 폴리곤, 직사각형, 원. 모두 마우스로 편집할 수 있습니다.
- 텍스트 레이블과 여러 색상 그라데이션을 지원하는 가중치 히트맵.

상호작용과 출력

- 드래그, 휠, 버튼 줌, 지도 회전(방위각), 축척 막대, 타일 저작자 표시.
- 클릭, 마커 드래그, 도형 편집, 클러스터 클릭, 레이어 변경, 줌 변경, 타일 오류에 대한 이벤트.
- 비트맵, PNG 또는 BMP 파일로 렌더링하며, 부모 창이 없는 컨트롤에서도 가능합니다.

데이터

- 마커, 폴리라인, 폴리곤의 GeoJSON, GPX, KML 가져오기와 내보내기.
- 지오메트리 함수: 거리, 방위각, 목적지 점, 측지 보간, 폴리곤 내 점 판정, 선 단순화, 폴리라인의 가장 가까운 점.
- OpenStreetMap .osm.pbf 및 .osm XML 파일의 스트리밍 리더.

서비스

- Nominatim, Google, HERE, Azure Maps, Mapbox, OpenRouteService, Geoapify를 통한 지오코딩과 역지오코딩.
- OSRM, OpenRouteService, Google Routes, HERE, Azure Maps, Mapbox, Geoapify를 통한 길 찾기.
- Photon, Nominatim, Google, HERE, Mapbox, Geoapify를 통한 장소 자동 완성.
- Open-Elevation, OpenTopoData, Google을 통한 점과 경로의 고도.
- TimeAPI, GeoNames, Google, Azure Maps를 통한 좌표의 시간대.

라우팅, 오프라인 데이터, 위치

- 경유지를 드래그할 수 있고 대체 경로를 지원하는 대화형 경로 편집.

- OpenStreetMap 데이터로 만든 그래프에서 자동차, 자전거, 도보를 위한 오프라인 라우팅.
- GeoNames 덤프로 만든 SQLite 인덱스의 오프라인 지오코딩.
- Windows, Android, iOS, macOS의 기기 위치와 테스트를 위한 위치 및 트랙 시뮬레이션.

Delphi 버전

지도 모델, VCL 컨트롤, GPX, KML, 지오메트리 함수는 Delphi 7부터 Delphi 13까지 컴파일되며, 패키지는 C++Builder 2007부터 C++Builder 13까지도 지원합니다. 일부 기능은 나중에 추가된 RTL 부분에 의존하므로, 해당 구성 요소는 그 기능을 갖춘 IDE 버전에서만 등록됩니다.

기능	유닛	최소 버전
VCL 지도 컨트롤, 오버레이, GPX, KML, 지오메트리	sgcMaps_VCL, sgcMaps_Classes	Delphi 7
PNG 출력	sgcMaps_VCL	Delphi 2009
서버의 타일, 타일 다운로더	sgcMaps_TileSource_HTTP, sgcMaps_TileDownloader	Delphi 2009(Indy 10, THTTPClient는 Delphi XE8부터)
경로 계산기, 오프라인 라우터, 서버의 벡터 타일	sgcMaps_RouteCalculator, sgcMaps_OfflineRouter, sgcMaps_TileSource_MVT	Delphi XE2
FireMonkey 지도 컨트롤, 기기 위치, GeoJSON	sgcMaps_FMX, sgcMaps_Location, sgcMaps_GeoJSON	Delphi XE6
MBTiles, 벡터 MBTiles, 오프라인 지오코더	sgcMaps_TileSource_MBTiles, sgcMaps_OfflineGeocoder	Delphi XE7 (FireDAC SQLite)
온라인 서비스	sgcMaps_Geocoding, sgcMaps_Directions, sgcMaps_Places, sgcMaps_Elevation, sgcMaps_TimeZone	Delphi XE8 (THTTPClient, System.JSON)

Lazarus에서 패키지는 LCL 지도 컨트롤, 폴더 타일 소스, 공급자, 래스터 엔진, 지오메트리 함수를 포함합니다. LCL 컨트롤은 타일을 다운로드할 수 없으므로 폴더의 타일만 표시하며, GeoJSON, GPX, KML은 FPC에서 컴파일되지 않습니다. Lazarus를 참조하십시오.

.NET

sgcMaps에는 .NET 8과 .NET 9용 .NET 포트인 sgcMaps for .NET도 있습니다. 같은 지도 모델, 온라인 서비스, 오프라인 데이터 파일을 갖추었으며, Windows Forms 컨트롤과 Linux 및 macOS에서도 실행되는 헤드리스 렌더러가 있습니다. .NET 개요를 참조하십시오.

설치

sgcMaps는 다른 eSeGeCe 비주얼 제품과 함께 통합 **sgcUI 설치 프로그램**으로 제공됩니다. 설치 프로그램은 라이선스에 포함된 제품을 알고 있으므로, IDE 선택 외에는 직접 고를 것이 없습니다.

sgcUI 설치 프로그램으로 설치하기

1. 실행 중인 IDE를 모두 닫고 sgcUI 설치 프로그램을 시작합니다.
2. eSeGeCe 계정으로 로그인합니다.
3. 설치 프로그램이 라이선스를 읽고 포함된 제품을 나열합니다. 라이선스가 sgcMaps를 포함하면 sgcMaps에 체크 표시가 되어 있습니다.
4. 설치할 Delphi 및 C++Builder 버전을 선택하고 설치를 시작합니다.
5. IDE를 엽니다. 구성 요소는 Tool Palette의 **SGC Maps** 페이지에 있습니다.

지도 컨트롤은 해당 프레임워크의 팔레트에만 표시됩니다. VCL 프로젝트에는 TsgcVCLMap, FireMonkey 프로젝트에는 TsgcFMXMap입니다. 비시각 구성 요소는 둘 다에 표시됩니다.

요구 사항

- Delphi XE7부터 런타임 패키지에는 **SQLite** 드라이버가 있는 **FireDAC**이 필요합니다(FireDAC, FireDACCommon, FireDACCommonDriver, FireDACSqliteDriver 패키지). sgcMaps는 이를 MBTiles 파일과 오프라인 지오코더 인덱스를 읽는 데 사용하므로, Delphi 에디션에 FireDAC SQLite가 포함되어 있어야 합니다.
- 온라인 서비스는 Delphi XE8부터 사용할 수 있는 THTTPClient(System.Net.HttpClient)와 System.JSON을 사용합니다.
- TsgcMapLocation은 Delphi XE6부터 사용할 수 있는 System.Sensors를 사용합니다.

패키지

Packages 폴더에는 IDE 버전마다 한 쌍의 패키지가 있습니다. 런타임 패키지 **sgcMapsD*nn*.dpk**와 디자인 타임 패키지 **dclsgcMapsD*nn*.dpk**입니다. 범위는 Delphi 7(D7)부터 Delphi 13(D13)까지입니다. C++Builder는 C++Builder 2007부터 C++Builder 13까지 해당하는 **sgcMapsC*nn*.cbproj**와 **dclsgcMapsC*nn*.cbproj** 프로젝트를 사용합니다. 수동으로 설치하려면 먼저 런타임 패키지를 컴파일한 다음 디자인 타임 패키지를 설치하고, Source 폴더를 IDE의 라이브러리 경로에 추가하십시오.

Lazarus

Lazarus에는 자체 패키지 **Packages\Lazarus\sgcMapsLazarus.lpk**가 있습니다. Package, Open Package File로 열고 컴파일한 다음 Use, Install을 선택하면 IDE가 다시 빌드됩니다. LCL, FCL, IDEIntf 패키지가 필요하며, 지도 컨트롤(LCL의 TsgcVCLMap), 폴더 타일 소스, 공급자, 래스터 엔진, 지오메트리 함수를 포함합니다. LCL 컨트롤은 타일을 다운로드할 수 없으므로 폴더의 타일만 표시합니다. GeoJSON, GPX, KML은 패키지에 나열되어 있지만 FPC에서는 컴파일되지 않으며, 온라인 서비스, 경로 계산기, 오프라인 라우터, 오프라인 지오코더, TsgcMapLocation은 Delphi에서만 사용할 수 있습니다. 자세한 내용과 예제는 Lazarus 항목에 있습니다.

유닛

코드에서 만드는 각 구성 요소의 유닛을 uses 절에 추가하십시오. 폼에 놓은 구성 요소는 유닛이 자동으로 추가됩니다.

유닛	내용
sgcMaps_Classes	지도 모델: 레이어, 마커, 폴리라인, 폴리곤, 직사각형, 원, 레이블, 히트맵, 옵션, 그리고 TsgcMapCoord.
sgcMaps_VCL	TsgcVCLMap.

sgcMaps_FMX	TsgcFMXMap.
sgcMaps_Providers	내장 타일 공급자와 sgcMapApplyProvider.
sgcMaps_Services	TsgcMapService, 온라인 서비스의 기반 클래스입니다.
sgcMaps_Geocoding	TsgcMapGeocoder와 지오코딩 결과 클래스입니다.
sgcMaps_Directions	TsgcMapDirections, 경로 결과 클래스, sgcMapAddRouteToPolylines입니다.
sgcMaps_Places	TsgcMapPlaces.
sgcMaps_Elevation	TsgcMapElevation.
sgcMaps_TimeZone	TsgcMapTimeZone.
sgcMaps_RouteCalculator	TsgcMapRouteCalculator.
sgcMaps_OfflineRouter	TsgcMapOfflineRouter와 TsgcMapRouteGraphBuilder입니다.
sgcMaps_OfflineGeocoder	TsgcMapOfflineGeocoder와 TsgcMapGeoIndexBuilder입니다.
sgcMaps_Location	TsgcMapLocation.

.NET

sgcUI 설치 프로그램은 .NET용으로는 아무것도 설치하지 않습니다. sgcMaps for .NET은 Registered 에디션 전용으로 자체 NuGet 패키지(esegece.sgcMaps와 그 관련 패키지)로 배포됩니다. .NET 설치를 참조하십시오.

빠른 시작 VCL

이 안내는 VCL 폼에 지도를 표시하는 방법을, 먼저 폼 디자이너로, 그다음 전부 코드로 보여 줍니다.

1. 디자인 타임

1. VCL 애플리케이션을 만들고 SGC Maps 페이지의 **TsgcVCLMap**을 폼에 놓습니다. **Align** 속성을 alClient로 설정합니다.
2. 폼의 **OnCreate** 이벤트에서 **Layers.Add**로 기본 레이어를 추가하고 **Provider** 속성을 설정합니다. 공급자가 있는 레이어는 다른 설정 없이도 해당 공급자의 타일을 다운로드합니다. 선택 사항인 **sgcMapApplyProvider** 호출은 공급자의 소스 형식, URL 템플릿, 최대 줌, 저작자 표시도 레이어에 복사합니다.
3. **SetCenter**와 **Zoom**으로 초기 보기를 설정하고, **AddMarker**, **AddPolyline**, **AddCircle**로 오버레이를 추가합니다.
4. 애플리케이션을 실행합니다. 지도를 드래그하여 이동하고, 마우스 휠이나 확대/축소 버튼으로 줌합니다.

이 단계에 해당하는 전체 OnCreate 핸들러는 TsgcVCLMap 항목의 예제입니다.

2. 코드에서

아래 프로그램은 .dfm 파일 없이 같은 지도를 만듭니다. 폼이 자체 컨트롤을 만들므로, 생성자가 상속된 **Create** 대신 **CreateNew**를 호출합니다. 마커를 클릭하면 팝업이 열리고, 다른 곳을 클릭하면 그 위치에 새 마커가 추가됩니다.

타일에 대해서

레이어는 다운로드한 모든 타일을 **CacheFolder**에 캐시하므로, 이미 본 보기는 연결 없이 즉시 그려집니다. OpenStreetMap 타일 서버는 자원봉사자가 운영하며 과도한 사용을 허용하지 않는 사용 정책이 있습니다. 운영 애플리케이션에는 MapTiler, Thunderforest, Stadia 같은 상용 공급자(레이어의 **ApiKey** 설정), 자체 타일 서버, 또는 폴더나 MBTiles 파일의 오프라인 타일을 사용하십시오. 타일 공급자는 저작자 표시가 보이도록 요구하므로 **Options.ShowAttribution**을 활성화한 상태로 유지하십시오.

다음 단계

- TsgcVCLMap: 컨트롤의 모든 속성, 메서드, 이벤트입니다.
- TsgcMapGeocoder: 주소를 검색하여 지도에 표시합니다.
- TsgcMapRouteCalculator: 사용자가 지도를 클릭하여 경로를 만들게 합니다.

빠른 시작 FMX

TsgcFMXMap은 FireMonkey 지도 컨트롤입니다. VCL 컨트롤과 지도 모델을 공유하므로 레이어, 마커, 도형은 빠른 시작 VCL과 똑같이 만듭니다. Delphi XE6 이상이 필요하며 FireMonkey의 데스크톱 및 모바일 플랫폼에서 실행됩니다.

1. 디자인 타임

1. 멀티 디바이스 애플리케이션을 만들고 SGC Maps 페이지의 **TsgcFMXMap**을 폼에 놓습니다. **Align** 속성을 Client로 설정합니다.
2. 폼의 **OnCreate** 이벤트에서 **Layers.Add**로 기본 레이어를 추가하고 **Provider** 속성을 설정하면 해당 공급자의 타일을 다운로드하기에 충분합니다. 선택 사항인 **sgcMapApplyProvider** 호출은 공급자의 소스 형식, URL 템플릿, 최대 줌, 저작자 표시도 레이어에 복사합니다. 모바일 기기에서는 애플리케이션 폴더가 읽기 전용이므로, 타일 캐시를 **TPath.GetCachePath** 같은 쓰기 가능한 폴더에 두십시오.
3. **SetCenter**와 **Zoom**으로 초기 보기를 설정하고 마커를 추가합니다.
4. 애플리케이션을 실행합니다. 손가락 하나로 이동하고, 핀치로 확대/축소하며, **Options.RotationEnabled**가 True이면 손가락 두 개로 회전합니다.

이 단계에 해당하는 전체 OnCreate 핸들러는 TsgcFMXMap 항목의 예제입니다.

2. 코드에서

아래 프로그램은 코드로 지도를 만듭니다. 마커 클러스터링을 켜서 지도를 축소하면 가까운 마커가 번호가 있는 원 하나로 합쳐지고, 마커를 탭하면 그 마커의 팝업이 표시됩니다.

다음 단계

- TsgcFMXMap: VCL 컨트롤과 다른 점입니다.
- TsgcMapLocation: 기기의 위치를 표시하고 따라갑니다.
- TsgcMapPlaces: 사용자가 입력하는 동안 장소를 제안합니다.

데모

sgcUI 설치본의 **Demos\Maps** 폴더에는 Delphi 프로젝트 일곱 개가 있습니다. 각 프로젝트는 기능 한 그룹을 보여 주는 작고 완결된 애플리케이션이며, 함께 사용하면 sgcMaps의 모든 구성 요소를 다룹니다.

데모	프레임워크	최소 버전	보여 주는 내용
01 Offline Map	VCL	Delphi 7	폴더, MBTiles, 온라인 타일, 마커, 도형, GeoJSON, GPX, KML 가져오기입니다.
02 Map Features	VCL	Delphi 7	공급자와 API 키, 클러스터링, 편집, 히트맵, 회전, 내보내기입니다.
03 Services	VCL	Delphi XE8	장소 검색, 지오코딩, 길 찾기, 고도, 시간대, 위치입니다.
04 Offline Services	VCL	Delphi XE7	오프라인 타일, 오프라인 지오코더, 오프라인 라우터입니다.
05 Static Render	콘솔	Delphi 7	창 없이 PNG 또는 BMP 파일로 렌더링한 지도입니다.
06 FireMonkey	FireMonkey	Delphi XE6	데스크톱과 모바일의 FireMonkey 컨트롤이며, 위치, 검색, 라우팅을 포함합니다.
07 Vector Tiles	VCL	Delphi XE2	서버 또는 로컬 MBTiles 파일의 벡터 타일이며, 스타일, 레티나, 회전을 포함합니다.

데모의 소스에는 API 키가 없습니다. 키가 필요한 공급자를 사용하는 데모에는 키를 입력하는 편집 상자가 있으며, 모든 데모는 키가 필요 없는 서비스와 타일로 시작합니다.
대부분의 데모는 공개 OpenStreetMap 서버에서 타일을 다운로드합니다. 데모를 체험하기에는 괜찮지만 운영 애플리케이션에는 적합하지 않습니다. 사용 정책은 문제 해결을 참조하십시오.

01 Offline Map

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\01.OfflineMap	sgcMapsDemo.dpr	VCL	Delphi 7(온라인 타일은 Delphi 2009부터, GeoJSON은 Delphi XE6부터, MBTiles는 Delphi XE7부터)

보여 주는 내용

TsgcVCLMap의 기본 레이어는 **Folder**, **MBTiles**, **Online** 옵션으로 선택하는 세 가지 소스 중 하나에서 가져올 수 있습니다. z/x/y 타일 폴더, MBTiles 파일, 또는 디스크 캐시가 있는 OpenStreetMap 타일 서버입니다. 지도는 줌 13의 마드리드에서 시작합니다.

- **Go To Coordinates**와 줌 상자로 보기를 이동합니다.
- **Marker at Center**, **Sample Markers**, **Clear Markers**는 마커를 추가하고 제거합니다.
- **Sample Circle**, **Sample Polyline**, **Sample Polygon**은 도형을 그립니다.
- **Zoom Controls** 및 **Scale Bar** 확인란은 지도의 해당 부분을 켜고 끕니다.
- **GeoJSON...**, **GPX...**, **KML...**은 파일을 지도로 가져옵니다.
- 로그에는 OnMapClick, OnMarkerClick, OnZoomChanged 이벤트가 표시됩니다.

구성 요소와 유닛

폴더, MBTiles, HTTP 레이어를 사용하는 TsgcVCLMap과 함수 sgcMapLoadGeoJSONFile, sgcMapLoadGPXFile, sgcMapLoadKMLFile입니다. 유닛은 sgcMaps_Classes, sgcMaps_TileSource, sgcMaps_VCL, sgcMaps_GeoJSON, sgcMaps_GPX, sgcMaps_KML입니다.

실행 방법

IDE에서 **sgcMapsDemo.dpr**을 열고 컴파일하여 실행합니다. 데모는 sgcUI 설치 프로그램이 설정하는 라이브러리 경로를 통해 sgcMaps 유닛을 찾습니다. 수동으로 설치한 경우에는 먼저 Source 폴더를 라이브러리 경로에 추가하십시오.

데이터와 키

- API 키가 필요하지 않습니다.
- 프로젝트 옆의 **data** 폴더에는 가져오기 버튼용 sample.geojson, sample.gpx, sample.kml이 있습니다. 데모는 먼저 실행 파일 옆에서, 그다음 현재 폴더에서 이 폴더를 찾습니다.
- 상대 타일 경로는 실행 파일이 있는 폴더를 기준으로 해석됩니다. 기본 경로는 z\x\y.png 파일이 들어 있는 폴더인 **tiles** 이며, 타일 다운로더로 채울 수 있습니다(오프라인 타일과 타일 다운로더 참조).
- 온라인 타일은 실행 파일 옆의 **tile_cache** 폴더에 캐시됩니다.
- 타일 폴더나 MBTiles 파일이 없으면 데모는 로그에 한 줄을 기록합니다.
- Delphi XE6보다 오래된 버전에서는 GeoJSON 버튼이 비활성화되고, Delphi XE7보다 오래된 버전에서는 MBTiles 옵션이 FireDAC이 필요하다고 로그에만 기록합니다.

02 Map Features

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\02.MapFeatures	sgcMapsFeaturesDemo.dpr	VCL	Delphi 7(온라인 타일과 PNG는 Delphi 2009부터, GeoJSON은 Delphi XE6부터)

보여 주는 내용

TsgcVCLMap의 그리기 및 상호작용 기능이며, 탭마다 하나씩 있습니다.

탭	할 수 있는 작업
Layers	기본 레이어를 선택하고, MapTiler, Thunderforest, Stadia의 API 키를 입력한 다음 **Apply key**를 눌러 해당 레이어를 추가하며, 지도에 레이어 전환기를 표시합니다.
Clusters	마드리드 주변에 임의의 마커 2000개를 추가하고 클러스터링을 켜고 끕니다.
Heatmap	가중치가 있는 점 5000개로 이루어진 히트맵을 추가하고 반경, 그라데이션, 불투명도를 변경합니다.
Drawing	드래그할 수 있는 마커, 편집 가능한 폴리라인, 구멍이 있는 편집 가능한 폴리곤, 편집 가능한 직사각형, 그리고 마드리드에서 뉴욕까지의 측지선을 추가합니다.
Rotation	회전을 활성화하고, 방위각을 설정하고, Ctrl+드래그 또는 오른쪽 드래그로 회전하며, 지도를 북쪽이 위로 오도록 재설정합니다.
내보내기	오버레이를 GeoJSON, GPX, KML로 내보내고, 지도를 PNG로 저장합니다(Delphi 2009 이전에는 BMP).

로그에는 OnMarkerDragEnd, OnShapeEdited, OnLayerChanged, OnClusterClick, OnBearingChanged가 표시됩니다.

구성 요소와 유닛

TsgcVCLMap, sgcMaps_Providers 의 공급자 목록(sgcMapGetProviderInfo), TsgcMapGPX, sgcMapSaveGeoJSONFile, sgcMapSaveKMLFile, sgcMapDistance 입니다. 유닛은 sgcMaps_Classes, sgcMaps_VCL, sgcMaps_Providers, sgcMaps_Geometry, sgcMaps_GPX, sgcMaps_KML, sgcMaps_GeoJSON입니다.

실행 방법

IDE에서 **sgcMapsFeaturesDemo.dpr**을 열고 컴파일하여 실행합니다. 데모는 sgcUI 설치 프로그램이 설정하는 라이브러리 경로를 통해 sgcMaps 유닛을 찾습니다. 수동으로 설치한 경우에는 먼저 Source 폴더를 라이브러리 경로에 추가하십시오.

데이터와 키

- 시작할 때 데모는 키가 필요 없는 모든 공급자마다 기본 레이어를 하나씩 추가합니다. OpenStreetMap, OpenTopoMap, Esri 레이어, CARTO Light 및 Dark입니다.
- MapTiler, Thunderforest, Stadia에는 해당 웹 사이트에서 발급받은 API 키가 필요합니다. Layers 탭에 입력하고 Apply key를 누르십시오.
- 공급자마다 자체 타일 캐시가 **%TEMP%\sgcMapsFeaturesDemo**에 있습니다.

- 내보낸 파일은 저장 대화 상자에서 선택한 위치에 기록됩니다.

03 Services

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\03.Services	sgcMapsServicesDemo.dpr	VCL	Delphi XE8

보여 주는 내용

모든 온라인 서비스를 하나의 폼에 담았습니다. 각 탭에는 공급자 목록과 API 키 상자가 있으며, 결과는 지도에 그려집니다.

탭	할 수 있는 작업
Search	입력하는 동안의 장소 제안(Photon, Nominatim, Google, HERE, Mapbox, Geoapify)과 주소의 지오코딩입니다.
Reverse	지도를 클릭하여 그 지점의 주소를 가져옵니다.
Directions	두 주소 사이 또는 지도에서 클릭한 경유지 사이(드래그 가능)의 경로이며, 이동 수단, 대체 경로, 그리고 OSRM, OpenRouteService, Google Routes, HERE, Azure Maps, Mapbox, Geoapify를 지원합니다.
Elevation	Open-Elevation, OpenTopoData 또는 Google에서 가져온 경로의 고도 프로파일입니다.
Time zone	TimeAPI, GeoNames, Google 또는 Azure Maps에서 가져온 지도 중심의 시간대입니다.
Location	경로를 따라 GPS 주행을 시뮬레이션하거나, 지도가 위치를 따라가도록 하거나, 기기의 위치를 사용합니다.

구성 요소와 유닛

TsgcMapPlaces, TsgcMapGeocoder, TsgcMapDirections, TsgcMapRouteCalculator, TsgcMapElevation, TsgcMapTimeZone, TsgcMapLocation을 TsgcVCLMap이 있는 폼에서 코드로 생성합니다.

실행 방법

IDE에서 **sgcMapsServicesDemo.dpr**을 열고 컴파일하여 실행합니다. 데모는 sgcUI 설치 프로그램이 설정하는 라이브러리 경로를 통해 sgcMaps 유닛을 찾습니다. 수동으로 설치한 경우에는 먼저 Source 폴더를 라이브러리 경로에 추가하십시오.

데이터와 키

- 기본 공급자는 키가 필요 없습니다. Photon, Nominatim, OSRM 데모 서버, OpenTopoData, TimeAPI입니다.
- Google, HERE, Azure Maps, Mapbox, OpenRouteService, Geoapify에는 개발자 콘솔에서 발급받은 키가 필요합니다. GeoNames는 GeoNames 사용자 이름을 키로 입력하십시오.
- 데모는 공개 서버의 요청에 따라 모든 서비스에 자체 UserAgent를 설정하며, 폼에 그 제한을 표시합니다. 초당 요청 1건이며 대량 요청은 허용되지 않습니다. 운영 환경에서는 자체 서버나 상용 공급자를 사용하십시오.
- 타일은 OpenStreetMap에서 가져오며 실행 파일 옆의 **tilecache** 폴더에 캐시됩니다.
- 서비스 오류는 로그에 기록됩니다. 기기 위치를 사용할 수 없으면 데모가 이를 알리고 해당 옵션의 선택을 해제합니다.

04 Offline Services

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\04.OfflineServices	sgcMapsOfflineDemo.dpr	VCL	Delphi XE7 (FireDAC SQLite)

보여 주는 내용

네트워크 연결 없이 하는 검색과 라우팅입니다. 데모는 무료로 다운로드한 데이터로 데이터 파일을 만들고, 다음 시작 때 다시 열며, 모든 요청에 로컬에서 응답합니다. 지도는 안도라에서 시작합니다.

- **Offline tiles**의 **Folder...** 또는 **MBTiles...**로 로컬 기본 지도를 설정합니다. 이를 설정하지 않았거나 파일이 없으면 데모는 OpenStreetMap 온라인 타일을 사용하고 로그에 그렇게 기록합니다.
- **Build from GeoNames...**는 GeoNames 덤프에서 지오코더 인덱스를 만들고, **Open index...**는 기존 인덱스를 엽니다.
- **Build from .osm.pbf...**는 OpenStreetMap 추출 파일에서 경로 그래프를 만들고, **Open graph...**는 기존 그래프를 엽니다.
- **Search**는 인덱스에서 장소를 찾고, 지도를 클릭하면 그 지점을 역지오코딩합니다.
- 지도를 클릭하면 드래그할 수 있는 경로 경유지가 추가됩니다. 경로는 Car, Bike, Foot 프로필로 계산되며 그 단계가 나열됩니다.

구성 요소와 유닛

TsgcMapOfflineGeocoder, TsgcMapGeoIndexBuilder, TsgcMapOfflineRouter, TsgcMapRouteGraphBuilder, 그리고 OnCalculateRoute 이벤트에서 오프라인 라우터를 호출하는 TsgcMapRouteCalculator입니다.

실행 방법

IDE에서 **sgcMapsOfflineDemo.dpr**을 열고 컴파일하여 실행합니다. 데모는 sgcUI 설치 프로그램이 설정하는 라이브러리 경로를 통해 sgcMaps 유닛을 찾습니다. 수동으로 설치한 경우에는 먼저 Source 폴더를 라이브러리 경로에 추가하십시오.

데이터와 키

- API 키가 필요하지 않습니다.
- GeoNames 덤프(https://download.geonames.org/export/dump/)이며, ES.zip이나 cities15000.zip 같은 파일입니다. admin1CodesASCII.txt와 countryInfo.txt가 같은 폴더에 있으면 데모는 이 파일도 사용합니다. 인덱스는 덤프 옆에 .sqlite 확장자로 기록됩니다.
- andorra-latest.osm.pbf 같은 OpenStreetMap 추출 파일이며, Geofabrik 같은 다운로드 서비스에서 받을 수 있습니다. 그래프는 그 옆에 .sgcgraph 확장자로 기록됩니다.
- 빌드는 진행 표시줄이 있는 스레드에서 실행되며, 빌드가 끝날 때까지 폼을 닫을 수 없습니다.
- 인덱스, 그래프, 타일의 경로는 실행 파일 옆의 .ini 파일에 저장되고 시작할 때 다시 열립니다.
- 도구는 명령줄에서 같은 파일을 빌드합니다.

05 Static Render

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\05.StaticRender	sgcMapsStaticRender.dpr	콘솔	Delphi 7(온라인 타일과 PNG는 Delphi 2009부터, 경로는 Delphi XE2부터, GeoJSON은 Delphi XE6 부터)

보여 주는 내용

부모 창 없이 TsgcVCLMap을 만들고, 한 보기의 타일을 불러와 이미지로 저장하는 콘솔 애플리케이션입니다. 정적 지도 서비스나 보고서의 지도 이미지를 만드는 기반이 됩니다. 정적 지도 렌더링을 참조하십시오.

실행 방법

프로젝트를 컴파일한 다음 명령 프롬프트에서 실행합니다.

```
sgcMapsStaticRender --center lat,lon --zoom z --size WxH
  --provider osm|opentopo|esri|carto --out file.png|file.bmp
  [--marker lat,lon]... [--geojson file]
  [--route lat,lon;lat,lon[;...] --graph file.sgcgraph]
```

매개변수	의미
--center lat,lon	보기의 중심입니다. 필수이지만, --route를 지정하면 보기가 경로에 맞춰 중심이 잡히므로 생략할 수 있습니다.
--zoom z	줌 레벨입니다. 기본값은 13입니다.
--size WxH	이미지 크기이며 단위는 픽셀입니다. 기본값은 800x600입니다.
--provider	osm(기본값), opentopo, esri, carto입니다. 이 공급자들은 키가 필요 없습니다.
--out file	기록할 이미지입니다. 필수입니다. 확장자가 .bmp이면 BMP 파일을, 그 밖의 확장자이면 PNG 파일을 기록합니다(Delphi 2009 이전에는 BMP 파일).
--marker lat,lon	마커를 추가합니다. 반복해서 지정할 수 있습니다.
--geojson file	GeoJSON 파일의 피처를 그립니다(Delphi XE6 이상).
--route and --graph	오프라인 라우터와 지정한 .sgcgraph 파일로 경유지를 지나는 경로를 계산하고 그립니다(Delphi XE2 이상).

좌표는 소수점 구분자로 점을 사용합니다. 프로그램은 공급자, 보기, 저장된 파일, 저작자 표시를 출력하며, 성공하면 코드 0, 잘못된 매개변수이면 1, 예외가 발생하면 2로 종료합니다.

```
sgcMapsStaticRender --center 48.8584,2.2945 --zoom 15 --size 1024x768 --provider osm --out eiffel.png
```

데이터와 키

- API 키가 필요하지 않습니다.

- 타일이 실행 파일 옆의 **tile_cache** 폴더에 없으면 다운로드되므로, 처음 실행할 때는 네트워크 연결이 필요합니다.
- RouteGraphBuilder 도구는 --route에 사용할 .sgcgraph 파일을 빌드합니다.

06 FireMonkey

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\06.FMX	sgcMapsFMXDemo.dpr	FireMonkey	Delphi XE6(검색과 라우팅은 Delphi XE8 부터)

보여 주는 내용

코드에서 생성한 TsgcFMXMap이며, 탭이 네 개 있습니다.

탭	할 수 있는 작업
Search	입력하는 동안의 장소 제안과, 선택한 장소로 가는 **Route from here**입니다.
Location	**Locate me**와 **Follow my location**입니다.
오버레이	마커 500개의 마커 클러스터링, 히트맵, 드래그할 수 있는 마커, 편집 가능한 폴리곤, **Clear overlays**입니다.
View	기본 레이어, 레이어 전환기, 두 손가락 회전, 방위각, **Reset compass**, 그리고 문서 폴더에 sgcMapsFMX.png를 저장하는 **Snapshot to PNG**입니다.

구성 요소와 유닛

TsgcFMXMap, TsgcMapLocation, Photon을 사용하는 TsgcMapPlaces, OSRM을 사용하는 TsgcMapDirections입니다. 지도는 코드에서 생성되므로 디자인 타임 패키지 없이도 데모가 실행됩니다.

실행 방법

IDE에서 **sgcMapsFMXDemo.dpr**을 열고 대상 플랫폼을 선택한 다음 컴파일하여 실행합니다. Android에서는 위치 권한이 런타임에 요청됩니다(Delphi 10.3 이상).

데이터와 키

- API 키가 필요하지 않습니다. 기본 레이어는 키가 필요 없는 공급자이며, Places와 Directions는 자체 UserAgent를 설정합니다.
- 타일 캐시는 기기의 임시 폴더에 있으며, 이 폴더는 모든 플랫폼에서 쓰기가 가능합니다.
- 기기에 위치 센서가 없거나, 5초 안에 위치가 수신되지 않거나, 권한이 거부되면 데모는 마드리드의 시뮬레이션된 위치를 사용하여 나머지 데모가 계속 동작하게 합니다.

07 Vector Tiles

폴더	프로젝트	프레임워크	최소 버전
Demos\Maps\07.VectorTiles	sgcMapsVectorDemo.dpr	VCL	Delphi XE2(로컬 MBTiles는 Delphi XE7부터)

보여 주는 내용

TsgcVCLMap이 세 가지 소스와 세 가지 스타일로 그리는 벡터 타일입니다. 벡터 타일을 참조하십시오.

- 소스: **OpenFreeMap**(온라인, 키 불필요), **MapTiler vector**(키 필요), 벡터 타일이 들어 있는 **로컬 .mbtiles** 파일입니다.
- 스타일: 기본 제공 스타일, **Liberty** 샘플 스타일, 또는 임의의 MapLibre 스타일 JSON 파일입니다.
- **Retina**는 타일을 512픽셀로 그리며, 방위각 슬라이더나 Ctrl+드래그로 지도를 회전합니다.
- 상태 표시줄에는 줌, 그려진 레이블 수, OnTileError가 보고한 타일 오류가 표시됩니다.

실행 방법

IDE에서 **sgcMapsVectorDemo.dpr**을 열고 컴파일하여 실행합니다. 데모는 sgcUI 설치 프로그램이 설정하는 라이브러리 경로를 통해 sgcMaps 유닛을 찾습니다. 수동으로 설치한 경우에는 먼저 Source 폴더를 라이브러리 경로에 추가하십시오.

데이터와 키

- OpenFreeMap은 키가 필요 없습니다. 시작할 때 데모는 OpenFreeMap TileJSON에서 현재 타일 URL을 읽고(Delphi XE8 이상), 읽을 수 없으면 고정 URL로 대체합니다.
- MapTiler에는 MapTiler Cloud 웹 사이트에서 발급받은 키가 필요합니다.
- 로컬 파일에는 OpenMapTiles 스키마의 벡터 타일이 들어 있어야 하며, 기본 제공 스타일이 그리는 것이 이 스키마입니다.
- Liberty 샘플 스타일은 설치본의 QA 픽스처에서 읽습니다. 파일을 찾을 수 없으면 해당 옵션은 비활성화됩니다.

TsgcVCLMap

유닛: sgcMaps_VCL
VCL 지도 컨트롤입니다. TCustomControl에서 파생되며 타일과 오버레이를 직접 그리므로, 브라우저, WebView, JavaScript가 필요 없습니다. Lazarus에서는 같은 클래스가 LCL 지도 컨트롤이며, 타일을 다운로드할 수 없으므로 폴더의 타일만 표시합니다(Lazarus 참조).
이 컨트롤은 지도 모델의 컬렉션을 표시합니다. 타일 레이어를 **Layers**에 추가하고, **SetCenter**와 **Zoom**으로 보기를 설정하며, Add 메서드로 또는 컬렉션에 직접 마커와 도형을 추가합니다. 사용자는 마우스로 이동하고, 휠이나 확대/축소 버튼으로 줌하며, **Options.RotationEnabled**가 True이면 Ctrl+드래그 또는 오른쪽 드래그로 지도를 회전합니다.

속성

속성	형식	기본값	설명
CenterLatitude	Double		보기 중심의 위도이며 십진 도 단위입니다(WGS 84).
CenterLongitude	Double		보기 중심의 경도이며 십진 도 단위입니다(WGS 84).
Zoom	Integer	13	줌 레벨이며 Options.MinZoom과 Options.MaxZoom으로 제한됩니다.
Layers	TsgcMapLayers		타일 레이어이며 컬렉션 순서대로 그려집니다. 아래의 레이어 속성을 참조하십시오.
Markers	TsgcMapMarkers		마커: Latitude, Longitude, PopupText, TooltipText, Color, IconBitmap, AnchorX, AnchorY, Draggable, Visible, Tag, Data.
폴리라인	TsgcMapPolylines		선: AddPoint, Color, Width, Opacity, Dashed, Geodesic, Editable, Visible, Tag.
Polygons	TsgcMapPolygons		채워진 폴리곤이며 구멍을 가질 수 있습니다.
Rectangles	TsgcMapRectangles		북, 서, 남, 동 가장자리로 지정하는 채워진 직사각형입니다.
Circles	TsgcMapCircles		원: CenterLat, CenterLon, RadiusMeters, FillColor, StrokeColor, StrokeWidth, FillOpacity.
Labels	TsgcMapLabels		좌표에 고정된 텍스트 레이블입니다.
히트맵	TsgcMapHeatmaps		가중치가 있는 점의 히트맵이며, 그라데이션은 mhgDefault, mhgFire, mhgCool, mhgGrayscale입니다.
Options	TsgcMapOptions		보기 옵션: MinZoom, MaxZoom, TileSize, ShowZoomControls, ShowScaleBar, ShowAttribution, AttributionText, BackgroundColor, DragEnabled, ZoomEnabled, Bearing, RotationEnabled, ShowLayerSwitcher.

MarkerOptions	TsgcMapMarkerOptions		기본 마커 크기, 색상, 글꼴, 그리고 Clustering(Enabled, GridSize, MinClusterSize, Color, FontColor, ZoomOnClick, MaxZoom).
ActivePopupMarker	TsgcMapMarker		팝업이 열려 있는 마커이며, 없으면 nil입니다. 읽기 전용입니다.
DrawnMarkerCount	Integer		마지막 그리기에서 그려진 마커입니다. 읽기 전용입니다.
DrawnClusterCount	Integer		마지막 그리기에서 그려진 클러스터입니다. 읽기 전용입니다.
DrawnLabelCount	Integer		마지막 그리기에서 그려진 레이블입니다. 읽기 전용입니다.
LastRenderMissingTiles	Integer		마지막 그리기에서 아직 사용할 수 없었던 타일입니다. 읽기 전용입니다.

레이어 속성

Layers의 각 항목은 TsgcMapLayer입니다.

속성	형식	기본값	설명
Name	string		레이어 전환기에 표시되는 이름입니다.
TileSourceType	TsgcMapTileSourceType	mtsFolder	mtsFolder(z/x/y 폴더), mtsMBTiles, mtsHTTP, mtsVectorMBTiles, mtsVectorHTTP 중 하나입니다.
TileSourcePath	string		오프라인 레이어의 폴더 또는 MBTiles 파일입니다.
URLTemplate	string		온라인 레이어의 타일 URL이며, {z}, {x}, {y} 자리 표시자를 사용합니다.
CacheFolder	string		온라인 레이어의 디스크 캐시입니다. 비워 두면 캐시를 사용하지 않습니다.
공급자	TsgcMapTileProvider	mtpCustom	내장 공급자입니다. 레이어는 다른 설정 없이 해당 타일을 다운로드하며, sgcMapApplyProvider는 공급자의 소스 형식, URL 템플릿, 최대 줌, 저작자 표시도 레이어에 복사합니다.
ApiKey	string		키가 필요한 공급자(MapTiler, Thunderforest, Stadia)의 키입니다.
Attribution	string		레이어의 저작자 표시 텍스트입니다.
Opacity	Double		0에서 1까지의 불투명도입니다.
Visible	Boolean	True	레이어를 표시하거나 숨깁니다.
MinZoom, MaxZoom	Integer	0, 19	레이어 타일의 줌 범위입니다.
IsBaseLayer	Boolean	True	기본 레이어이면 True이고, 위에 그려지는 오버레이이면 False입니다. SetLayerVisible이나 레이어

			전환기에서 기본 레이어를 표시하면 다른 기본 레이어는 숨겨집니다.
Retina	Boolean	False	고해상도 타일을 요청합니다.
VectorStyleFile	string		벡터 레이어의 MapLibre 또는 Mapbox GL 스타일입니다.

메서드

메서드	설명
SetCenter(aLat, aLon: Double)	보기의 중심을 이동합니다.
ZoomIn, ZoomOut	줌을 한 레벨 변경합니다.
FitBounds(aNorthLat, aWestLon, aSouthLat, aEastLon: Double; aPaddingPx: Integer = 20)	영역이 보이도록 보기의 중심과 줌을 맞춥니다.
FitToMarkers(aPaddingPx: Integer = 20)	모든 마커가 보이도록 보기의 중심과 줌을 맞춥니다.
FitToPolyline(aPolyline: TsgcMapPolyline; aPaddingPx: Integer = 20)	폴리라인(예: 경로)이 보이도록 보기의 중심과 줌을 맞춥니다.
AddMarker(aLat, aLon: Double; const aPopup: string = ''; const aTooltip: string = '')	선택적인 팝업 텍스트와 툴팁이 있는 마커를 추가합니다.
ClearMarkers	모든 마커를 제거합니다.
AddPolyline: TsgcMapPolyline	빈 폴리라인을 추가하며, 점은 AddPoint로 추가합니다.
AddPolygon: TsgcMapPolygon	빈 폴리곤을 추가합니다.
AddCircle(aLat, aLon, aRadiusMeters: Double): TsgcMapCircle	원을 추가합니다.
AddRectangle(aNorthLat, aWestLon, aSouthLat, aEastLon: Double): TsgcMapRectangle	직사각형을 추가합니다.
AddLabel(aLat, aLon: Double; const aText: string): TsgcMapLabel	텍스트 레이블을 추가합니다.
ShowPopup(aMarker: TsgcMapMarker), HidePopup	마커의 팝업을 열거나 닫습니다.
MarkerAtScreen(aX, aY): TsgcMapMarker	컨트롤의 한 지점에 있는 마커이며, 없으면 nil입니다.
ClusterAtScreen(aX, aY; out aCount: Integer; out aLat, aLon: Double): Boolean	컨트롤의 한 지점에 클러스터가 있으면 True이며, 크기와 위치를 함께 반환합니다.
SetLayerVisible(aIndex: Integer; aVisible: Boolean)	레이어를 표시하거나 숨깁니다. 기본 레이어를 표시하면 다른 기본 레이어가 숨겨지며, 레이어 전환기도 같은 동작을 합니다.
RefreshTiles	보기의 타일을 다시 불러옵니다.
AttributionText: string	표시되는 레이어의 저작자 표시입니다.

RenderToBitmap(aBitmap: TBitmap; aIncludeControls: Boolean = False)	현재 보기를 비트맵에 그립니다.
SaveToPNG, SaveToBMP(const aFileName: string; aWidth, aHeight: Integer; aIncludeControls: Boolean = False)	보기를 지정한 크기의 PNG 또는 BMP 파일로 렌더링합니다. 부모 창 없이도 동작합니다.

이벤트

이벤트	설명
OnMapClick	지도 클릭이며 위도와 경도가 함께 전달됩니다.
OnMarkerClick	마커 클릭입니다.
OnMarkerDragStart, OnMarkerDrag, OnMarkerDragEnd	사용자가 Draggable 속성이 True인 마커를 드래그합니다.
OnShapeEdited	사용자가 편집 가능한 폴리라인, 폴리곤, 직사각형의 꼭짓점을 이동했습니다.
OnClusterClick	클러스터 클릭이며 크기와 위치가 함께 전달됩니다. 클릭을 직접 처리하려면 aHandled를 True로 설정하십시오.
OnLayerChanged	사용자가 레이어 전환기에서 다른 레이어를 선택했습니다.
OnZoomChanged	줌 레벨이 변경되었습니다.
OnBearingChanged	지도가 회전되었습니다.
OnTileError	타일을 불러올 수 없었으며, 레이어, 타일 좌표, 오류가 함께 전달됩니다.

예제: 지도 설정

OpenStreetMap 레이어를 추가하고, 보기의 중심을 맞추고, 마커, 선, 원을 추가하는 OnCreate 핸들러입니다.

예제: 이벤트와 스냅숏

참고 항목

- TsgcFMXMap
- 빠른 시작 VCL
- TsgcMapRouteCalculator

TsgcFMXMap

유닛: sgcMaps_FMX
Delphi XE6부터 사용할 수 있는 FireMonkey 지도 컨트롤입니다. FireMonkey TControl에서 파생되며, TsgcVCLMap과 지도 모델을 공유합니다. 같은 Layers, Markers, Polylines, Polygons, Rectangles, Circles, Labels, Heatmaps, Options, MarkerOptions, 같은 메서드, 같은 이벤트를 가집니다.

TsgcVCLMap과의 차이점

- 터치 입력: 손가락 하나로 이동하고, 핀치로 확대/축소하며, **Options.RotationEnabled**가 True이면 손가락 두 개로 지도를 회전합니다.
- **MarkerAtScreen**과 **ClusterAtScreen**은 FireMonkey의 좌표 형식인 Single 좌표를 받습니다.
- **RenderToBitmap**은 FMX TBitmap을 받으며, 컨트롤은 **HitTest**(기본값 True)를 게시합니다.
- 모바일 플랫폼에서는 애플리케이션 폴더가 읽기 전용입니다. 레이어의 **CacheFolder**는 TPath.GetCachePath 같은 쓰기 가능한 폴더에 두고, 스냅숏은 TPath.GetDocumentsPath에 저장하십시오.

속성

속성	형식	기본값	설명
CenterLatitude	Double		보기 중심의 위도이며 십진 도 단위입니다(WGS 84).
CenterLongitude	Double		보기 중심의 경도이며 십진 도 단위입니다(WGS 84).
Zoom	Integer	13	줌 레벨이며 Options.MinZoom과 Options.MaxZoom으로 제한됩니다.
Layers	TsgcMapLayers		타일 레이어이며 컬렉션 순서대로 그려집니다. 아래의 레이어 속성을 참조하십시오.
Markers, Polylines, Polygons, Rectangles, Circles, Labels, Heatmaps	컬렉션		오버레이이며 TsgcVCLMap과 같습니다.
Options, MarkerOptions	TsgcMapOptions, TsgcMapMarkerOptions		보기, 마커, 클러스터링 옵션이며 TsgcVCLMap과 같습니다.
Canvas	TCanvas		컨트롤의 FireMonkey 캔버스입니다. 읽기 전용입니다.
ActivePopupMarker	TsgcMapMarker		팝업이 열려 있는 마커이며, 없으면 nil입니다. 읽기 전용입니다.
DrawnMarkerCount	Integer		마지막 그리기에서 그려진 마커입니다. 읽기 전용입니다.
DrawnClusterCount	Integer		마지막 그리기에서 그려진 클러스터입니다. 읽기 전용입니다.
DrawnLabelCount	Integer		마지막 그리기에서 그려진 레이블입니다. 읽기 전용입니다.

LastRenderMissingTiles	Integer		마지막 그리기에서 아직 사용할 수 없었던 타일입니다. 읽기 전용입니다.

메서드

메서드	설명
SetCenter(aLat, aLon: Double)	보기의 중심을 이동합니다.
ZoomIn, ZoomOut	줌을 한 레벨 변경합니다.
FitBounds(aNorthLat, aWestLon, aSouthLat, aEastLon: Double; aPaddingPx: Integer = 20)	영역이 보이도록 보기의 중심과 줌을 맞춥니다.
FitToMarkers(aPaddingPx: Integer = 20)	모든 마커가 보이도록 보기의 중심과 줌을 맞춥니다.
FitToPolyline(aPolyline: TsgcMapPolyline; aPaddingPx: Integer = 20)	폴리라인(예: 경로)이 보이도록 보기의 중심과 줌을 맞춥니다.
AddMarker(aLat, aLon: Double; const aPopup: string = ''; const aTooltip: string = '')	선택적인 팝업 텍스트와 툴팁이 있는 마커를 추가합니다.
ClearMarkers	모든 마커를 제거합니다.
AddPolyline: TsgcMapPolyline	빈 폴리라인을 추가하며, 점은 AddPoint로 추가합니다.
AddPolygon: TsgcMapPolygon	빈 폴리곤을 추가합니다.
AddCircle(aLat, aLon, aRadiusMeters: Double): TsgcMapCircle	원을 추가합니다.
AddRectangle(aNorthLat, aWestLon, aSouthLat, aEastLon: Double): TsgcMapRectangle	직사각형을 추가합니다.
AddLabel(aLat, aLon: Double; const aText: string): TsgcMapLabel	텍스트 레이블을 추가합니다.
ShowPopup(aMarker: TsgcMapMarker), HidePopup	마커의 팝업을 열거나 닫습니다.
MarkerAtScreen(aX, aY): TsgcMapMarker	컨트롤의 한 지점에 있는 마커이며, 없으면 nil입니다.
ClusterAtScreen(aX, aY; out aCount: Integer; out aLat, aLon: Double): Boolean	컨트롤의 한 지점에 클러스터가 있으면 True이며, 크기와 위치를 함께 반환합니다.
SetLayerVisible(aIndex: Integer; aVisible: Boolean)	레이어를 표시하거나 숨깁니다. 기본 레이어를 표시하면 다른 기본 레이어가 숨겨지며, 레이어 전환기도 같은 동작을 합니다.
RefreshTiles	보기의 타일을 다시 불러옵니다.
AttributionText: string	표시되는 레이어의 저작자 표시입니다.
RenderToBitmap(aBitmap: TBitmap; aIncludeControls: Boolean = False)	현재 보기를 비트맵에 그립니다.

SaveToPNG, SaveToBMP(const aFileName: string; aWidth, aHeight: Integer; aIncludeControls: Boolean = False)	보기를 지정한 크기의 PNG 또는 BMP 파일로 렌더링합니다. 부모 창 없이도 동작합니다.

이벤트

이벤트	설명
OnMapClick	지도 클릭이며 위도와 경도가 함께 전달됩니다.
OnMarkerClick	마커 클릭입니다.
OnMarkerDragStart, OnMarkerDrag, OnMarkerDragEnd	사용자가 Draggable 속성이 True인 마커를 드래그합니다.
OnShapeEdited	사용자가 편집 가능한 폴리라인, 폴리곤, 직사각형의 꼭짓점을 이동했습니다.
OnClusterClick	클러스터 클릭이며 크기와 위치가 함께 전달됩니다. 클릭을 직접 처리하려면 aHandled를 True로 설정하십시오.
OnLayerChanged	사용자가 레이어 전환기에서 다른 레이어를 선택했습니다.
OnZoomChanged	줌 레벨이 변경되었습니다.
OnBearingChanged	지도가 회전되었습니다.
OnTileError	타일을 불러올 수 없었으며, 레이어, 타일 좌표, 오류가 함께 전달됩니다.

예제: 지도 설정

OpenStreetMap 레이어를 애플리케이션의 캐시 폴더에 캐시와 함께 추가하고, 회전과 클러스터링을 켜고, 스냅숏을 문서 폴더에 저장하는 OnCreate 핸들러입니다.

참고 항목

- TsgcVCLMap
- 빠른 시작 FMX
- TsgcMapLocation

타일, 레이어, 공급자

지도는 **타일**로 그려집니다. 타일은 기본적으로 너비 256픽셀의 정사각형 이미지이며, 각 줌 레벨에서 세계 전체를 덮습니다. 줌 0에서는 타일 하나가 세계 전체를 보여 주고, 레벨이 하나 올라갈 때마다 각 방향의 타일 수가 두 배가 됩니다. 타일은 줌, x, y로 주소를 지정하며, 이는 OpenStreetMap과 거의 모든 타일 서버가 사용하는 z/x/y 방식입니다.
TsgcVCLMap 과 TsgcFMXMap 의 **Layers** 컬렉션은 타일의 출처를 지정합니다. 각 항목은 **TsgcMapLayer**(유닛 sgcMaps_Classes)이며, **TileSourceType**이 소스를 선택합니다.

TileSourceType	타일의 출처	경로 또는 URL
mtsFolder	디스크에 있는 .png, .jpg, .jpeg 파일의 z/x/y 폴더입니다.	TileSourcePath
mtsMBTiles	래스터 MBTiles 파일입니다.	TileSourcePath
mtsHTTP	URL 템플릿을 통한 타일 서버입니다.	URLTemplate
mtsVectorHTTP	벡터 타일 서버입니다. 벡터 타일을 참조하십시오.	URLTemplate
mtsVectorMBTiles	벡터 MBTiles 파일입니다. 벡터 타일을 참조하십시오.	TileSourcePath

폴더와 MBTiles 소스는 네트워크 연결 없이 동작합니다. 준비 방법은 오프라인 타일과 타일 다운로더에서 설명합니다.

내장 공급자

레이어의 **Provider** 속성은 **TsgcMapTileProvider**의 내장 타일 서버 중 하나를 지정합니다. 이 속성을 설정한 후 **sgcMapApplyProvider**(유닛 sgcMaps_Providers)를 호출하십시오. 이 함수는 TileSourceType을 mtsHTTP로 설정하고, 공급자의 URL 템플릿과 최대 줌을 레이어에 복사하며, Attribution이 비어 있으면 채웁니다. Provider가 mtpCustom이 아닌 레이어는 이 호출이 없어도 해당 공급자의 타일을 다운로드하지만, 그 경우 MaxZoom은 기본값인 19를 유지하고 레이어 속성에 공급자 설정이 표시되지 않습니다.

공급자	API 키	최대 줌	레티나 타일
mtpOpenStreetMap	아니요	19	아니요
mtpOpenTopoMap	아니요	17	아니요
mtpEsriWorldImagery	아니요	19	아니요
mtpEsriWorldStreet	아니요	19	아니요
mtpCartoLight	아니요	20	예
mtpCartoDark	아니요	20	예
mtpMapTilerStreets	예	22	예
mtpMapTilerSatellite	예	20	예
mtpThunderforestCycle	예	22	예
mtpThunderforestTransport	예	22	예
mtpStadiaAlidadeSmooth	예	20	예
mtpStadiaAlidadeSmoothDark	예	20	예

키가 필요한 공급자는 레이어의 **ApiKey** 속성에 키를 설정하십시오. 공개 서버에는 사용 정책이 있습니다. 예를 들어 tile.openstreetmap.org는 과도한 사용과 대량 다운로드를 금지하므로, 운영 애플리케이션은 상용 공급자나 자체 타일 서버를 사용해야 합니다.

URL 템플릿

그 밖의 서버에는 TileSourceType을 mtsHTTP로 설정하고 **URLTemplate**을 작성하십시오. 다음 자리 표시자가 모든 타일마다 치환됩니다.

자리 표시자	치환되는 값
{z}, {x}, {y}	줌과 타일의 열 및 행입니다.
{s}	타일마다 순환하는 하위 도메인 문자이며, 요청이 서버들에 분산되도록 합니다. 공급자가 다른 문자를 정의하지 않으면 a, b, c입니다.
{r}	레이어의 Retina가 True이면 @2x이고, 그렇지 않으면 비어 있습니다.
{apikey}	레이어의 ApiKey입니다.

같은 치환을 **sgcMapExpandTileURL**로도 사용할 수 있습니다. **sgcMapProviderNames**는 문자열 목록을 공급자 이름으로 채우고, **sgcMapGetProviderInfo**는 이름, URL 템플릿, 하위 도메인, 저작자 표시, 최대 줌, 그리고 공급자가 API 키를 필요로 하는지와 레티나 타일을 지원하는지를 담은 **TsgcMapProviderInfo** 레코드를 반환합니다.

여러 레이어

레이어는 **기본 레이어**(IsBaseLayer가 True이며, 기본값입니다) 또는 **오버레이** 중 하나입니다. 지도는 처음으로 보이는 기본 레이어를 그린 다음, 그 위에 보이는 모든 오버레이를 컬렉션 순서대로 그립니다. Opacity는 레이어의 투명도를 설정하고, MinZoom과 MaxZoom은 레이어가 그려지는 줌 레벨을 제한하며, Retina는 공급자에게 두 배 해상도의 타일을 요청합니다. 사용자가 지도에서 레이어를 선택할 수 있도록 하려면 **Options.ShowLayerSwitcher**를 True로 설정하십시오. 기본 레이어는 라디오 항목처럼 동작하여, **SetLayerVisible**로 하나를 표시하면 다른 기본 레이어는 숨겨집니다. **OnLayerChanged** 이벤트는 표시되거나 숨겨진 레이어를 알려 줍니다.

타일 캐시

다운로드한 모든 타일을 z/x/y 트리로 디스크에 보관하려면 **CacheFolder**를 설정하십시오. 캐시에 이미 있는 타일은 서버에 요청하지 않고 그려지므로, 사용자가 한 번 본 지도는 연결이 없어도 계속 동작합니다. 지도는 최근 타일을 메모리에도 보관하며, **RefreshTiles**는 보기의 타일을 다시 불러옵니다. 타일을 불러올 수 없으면 레이어, 타일 주소, 오류 텍스트와 함께 **OnTileError**가 발생하며, **LastRenderMissingTiles**는 마지막 그리기에서 누락된 타일 수를 셉니다.

지원 현황

프레임워크	최소 버전
VCL, 폴더 타일	Delphi 7
VCL, 서버의 타일(공급자, URL 템플릿, 캐시)	Delphi 2009
VCL과 FireMonkey, MBTiles	Delphi XE7 (FireDAC SQLite)
FireMonkey	Delphi XE6
Lazarus (LCL)	폴더 타일만 지원

Delphi 2009부터 XE7까지는 타일을 Indy 10으로 다운로드하며, https에는 실행 파일 옆에 OpenSSL 1.0.2 라이브러리(libeay32.dll과 ssleay32.dll)가 필요합니다. Delphi XE8부터는 지도가 운영 체제의 TLS 지원을 사용하는 THTTPClient를

사용하므로 DLL이 필요 없습니다. Delphi 7, Delphi 2007, Lazarus에는 타일 다운로드가 없으므로 타일 폴더를 사용하십시오.

참고 항목

- TsgcVCLMap
- 벡터 타일
- 오프라인 타일과 타일 다운로더

벡터 타일

벡터 타일은 도로, 건물, 수역, 지명을 픽셀이 아닌 지오메트리로 Mapbox Vector Tile(MVT) 형식에 저장합니다. sgcMaps는 타일을 디코딩하여 **스타일**에 따라 직접 그립니다. 스타일은 MapLibre 또는 Mapbox GL 스타일 JSON으로, 어떤 피처를 어떤 색상과 너비로 어떤 줌 레벨에서 그릴지를 지정합니다. 따라서 하나의 벡터 타일 세트가 스타일에 따라 밝게, 어둡게, 또는 미니멀하게 보일 수 있으며, 타일은 어떤 픽셀 밀도에서도 선명하게 유지됩니다.

벡터 레이어

벡터 소스 형식 중 하나로 지도에 레이어를 추가합니다.

- **mtsVectorHTTP**: 서버의 타일입니다. URLTemplate은 래스터 타일과 같은 자리 표시자를 사용하며(예: https://api.maptiler.com/tiles/v3/{z}/{x}/{y}.pbf?key={apikey}), ApiKey가 {apikey}를 채웁니다.
- **mtsVectorMBTiles**: TileSourcePath에 설정한 로컬 벡터 MBTiles 파일의 타일입니다. tilemaker나 planetiler로 만든 파일이 그 예입니다.

VectorStyleFile은 UTF-8로 저장한 스타일 JSON 파일의 경로입니다. 비어 있으면 지도는 내장 스타일을 사용하는데, 이는 **OpenMapTiles** 스키마(수역, 토지 피복, 토지 이용, 공원, 건물, 도로, 경계, 수로, 그리고 지명, 도로명, 수역명)를 위한 밝은 스타일입니다. 이 스키마는 MapTiler, OpenFreeMap, 대부분의 OpenStreetMap 벡터 타일이 사용하는 것입니다. **Retina**를 True로 설정하면 고 DPI 화면을 위해 타일이 256픽셀 대신 512픽셀로 렌더링됩니다. 스타일의 symbol 레이어에서 나온 레이블은 타일별로 수집되며, 지도가 회전한 경우에도 지도가 똑바로 세워 그립니다.

벡터 타일 소스

레이어는 **TsgcMapTileSource_MVT**(유닛 sgcMaps_TileSource_MVT)를 만듭니다. 지도 밖에서 타일을 렌더링하려면 직접 하나 만들 수도 있습니다.

멤버	설명
SourceKind	mvsHTTP 또는 mvsMBTiles입니다.
URLTemplate, ApiKey, UserAgent	mvsHTTP의 서버입니다.
Path	mvsMBTiles의 .mbtiles 파일입니다.
StyleJSON, StyleFile	스타일입니다. StyleJSON이 StyleFile보다 우선하며, 둘 다 비어 있으면 Open이 내장 스타일을 불러옵니다.
TileSize	256(기본값) 또는 512픽셀입니다.
MaxDataZoom	데이터에 저장된 가장 높은 줌이며 기본값은 14입니다. 더 깊은 줌 레벨은 이 레벨의 타일에서 그려집니다.
CacheSize	메모리에 보관하는 렌더링된 타일의 수이며 기본값은 256입니다.
Open, Close	Open은 리더를 만들고 스타일을 불러옵니다. 스타일이 유효하지 않으면 예외를 발생시킵니다.
GetTile	타일을 PNG 이미지로 렌더링하여 반환합니다.
GetTileLabels	타일의 레이블을 위치와 함께 반환합니다.
ClearCache	렌더링된 타일을 버립니다. 스타일이나 TileSize를 변경한 후 호출하십시오.
Style	사용 중인 TsgcMapVectorStyle이며, Open 후에 유효합니다.

디코딩과 스타일

유닛 **sgcMaps_MVT**에 기본 요소들이 들어 있습니다. **TsgcMapVectorTile**은 바이트, 스트림 또는 파일에서 (원시 형식이거나 gzip 또는 zlib으로 압축된) 타일을 레이어(**TsgcMapVectorLayer**)와 피처(**TsgcMapVectorFeature**)로 디코딩하며, 피처는 지오메트리 형식, 태그, 지오메트리를 가집니다. **TsgcMapVectorStyle**은 LoadFromJSON 또는 LoadFromFile로 스타일을 불러오고, LoadDefault로 내장 스타일을 불러옵니다.
스타일 지원은 background, fill, line, symbol 레이어 유형을 다루며, raster, fill-extrusion, circle, heatmap, hillshade 레이어는 무시됩니다. 필터, 줌 함수, 일반적인 표현식(get, has, match, case, step, interpolate, coalesce와 비교 연산자)이 평가됩니다. 아이콘, 스프라이트, 패턴, 텍스트 헤일로, 글리프는 사용되지 않습니다.

지원 현황

프레임워크	최소 버전
VCL과 FireMonkey, 서버의 벡터 타일	Delphi XE2(FireMonkey는 XE6부터)
VCL과 FireMonkey, 벡터 MBTiles	Delphi XE7 (FireDAC SQLite)
Lazarus (LCL)	사용할 수 없음

참고 항목

- 타일, 레이어, 공급자
- TsgcVCLMap
- TsgcFMXMap

마커와 클러스터링

마커는 지도의 한 지점에 핀을 꽂습니다. TsgcVCLMap과 TsgcFMXMap의 **Markers** 컬렉션은 **TsgcMapMarker** 항목을 담고 있으며, **AddMarker**는 팝업 텍스트와 툴팁이 있는 마커를 한 번의 호출로 추가합니다.

마커 속성

속성	설명
Latitude, Longitude	마커의 위치입니다.
PopupText	ShowPopup으로 여는 팝업의 텍스트이며, 보통 OnMarkerClick에서 호출합니다. HidePopup은 팝업을 닫고, ActivePopupMarker는 팝업이 열려 있는 마커입니다.
TooltipText	마우스가 마커 위에 있는 동안 표시되는 텍스트입니다.
Color	기본 핀의 색상입니다. MarkerOptions.DefaultColor, DefaultWidth, DefaultHeight는 모든 마커의 핀을 설정합니다.
IconBitmap	사용자 지정 아이콘이며, PNG(Delphi 2009 이상 및 Lazarus), JPEG 또는 BMP 이미지에서 LoadFromFile 또는 LoadFromStream으로 불러옵니다. 폼에 저장되므로 디자인 타임에 설정한 아이콘은 런타임에 파일이 필요 없습니다.
AnchorX, AnchorY	좌표에 닿는 아이콘의 지점이며, 아이콘 너비와 높이에 대한 비율입니다. 기본값인 0.5와 1은 아래쪽 가운데입니다.
Draggable	사용자가 마커를 이동할 수 있게 합니다. OnMarkerDragStart, OnMarkerDrag, OnMarkerDragEnd가 드래그를 따라 발생합니다.
Visible, Tag, Data	표시 여부, 그리고 사용자 자신의 데이터를 위한 정수 하나와 문자열 하나입니다.

MarkerAtScreen은 컨트롤의 한 지점에 있는 마커를 반환하고, **FitToMarkers**는 모든 마커에 맞게 보기를 확대하고 중심을 맞추며, **ClearMarkers**는 모든 마커를 제거합니다.

클러스터링

지도를 축소하면 수천 개의 마커 때문에 지도가 느려지고 읽기 어려워집니다. **MarkerOptions.Clustering.Enabled**를 True로 설정하면 가까운 마커가 하나의 원으로 합쳐져 그 안에 있는 마커의 수를 표시합니다. 화면은 **GridSize**픽셀의 정사각형 셀로 나뉘며, 마커가 **MinClusterSize**개 이상인 셀은 마커들의 중심에 그려지는 클러스터가 됩니다. **MaxZoom**(기본값 18)보다 높은 줌에서는 마커가 클러스터링되지 않습니다. **Color**와 **FontColor**는 원의 모양을 설정합니다.
클러스터를 클릭하면 마커 수와 클러스터의 위치와 함께 **OnClusterClick**이 발생합니다. 핸들러가 aHandled를 False로 두고 **ZoomOnClick**이 True(기본값)이면, 지도는 클러스터의 마커가 분리될 때까지 확대합니다. **ClusterAtScreen**은 컨트롤의 한 지점에 있는 클러스터를 찾고, **DrawnClusterCount**와 **DrawnMarkerCount**는 마지막 그리기에서 그려진 수를 보고합니다.
대량으로 추가할 때는 Markers.BeginUpdate와 Markers.EndUpdate로 감싸면, 마커마다 다시 그리지 않고 지도를 한 번만 다시 그립니다. 클러스터링 함수 자체인 **sgcMapClusterPoints**는 sgcMaps_Geometry에서 공개되어 있어, 직접 만든 화면 좌표에 사용할 수 있습니다.

지원 현황

프레임워크	최소 버전

VCL	Delphi 7
FireMonkey	Delphi XE6
Lazarus (LCL)	예

참고 항목

- TsgcVCLMap
- 도형과 레이블
- 히트맵

도형과 레이블

마커 외에도 TsgcVCLMap과 TsgcFMXMap은 타일 위에 폴리라인, 폴리곤, 직사각형, 원, 텍스트 레이블을 그립니다. 각 종류에는 자체 컬렉션이 있으며, 컨트롤에는 항목을 만들어 반환하는 Add 메서드가 있습니다.

컬렉션	Add 메서드	항목	주요 속성
Polylines	AddPolyline	TsgcMapPolyline	AddPoint, ClearPoints, PointCount, Color, Width, Opacity, Dashed, Geodesic, Editable
Polygons	AddPolygon	TsgcMapPolygon	AddPoint, Holes, FillColor, FillOpacity, StrokeColor, StrokeWidth, Editable
Rectangles	AddRectangle	TsgcMapRectangle	NorthLatitude, WestLongitude, SouthLatitude, EastLongitude, FillColor, FillOpacity, StrokeColor, StrokeWidth, Editable
Circles	AddCircle	TsgcMapCircle	CenterLat, CenterLon, RadiusMeters, FillColor, FillOpacity, StrokeColor, StrokeWidth
Labels	AddLabel	TsgcMapLabel	Latitude, Longitude, Text, FontSize, FontColor, BackgroundColor, ShowBackground

모든 항목에는 Visible도 있으며, 도형에는 사용자 자신이 쓸 수 있는 Tag가 있습니다.

폴리라인

폴리라인은 지도에서 점들을 직선으로 연결합니다. **Dashed**를 사용하면 점선으로 그려지고, **Geodesic**을 사용하면 각 구간이 지구상의 최단 경로인 대원을 따르며, 이는 항공편과 장거리에 알맞은 선택입니다. **FitToPolyline**은 폴리라인에 맞게 보기를 맞추며, 길 찾기 및 라우팅 구성 요소의 경로도 폴리라인으로 추가됩니다.

구멍이 있는 폴리곤

폴리곤은 바깥 링의 안쪽이 채워집니다. **Holes**는 비어 있는 안쪽 링을 추가합니다. **AddHole**은 새 구멍의 인덱스를 반환하고, **AddPoint**는 그 구멍에 꼭짓점을 추가합니다. 구멍이 있는 GeoJSON 및 KML 폴리곤도 같은 방식으로 가져옵니다.

편집

폴리라인, 폴리곤, 직사각형에서 **Editable**을 True로 설정하면 사용자가 마우스로 모양을 바꿀 수 있습니다. 꼭짓점은 끌어서 이동하고, 구간 가운데의 핸들은 새 꼭짓점을 삽입하며, 꼭짓점을 마우스 오른쪽 버튼으로 클릭하면 삭제됩니다. 편집이 끝나면 지도는 편집된 항목과 함께 **OnShapeEdited**를 발생시킵니다. 원과 레이블은 편집할 수 없습니다.

지원 현황

프레임워크	최소 버전
VCL	Delphi 7
FireMonkey	Delphi XE6
Lazarus (LCL)	예

참고 항목

- TsgcVCLMap
- 마커와 클러스터링
- 지오메트리 함수
- GeoJSON, GPX, KML

히트맵

히트맵은 가중치가 있는 점이 어디에 몰려 있는지 보여 줍니다. 센서 측정값, 매출, 사건, GPS 위치 등이 그 예입니다. 각 점은 자신의 가중치를 원 안에 퍼뜨리고, 가중치는 합산되며, 합계는 차가운 색에서 뜨거운 색으로 칠해집니다.
TsgcVCLMap과 TsgcFMXMap의 **Heatmaps** 컬렉션은 **TsgcMapHeatmap** 항목을 담고 있습니다. 점은 **AddPoint**(위도, 경도, 가중치)로 추가하며, 가중치를 생략하면 1입니다. Count, Points, Clear로 점에 접근할 수 있고, **TsgcMapHeatmapPoint** 레코드가 점 하나를 담습니다.

속성	기본값	설명
Radius	25	모든 점의 영향 반경이며 단위는 픽셀입니다.
Opacity	0.6	가장 뜨거운 색의 불투명도입니다.
MaxIntensity	0	가장 뜨거운 색으로 그려지는 강도입니다. 0이면 색상이 그려진 가장 높은 강도에 맞춰지며, 고정 값은 보기 사이에서 색상을 비교할 수 있게 유지합니다.
Gradient	mhgDefault	mhgDefault(파랑, 청록, 라임, 노랑, 빨강), mhgFire, mhgCool 또는 mhgGrayscale입니다.
Visible	True	히트맵을 표시하거나 숨깁니다.

히트맵은 화면 픽셀 단위로 그려지므로 모양이 줌을 따릅니다. 확대하면 스팟이 분리됩니다. 렌더링 함수인 sgcMaps_Geometry의 **sgcMapRenderHeatmap**도 공개되어 있어, 직접 만든 화면 좌표로 ARGB 버퍼를 채울 수 있습니다.

지원 현황

프레임워크	최소 버전
VCL	Delphi 7
FireMonkey	Delphi XE6
Lazarus (LCL)	예

참고 항목

- TsgcVCLMap
- 마커와 클러스터링

GeoJSON, GPX, KML

sgcMaps는 지리 데이터의 일반적인 교환 형식 세 가지를 읽고 씁니다. 가져오기 함수는 지도의 컬렉션에 항목을 추가하고, 내보내기 함수는 지도의 항목을 기록하므로, TsgcVCLMap이나 TsgcFMXMap의 어떤 컬렉션도 전달할 수 있습니다.

형식	유닛	가져오기	내보내기
GeoJSON (RFC 7946)	sgcMaps_GeoJSON	Point와 MultiPoint는 마커로, LineString과 MultiLineString은 폴리라인으로, Polygon과 MultiPolygon은 구멍이 있는 폴리곤으로 변환되며, GeometryCollection, Feature, FeatureCollection도 지원합니다.	simplestyle 속성이 있는 FeatureCollection입니다. 직사각형은 폴리곤으로 기록됩니다.
GPX	sgcMaps_GPX	웨이포인트는 마커로, 트랙과 경로는 폴리라인으로 변환됩니다.	마커는 웨이포인트로, 폴리라인은 트랙으로 기록됩니다. ExportPolygonsAsTracks가 True이면 폴리곤과 직사각형은 닫힌 트랙으로 기록됩니다.
KML	sgcMaps_KML	Point는 마커로, LineString은 폴리라인으로, 내부 경계가 있는 Polygon은 폴리곤으로 변환되며, 인라인 스타일과 공유 스타일의 색상이 적용됩니다.	항목마다 인라인 스타일이 있는 Placemark 하나입니다. 직사각형은 폴리곤으로 기록됩니다.

각 형식에는 LoadFromString, LoadFromFile, AsText, SaveToStream, SaveToFile을 가진 클래스(**TsgcMapGeoJSON**, **TsgcMapGPX**, **TsgcMapKML**)가 있으며, 같은 작업을 일반 함수로도 제공합니다. **sgcMapLoadGeoJSON**, **sgcMapLoadGeoJSONFile**, **sgcMapGeoJSONAsText**, **sgcMapSaveGeoJSONFile**, 그리고 그에 대응하는 GPX와 KML 함수입니다. nil로 전달한 컬렉션은 건너뜁니다.

GPX 트랙 통계

TsgcMapGPX는 마지막으로 불러온 파일의 모든 트랙에 대한 통계를 보관합니다. **TrackStatsCount**와 **TrackStats**는 이름, 점의 수, 미터 단위 거리, 최소 및 최대 고도, 총 상승과 하강을 담은 **TsgcMapGPXTrackStats** 레코드를 반환합니다. **ColorByElevation**를 True로 설정하면 모든 트랙이 고도에 따라 색칠된 조각으로 나뉘며, **ElevationGradient**(TsgcMapElevationGradient)의 LowColor에서 MidColor를 거쳐 HighColor까지 사용합니다.

지원 현황

프레임워크	최소 버전
GPX와 KML, VCL과 FireMonkey	Delphi 7(FireMonkey는 XE6부터)
GeoJSON, VCL과 FireMonkey	Delphi XE6 (System.JSON)
Lazarus (LCL)	사용할 수 없음

GPX와 KML은 TXMLDocument로 파싱됩니다. Windows에서는 MSXML, 그 밖의 플랫폼에서는 Delphi XE7부터 OmniXML을 사용합니다.

참고 항목

- 도형과 레이블

- 마커와 클러스터링
- TsgcVCLMap

지오메트리 함수

유닛 **sgcMaps_Geometry**에는 지도 컨트롤 없이 좌표에 대해 동작하는 함수가 들어 있으므로, 폼뿐 아니라 서비스나 콘솔 애플리케이션에서도 사용할 수 있습니다. 좌표는 WGS84 도 단위이고, 거리는 평균 지구 반지름(SGC_MAP_EARTH_RADIUS, 6371008.8 m)의 구 위에서 잰 미터 단위이며, 방위각은 진북에서 시계 방향으로 0부터 360까지의 각도입니다.

함수	반환값
sgcMapDistance	두 점 사이의 대원 거리이며 단위는 미터입니다.
sgcMapInitialBearing	한 점에서 다른 점으로 가는 초기 방위각입니다.
sgcMapDestinationPoint	시작점에서 방위각과 거리만큼 이동해 도달한 점입니다.
sgcMapGeodesicInterpolate	두 점 사이 대원의 점들이며, 지정한 수의 구간으로 나뉩니다.
sgcMapPointInPolygon	점이 링 안에 있으면 True입니다.
sgcMapPointInPolygonWithHoles	점이 바깥 링 안에 있고 그 구멍들 밖에 있으면 True입니다.
sgcMapSimplify	허용 오차(미터) 안에서 점 수를 줄인 선입니다(Douglas-Peucker).
sgcMapNearestPointOnPolyline	점에서 선까지의 거리이며, 가장 가까운 점과 그 구간의 인덱스를 함께 반환합니다.

점의 배열에는 **TsgcMapCoord** 레코드의 배열인 **TsgcMapCoordArray**를 사용하며, **sgcMapCoord**는 좌표 하나를 만듭니다.

경계와 타일 계산

지도 컨트롤의 **FitBounds**는 상자(북, 서, 남, 동)를 받아 픽셀 단위 여백을 유지하면서 맞는 가장 큰 줌으로 표시하며, 서쪽 경도가 동쪽 경도보다 크면 날짜 변경선을 가로지릅니다. FitToMarkers와 FitToPolyline은 마커와 폴리라인에 대해 같은 일을 합니다. 유닛 sgcMaps_Classes는 웹 메르카토르 투영의 타일 계산을 추가합니다. **sgcMapLatLonToTileF**와 **sgcMapTileFToLatLon**은 좌표와 소수 타일 번호 사이를, **sgcMapLatLonToPixel**과 **sgcMapPixelToLatLon**은 좌표와 세계 픽셀 사이를 변환하고, **sgcMapTileCount**는 한 줌에서 한 변의 타일 수를 알려 줍니다.

지원 현황

프레임워크	최소 버전
VCL, FireMonkey, 콘솔, 서비스	Delphi 7
Lazarus	예

참고 항목

- 도형과 레이블
- 마커와 클러스터링
- TsgcVCLMap

정적 지도 렌더링

지도 컨트롤은 창 없이 렌더링할 수 있습니다. 부모 없이 생성한 TsgcVCLMap은 요청 시 비트맵에 그리며, 이를 통해 콘솔 애플리케이션, 웹 서버, Windows 서비스를 정적 지도 렌더러로 만들 수 있습니다. 보고서용 썸네일, 이메일에 첨부하는 지도, HTTP로 제공하는 이미지가 그 예입니다.

렌더링 메서드

메서드	설명
RenderToBitmap(aBitmap, aIncludeControls)	지도를 자신의 크기인 비트맵에 그립니다. 빈 비트맵은 컨트롤의 크기를 가집니다.
SaveToPNG(aFileName, aWidth, aHeight, aIncludeControls)	지정한 크기의 PNG 파일을 기록합니다.
SaveToBMP(aFileName, aWidth, aHeight, aIncludeControls)	지정한 크기의 BMP 파일을 기록합니다.

이미지에는 타일, 오버레이, 축척 막대, 저작자 표시가 나타납니다. 확대/축소 버튼, 레이어 전환기, 나침반은 aIncludeControls가 True가 아니면 제외됩니다. 타일은 렌더링하는 동안 읽으며 서버의 타일은 그때 다운로드되므로, 호출은 타일이 도착할 때까지 차단됩니다. 이후 **LastRenderMissingTiles**는 불러오지 못한 타일 수를 알려 주고, **AttributionText**는 이미지 옆에 표시할 저작권 텍스트를 반환합니다.
폼에서는 같은 메서드가 사용자가 보고 있는 보기를 내보냅니다.

래스터라이저

벡터 타일은 VCL이나 FireMonkey에 의존하지 않는 안티앨리어싱 소프트웨어 래스터라이저인 **TsgcMapRasterizer**(유닛 sgcMaps_Raster)로 그리며, 이를 사용해 직접 이미지를 그릴 수도 있습니다. 경로를 채우고 획으로 그리며(BeginPath, MoveTo, LineTo, ClosePath, non-zero 또는 even-odd 규칙의 FillPath, 조인, 캡, 대시를 지원하는 StrokePath), 사각형으로 클리핑하고 결과를 SaveToBMPFile과 SaveToPNGFile로 저장합니다. 색상은 알파가 미리 곱해지지 않은 $AARRGGBB 값입니다.

지원 현황

프레임워크	최소 버전
VCL, BMP 출력	Delphi 7
VCL, PNG 출력	Delphi 2009
FireMonkey	Delphi XE6
Lazarus (LCL)	예, PNG와 BMP

서버의 타일에는 Delphi 2009 이상이 필요하며, 컨트롤이 폴더 타일만 그리는 Lazarus에서는 사용할 수 없습니다(타일, 레이어, 공급자 참조). 래스터라이저는 Delphi XE2부터 System.ZLib으로 PNG 파일을 압축하며, 더 오래된 버전에서는 압축하지 않고 기록합니다.

참고 항목

- TsgcVCLMap

- TsgcFMXMap
- 타일, 레이어, 공급자

오프라인 타일과 타일 다운로더

차량, 공장, 선박처럼 네트워크 연결 없이 동작해야 하는 지도는 로컬 파일에서 타일을 그립니다. sgcMaps는 두 가지 오프라인 타일 팩을 읽습니다.

- **폴더**(TileSourceType mtsFolder): TileSourcePath에는 줌마다 하위 폴더 하나, 열마다 하위 폴더 하나, 타일마다 파일 하나가 있으며, 파일 이름은 z\x\y.png, .jpg 또는 .jpeg입니다. 이는 TsgcMapTileDownloader, 레이어의 디스크 캐시, 여러 타일 도구가 기록하는 구조입니다.
- **MBTiles** 파일(TileSourceType mtsMBTiles): 모든 타일을 담은 단일 SQLite 파일입니다. 복사하기 쉬우며 MapTiler, TileMill, gdal 같은 도구로 만듭니다. 벡터 MBTiles 파일은 벡터 타일에서 설명합니다.

타일 팩 다운로드

TsgcMapTileDownloader(유닛 sgcMaps_TileDownloader)는 한 영역과 줌 레벨 범위의 모든 타일로 폴더를 채웁니다.

멤버	설명
URLTemplate	{z} {x} {y} 자리 표시자가 있는 타일 서버입니다. 의도적으로 기본값이 없습니다.
OutputPath	z/x/y 트리의 루트 폴더입니다.
UserAgent	모든 요청과 함께 전송되는 User-Agent 헤더입니다.
DelayMS	두 다운로드 사이의 대기 시간이며 기본값은 250 ms입니다.
Timeout	각 요청의 타임아웃이며 기본값은 15000 ms입니다.
CountTiles	한 영역과 줌 범위의 타일 수이며, 다운로드하기 전에 크기를 확인하는 데 사용합니다.
GetTileRange	한 줌에서 한 영역의 첫 번째 및 마지막 타일 열과 행입니다.
Download	타일을 다운로드합니다. 이미 디스크에 있는 타일은 건너뛰므로, 중단된 다운로드를 이어서 할 수 있습니다.
OnProgress	전체, 다운로드됨, 건너뜀, 실패한 타일 수와 현재 줌을 보고합니다. 중지하려면 aCancel을 True로 설정하십시오.

Download는 동기 방식입니다. 호출한 스레드를 차단하며 같은 스레드에서 OnProgress를 발생시킵니다. 사용자 인터페이스가 응답하도록 하려면 작업자 스레드에서 호출하십시오. URLTemplate이나 OutputPath가 비어 있으면 예외를 발생시킵니다. 타일 사용 정책에 유의하십시오. tile.openstreetmap.org를 포함한 대부분의 공개 타일 서버는 대량 다운로드와 오프라인 선행 가져오기를 금지합니다. 다운로더는 자체 타일 서버나 약관이 허용하는 공급자를 가리키게 하고, DelayMS는 0보다 크게 유지하십시오. 타일 수는 줌 레벨마다 네 배로 늘어나므로, 깊은 레벨을 다운로드하기 전에 CountTiles를 확인하십시오.

지원 현황

프레임워크	최소 버전
폴더 타일	Delphi 7과 Lazarus
MBTiles	Delphi XE7 (FireDAC SQLite)
TsgcMapTileDownloader	Delphi 2009(Indy 10, THTTPClient는 Delphi XE8부터)
Lazarus(LCL), MBTiles와 다운로더	사용할 수 없음

참고 항목

- 타일, 레이어, 공급자
- 벡터 타일
- 설치

OpenStreetMap 리더

OpenStreetMap 데이터는 **.osm.pbf** 파일(Protocol Buffers, 작고 빠름)과 **.osm** XML 파일로 배포됩니다. 국가나 지역별 추출 파일은 Geofabrik과 다른 미러에서 제공합니다. **TsgcMapOSMReader**(유닛 sgcMaps_OSM)는 두 형식을 스트림으로 읽습니다. 문서 전체를 메모리에 만들지 않으므로 어떤 크기의 파일도 읽을 수 있으며, 파일을 읽는 동안 모든 요소를 이벤트로 전달합니다.

멤버	설명
ReadFile, ReadStream	파일 또는 스트림을 읽습니다. 형식은 내용에서 감지되며, DetectFormat은 읽지 않고 형식을 반환합니다.
ReadNodes, ReadWays, ReadRelations	전달할 요소이며, 모두 기본값이 True입니다. 필요 없는 것을 끄면 읽기가 더 빨라집니다.
SkipNodeTags	노드의 태그를 건너뛰며, 좌표만 필요한 리더에 사용합니다.
OnNode	id, 위도, 경도, 태그가 있는 노드입니다.
OnWay	id, 노드의 id들, 태그가 있는 웨이입니다.
OnRelation	id, 멤버(TsgcMapOSMMember: 형식, 참조, 역할), 태그가 있는 릴레이션입니다.
OnFilterWay	웨이가 전달되기 전에 호출됩니다. 건너뛰려면 aAccept를 False로 설정하십시오.
OnProgress	읽은 바이트 수와 전체 크기입니다.

태그는 Count, Keys, Values, Value, Has, IndexOf를 가진 **TsgcMapOSMTags** 객체로 전달됩니다. 태그 객체와 이벤트에 전달되는 배열은 다음 요소에 재사용되므로, 필요한 것은 핸들러 안에서 복사하십시오.
zlib으로 압축되었거나 원시로 저장된 PBF 블록이 지원됩니다. lzma, lz4, zstd, bzip2로 압축된 블록과 OsmSchema-V0.6 및 DenseNodes 이외의 기능이 필요한 파일은 **EsgcMapOSMError**를 발생시킵니다. 이 리더는 TsgcMapRouteGraphBuilder의 입력이며, 경로 그래프와 오프라인 라우팅을 참조하십시오.

지원 현황

프레임워크	최소 버전
VCL, FireMonkey, 콘솔, 서비스	Delphi XE2 (System.ZLib)
Lazarus	사용할 수 없음

참고 항목

- 경로 그래프와 오프라인 라우팅
- TsgcMapOfflineRouter

오프라인 지오코딩 인덱스

네트워크 연결 없는 지오코딩에는 지명의 로컬 데이터베이스가 필요합니다. sgcMaps는 무료 **GeoNames** 덤프로 데이터베이스를 만들고 SQLite로 질의하며, 두 단계로 이루어집니다.

1. TsgcMapGeoIndexBuilder는 GeoNames 덤프(cities500.zip, cities1000.zip, ES.zip 같은 국가 파일, allCountries.zip 또는 압축을 푼 .txt 파일)를 읽고, 지역과 국가 이름을 위해 선택적으로 admin1CodesASCII.txt와 countryInfo.txt도 읽어 인덱스가 있는 SQLite 데이터베이스를 기록합니다. FeatureClasses는 보관할 장소의 종류를 선택하며, P는 거주 지역, A는 행정 구역입니다.
2. TsgcMapOfflineGeocoder는 그 데이터베이스를 열고 Geocode(지명이며, 뒤에 국가를 붙일 수 있음), ReverseGeocode, ReverseGeocodeCountry에 응답합니다.

인덱스는 개발 머신이나 설치 시점에 한 번 만들고 애플리케이션과 함께 배포합니다. 결과는 온라인 TsgcMapGeocoder와 같은 **TsgcMapGeocodingResults** 클래스를 사용하므로, 결과를 표시하는 코드를 둘 다에 사용할 수 있습니다.

클릭 시 역지오코딩

ReverseGeocode는 좌표에서 50 km 이내의 장소를 반환하며, 거주 지역이 먼저 오고 각 그룹은 거리순으로 정렬됩니다. ReverseGeocodeCountry는 가장 가까운 장소가 속한 국가의 ISO 코드를 반환합니다.
각 구성 요소는 SQLite 연결을 하나 유지하므로, 구성 요소는 한 번에 하나의 스레드에서 사용하십시오. 병렬 쿼리에는 스레드마다 하나씩 만드십시오.

지원 현황

프레임워크	최소 버전
VCL, FireMonkey, 콘솔, 서비스	Delphi XE7 (FireDAC SQLite)
Lazarus	사용할 수 없음

참고 항목

- TsgcMapGeoIndexBuilder
- TsgcMapOfflineGeocoder
- TsgcMapGeocoder

경로 그래프와 오프라인 라우팅

네트워크 연결 없는 라우팅에는 라우터가 빠르게 검색할 수 있는 형태의 도로망이 디스크에 있어야 합니다. sgcMaps는 두 단계로 이를 처리합니다.

1. **TsgcMapRouteGraphBuilder**(유닛 sgcMaps_OfflineRouter)는 OpenStreetMap 데이터인 .osm.pbf 또는 .osm 파일을 읽고, 자동차, 자전거, 도보 프로필이 담긴 .sgcgraph 파일인 컴팩트한 바이너리 **경로 그래프**를 기록합니다.
2. TsgcMapOfflineRouter는 그래프를 불러오고, 모든 경유지를 선택한 프로필의 가장 가까운 도로에 맞춘 다음 양방향 A* 탐색으로 경로를 찾습니다.

그래프 빌드

Build(입력 파일, 출력 파일)는 세 단계로 실행되며 **Phase**가 보고합니다. 1은 웨이를 읽고, 2는 그 웨이가 사용하는 노드의 좌표를 읽고, 3은 그래프를 기록합니다. **OnProgress**는 입력을 읽는 동안 읽은 바이트 수와 입력의 전체 크기를 보고합니다. 완료되면 WayCount, NodeCount, MissingNodes, VertexCount, EdgeCount, ShapeCount, NameCount가 그래프를 설명합니다. 그래프는 모든 도로에 대해 길이, 자동차 속도, 프로필별로 통행할 수 있는 방향, 중간 점, 이름을 저장합니다.
애플리케이션이 다루는 지역에 대해 그래프를 한 번 빌드하고 .sgcgraph 파일을 배포하십시오. RouteGraphBuilder 콘솔 도구는 명령줄에서 같은 일을 합니다.

라우팅

GraphFile과 Profile(morCar, morBike, morFoot)을 설정하고 Open을 호출합니다. **GetRoute**는 둘 이상의 경유지를 받아, 온라인 TsgcMapDirections가 사용하는 클래스인 **TsgcMapRoute**를 지오메트리, 거리, 소요 시간, 경유지 쌍마다 하나씩의 구간, 회전 안내로 채웁니다. **sgcMapAddRouteToPolylines**는 이를 지도에 그립니다.
오프라인 라우터는 OnCalculateRoute 이벤트를 통해 TsgcMapRouteCalculator의 엔진이 될 수도 있으며, 이렇게 하면 네트워크 연결 없이도 사용자가 경유지를 드래그할 수 있습니다.

지원 현황

프레임워크	최소 버전
VCL, FireMonkey, 콘솔, 서비스	Delphi XE2
Lazarus	사용할 수 없음

참고 항목

- OpenStreetMap 리더
- TsgcMapOfflineRouter
- TsgcMapRouteCalculator

TsgcMapService

유닛: sgcMaps_Services
모든 온라인 지도 서비스의 기반 클래스로, TsgcMapGeocoder, TsgcMapDirections, TsgcMapPlaces, TsgcMapElevation, TsgcMapTimeZone가 여기서 파생됩니다. 팔레트에는 등록되어 있지 않습니다. 서비스는 THTTPClient로 HTTPS 요청을 보내고 System.JSON으로 응답을 파싱하므로 Delphi XE8 이상이 필요하며 Lazarus에서는 사용할 수 없습니다.

동기 호출과 비동기 호출

- **동기** 메서드(Geocode, GetRoute, Suggest...)는 응답이 도착할 때까지 호출한 스레드를 차단합니다. 이 메서드는 예외를 발생시키지 않으며, 실패하면 False를 반환하고 **LastError**를 설정합니다.
- **비동기** 메서드(GeocodeAsync, GetRouteAsync, SuggestAsync...)는 요청을 백그라운드 스레드에서 실행하고, TThread.Queue를 통해 메인 스레드에서 결과 이벤트 또는 **OnError**를 발생시킵니다. VCL과 FireMonkey 애플리케이션에는 다른 것이 필요 없습니다. 콘솔 애플리케이션이나 서비스는 CheckSynchronize를 주기적으로 호출하거나 동기 메서드를 사용해야 합니다.
- 비동기 요청이 이벤트를 발생시킬 때까지 구성 요소가 살아 있어야 합니다. 이벤트에 전달되는 결과 객체는 구성 요소가 소유하며 이벤트가 반환될 때 해제되므로, 보관해야 할 것은 복사하십시오.

공개 서버와 사용 정책

기본 공급자는 사용 정책이 있는 무료 공개 서비스입니다. Nominatim, OSRM 데모 서버, OpenTopoData 공개 API는 초당 최대 한 건의 요청만 허용하며, 구성 요소가 그 간격을 강제합니다. 또한 식별 가능한 **UserAgent**를 요구합니다. 운영 환경과 대량 요청에는 자체 서버나 상용 공급자를 사용하십시오.

속성

속성	형식	기본값	설명
ApiKey	string		키가 필요한 공급자의 키입니다. GeoNames 시간대 공급자에서는 GeoNames 사용자 이름입니다.
Language	string	en	결과의 언어이며, 이를 지원하는 공급자에 적용됩니다.
Timeout	Integer	15000	밀리초 단위의 요청 타임아웃입니다.
UserAgent	string	sgcMaps/2.0	모든 요청의 User-Agent 헤더입니다. 여기에 애플리케이션을 식별할 수 있는 이름을 지정하십시오.
Referer	string		비어 있지 않으면 전송되는 Referer 헤더입니다.
LastError	string		마지막으로 실패한 요청의 오류입니다. 읽기 전용입니다.

이벤트

이벤트	설명
OnError	비동기 요청이 실패했습니다. 오류 텍스트와 함께 메인 스레드에서 발생합니다.

예제: 서비스의 공통 설정

프로시저 하나로 폼의 모든 서비스를 구성합니다. 속성이 기반 클래스에 선언되어 있기 때문입니다.

참고 항목

- TsgcMapGeocoder
- TsgcMapDirections
- TsgcMapPlaces
- TsgcMapElevation
- TsgcMapTimeZone

TsgcMapGeocoder

유닛: sgcMaps_Geocoding
주소나 지명을 좌표로(지오코딩), 좌표를 주소로(역지오코딩) 변환합니다.
이 구성 요소는 TsgcMapService 에서 파생되며 **ApiKey**, **Language**, **Timeout**, **UserAgent**, **Referer**, **LastError**, **OnError**를 상속합니다. Delphi XE8 이상이 필요합니다. 동기 메서드는 예외를 발생시키지 않고 False를 반환하며 LastError를 설정합니다. 비동기 메서드는 이벤트를 메인 스레드에서 발생시킵니다.

공급자

공급자	서비스	키
mgpNominatim	Nominatim(OpenStreetMap)입니다. 기본값입니다. NominatimURL은 자체 호스팅 인스턴스를 가리킵니다.	아니요
mgpGoogle	Google Geocoding API	예
mgpHere	HERE Geocoding and Search	예
mgpAzure	Azure Maps Search	예
mgpMapBox	Mapbox Geocoding	예
mgpOpenRouteService	OpenRouteService Geocode	예
mgpGeoApify	Geoapify Geocoding	예

속성

속성	형식	기본값	설명
공급자	TsgcMapGeocodingProvider	mgpNominatim	지오코딩 서비스입니다.
NominatimURL	string	https://nominatim.openstreetmap.org	Nominatim 서버의 기본 URL입니다.
MaxResults	Integer	5	최대 결과 수입니다.
CountryCodes	string		ISO 3166-1 alpha-2 코드('es,fr')를 쉼표로 구분한 국가 필터입니다. HERE는 alpha-3 코드('ESP,FRA')를 요구하며 Azure는 이를 무시합니다.

메서드

메서드	설명
Geocode(const aQuery: string; aResults: TsgcMapGeocodingResults): Boolean	주소나 장소를 검색합니다. 결과가 하나 이상 발견되면 True입니다.

ReverseGeocode(aLat, aLon: Double; aResults: TsgcMapGeocodingResults): Boolean	좌표의 주소를 찾습니다.
GeocodeAsync(const aQuery: string)	비동기 Geocode입니다. 결과는 OnGeocodeResult로 전달됩니다.
ReverseGeocodeAsync(aLat, aLon: Double)	비동기 ReverseGeocode입니다. 결과는 OnGeocodeResult로 전달됩니다.
BuildURL, BuildReverseURL	현재 공급자의 요청 URL이며, 로깅과 테스트에 유용합니다.

이벤트

이벤트	설명
OnGeocodeResult	비동기 요청의 결과이며 메인 스레드에서 전달됩니다. 목록은 이벤트가 반환될 때 해제됩니다.

결과

각 결과는 **Latitude**, **Longitude**, **DisplayName**, **Street**, **HouseNumber**, **PostalCode**, **City**, **State**, **Country**, **CountryCode**(공급자가 반환하면 ISO alpha-2, HERE는 alpha-3), **Confidence**(0에서 1, 알 수 없으면 0), 그리고 경계 상자 **BoundsNorth**, **BoundsSouth**, **BoundsEast**, **BoundsWest**(HasBounds 참조)를 가진 TsgcMapGeocodingResult입니다. 같은 결과 클래스를 TsgcMapOfflineGeocoder도 사용합니다.

예제: 지오코딩과 역지오코딩

참고 항목

- TsgcMapService
- TsgcMapPlaces
- TsgcMapOfflineGeocoder

TsgcMapDirections

유닛: sgcMaps_Directions
온라인 라우팅 서비스로 둘 이상의 경유지를 지나는 경로를 계산합니다. 거리는 미터, 소요 시간은 초 단위입니다.
이 구성 요소는 TsgcMapService에서 파생되며 **ApiKey**, **Language**, **Timeout**, **UserAgent**, **Referer**, **LastError**, **OnError**를 상속합니다. Delphi XE8 이상이 필요합니다. 동기 메서드는 예외를 발생시키지 않고 False를 반환하며 LastError를 설정합니다. 비동기 메서드는 이벤트를 메인 스레드에서 발생시킵니다.

공급자

공급자	서비스	키
mdpOSRM	OSRM입니다. 기본 OSRMURL은 OSRM 프로젝트의 공개 데모 서버이며, 가벼운 사용만 허용하고 자동차 프로필만 제공합니다. 운영 환경과 도보 및 자전거에는 자체 osrm-routed를 사용하십시오.	아니요
mdpOpenRouteService	OpenRouteService Directions	예
mdpGoogleRoutes	Google Routes API	예
mdpHere	HERE Routing	예
mdpAzure	Azure Maps Route	예
mdpMapBox	Mapbox Directions	예
mdpGeoApify	Geoapify Routing	예

모든 공급자는 회전 안내를 **Language**로 반환하지만 OSRM은 반환하지 않습니다. OSRM에는 구성 요소가 짧은 영어 안내를 만듭니다.

속성

속성	형식	기본값	설명
공급자	TsgcMapDirectionsProvider	mdpOSRM	라우팅 서비스입니다.
OSRMURL	string	https://router.project-osrm.org	OSRM 서버의 기본 URL입니다.
TravelMode	TsgcMapTravelMode	mtmDriving	mtmDriving, mtmWalking, mtmCycling 또는 mtmTruck입니다. IsTravelModeSupported는 공급자가 허용하는 모드를 알려 줍니다.
Alternatives	Boolean	False	대체 경로를 요청합니다.
AvoidTolls, AvoidHighways, AvoidFerries	Boolean	False	경로 제한이며, 이를 지원하는 공급자에 적용됩니다.
OptimizeWaypoints	Boolean	False	공급자가 중간 경유지의 순서를 바꿀 수 있게 합니다.

메서드

메서드	설명
GetRoute(const aWaypoints: array of TsgcMapCoord; aRoutes: TsgcMapRoutes): Boolean	경로를 계산합니다. 경로가 하나 이상 발견되면 True이며, 인덱스 0이 주 경로입니다.
GetRouteAsync(const aWaypoints: array of TsgcMapCoord)	비동기 GetRoute입니다. 경로는 OnRouteResult로 전달됩니다.
ValidateRequest(const aWaypoints: array of TsgcMapCoord): string	요청을 보낼 수 있으면 빈 문자열이고, 그렇지 않으면 보낼 수 없는 이유입니다.
IsTravelModeSupported(aProvider; aMode): Boolean	클래스 함수입니다. 공급자가 이동 수단을 지원하는지 여부를 반환합니다.

이벤트

이벤트	설명
OnRouteResult	비동기 요청의 경로이며 메인 스레드에서 전달됩니다. 목록은 이벤트가 반환될 때 해제됩니다.

경로 결과

클래스	멤버
TsgcMapRoutes	TsgcMapRoute의 Count와 Items[Index](기본 속성)입니다.
TsgcMapRoute	Distance, Duration, Summary, 지오메트리(Points, PointCount, GetPoints), 구간(Legs, LegCount), 경계 상자(BoundsNorth, BoundsSouth, BoundsEast, BoundsWest, CalcBounds)입니다.
TsgcMapRouteLeg	연속한 두 경유지 사이의 경로 부분입니다. Distance, Duration, Summary, Steps, StepCount입니다.
TsgcMapRouteStep	하나의 기동: Instruction, Maneuver, StreetName, Distance, Duration, Latitude, Longitude, PointIndex(경로 지오메트리에서의 위치)입니다.

sgcMapAddRouteToPolylines(aRoute, aPolylines, aColor, aWidth)는 경로의 지오메트리로 이루어진 폴리라인을 지도에 추가하고 반환합니다.

예제: 경로 계산

참고 항목

- TsgcMapService
- TsgcMapRouteCalculator
- TsgcMapOfflineRouter

TsgcMapPlaces

유닛: sgcMaps_Places
장소 자동 완성입니다. 사용자가 입력하는 텍스트에 대한 제안이며, 보통 검색 상자 아래의 목록에 표시합니다.
이 구성 요소는 TsgcMapService 에서 파생되며 **ApiKey**, **Language**, **Timeout**, **UserAgent**, **Referer**, **LastError**, **OnError**를 상속합니다. Delphi XE8 이상이 필요합니다. 동기 메서드는 예외를 발생시키지 않고 False를 반환하며 LastError를 설정합니다. 비동기 메서드는 이벤트를 메인 스레드에서 발생시킵니다.

공급자

공급자	서비스	키
mplPhoton	Photon(komoot)입니다. 기본값입니다. 언어는 de, en, fr만 허용하며, 그 밖의 Language는 전송되지 않습니다.	아니요
mplNominatim	Nominatim입니다. 초당 한 건의 요청으로 제한되므로, 키 입력마다 자동 완성하는 데 공개 서버를 사용하지 마십시오.	아니요
mplGoogle	Google Places API (New)입니다. 제안에는 좌표가 없으므로 Retrieve를 호출하십시오.	예
mplHere	HERE Autosuggest	예
mplMapBox	Mapbox Search Box API입니다. 제안에는 좌표가 없으므로 Retrieve를 호출하십시오.	예
mplGeoApify	Geoapify Autocomplete	예

속성

속성	형식	기본값	설명
공급자	TsgcMapPlacesProvider	mplPhoton	자동 완성 서비스입니다.
NominatimURL	string	https://nominatim.openstreetmap.org	Nominatim 서버의 기본 URL입니다.
MaxResults	Integer	5	최대 제안 수입니다.
BiasLatitude, BiasLongitude	Double	0	이 위치 근처의 결과가 우선하며 Distance가 채워집니다. 둘 다 0이면 사용되지 않습니다.
BiasRadius	Double	0	편향 위치 주변의 반경이며 단위는 미터입니다. 0이면 반경이 없습니다. Google, HERE, Nominatim에서 사용됩니다.
SessionToken	string		현재 검색 세션의 토큰입니다. 공개 속성입니다.

메서드

메서드	설명
Suggest(const aText: string; aResults: TsgcMapPlaceResults): Boolean	텍스트에 대한 제안을 반환합니다. 하나 이상 발견되면 True입니다.
SuggestAsync(const aText: string)	비동기 Suggest입니다. 제안은 OnSuggestResult로 전달됩니다.
Retrieve(aPlace: TsgcMapPlace): Boolean	제안의 좌표와, 누락된 이름이나 주소를 채웁니다. Google과 Mapbox만 요청을 보내며, 다른 공급자에서는 HasCoordinates를 반환합니다.
NewSession	새 SessionToken으로 새 검색 세션을 시작합니다.

이벤트

이벤트	설명
OnSuggestResult	비동기 요청의 제안이며 메인 스레드에서 전달됩니다. 목록은 이벤트가 반환될 때 해제됩니다.

결과

각 제안은 **Id**(Retrieve가 사용하는 공급자의 식별자), **Name**, **Address**, **Latitude**, **Longitude**, **HasCoordinates**, **Category**, **Distance**(편향 위치로부터의 미터이며 알 수 없으면 -1)를 가진 TsgcMapPlace입니다.

예제: 입력하는 동안의 제안

참고 항목

- TsgcMapService
- TsgcMapGeocoder

TsgcMapElevation

유닛: sgcMaps_Elevation
점 목록의 지형 고도를 반환하거나, 경로를 따라 같은 간격으로 뽑은 샘플의 지형 고도를 반환합니다. 예를 들면 경로의 고도 프로파일입니다. 긴 목록은 자동으로 배치로 나뉩니다.
이 구성 요소는 TsgcMapService에서 파생되며 **ApiKey**, **Language**, **Timeout**, **UserAgent**, **Referer**, **LastError**, **OnError**를 상속합니다. Delphi XE8 이상이 필요합니다. 동기 메서드는 예외를 발생시키지 않고 False를 반환하며 LastError를 설정합니다. 비동기 메서드는 이벤트를 메인 스레드에서 발생시킵니다.

공급자

공급자	서비스	요청당 점 수	키
meOpenElevation	Open-Elevation	200	아니요
meOpenTopoData	OpenTopoData입니다. 기본값입니다. 공개 API는 초당 한 건의 요청으로 제한됩니다.	100	아니요
meGoogle	Google Elevation API	256	예

속성

속성	형식	기본값	설명
공급자	TsgcMapElevationProvider	meOpenTopoData	고도 서비스입니다.
Dataset	string	srtm90m	OpenTopoData 데이터셋: srtm90m, srtm30m, aster30m, eudem25m 등입니다.

메서드

메서드	설명
GetElevations(const aPoints: array of TsgcMapCoord; aResults: TsgcMapElevationResults): Boolean	점마다 결과 하나이며, 순서는 같습니다.
GetElevationAlongPath(const aPoints: array of TsgcMapCoord; aSamples: Integer; aResults: TsgcMapElevationResults): Boolean	폴리라인 aPoints를 따라 거리 기준으로 같은 간격으로 배치한 aSamples개의 점(최소 2개, 처음과 마지막 포함)입니다.
GetElevationsAsync, GetElevationAlongPathAsync	비동기 버전입니다. 결과는 OnElevationResult로 전달됩니다.
BatchSize: Integer	현재 공급자의 요청당 최대 점 수입니다.
SamplePath	클래스 함수입니다. 폴리라인에서 같은 간격으로 뽑은 샘플과 첫 번째 점으로부터의 거리입니다.

이벤트

이벤트	설명
OnElevationResult	비동기 요청의 결과이며 메인 스레드에서 전달됩니다. 목록은 이벤트가 반환될 때 해제됩니다.

결과

각 결과는 **Latitude**, **Longitude**, **Elevation**(해발 미터), **HasElevation**(공급자에 그 점의 데이터가 없으면 False), **Distance**(첫 번째 점으로부터의 미터이며 GetElevationAlongPath가 채움)를 가진 TsgcMapElevationResult입니다.

예제: 고도 프로파일

참고 항목

- TsgcMapService
- TsgcMapDirections

TsgcMapTimeZone

유닛: sgcMaps_TimeZone
좌표의 시간대를 반환합니다. IANA 식별자, UTC로부터의 오프셋, 현지 시간입니다.
이 구성 요소는 TsgcMapService에서 파생되며 **ApiKey**, **Language**, **Timeout**, **UserAgent**, **Referer**, **LastError**, **OnError**를 상속합니다. Delphi XE8 이상이 필요합니다. 동기 메서드는 예외를 발생시키지 않고 False를 반환하며 LastError를 설정합니다. 비동기 메서드는 이벤트를 메인 스레드에서 발생시킵니다.

공급자

공급자	서비스	키
mtzTimeAPI	TimeAPI(timeapi.io)입니다. 기본값입니다.	아니요
mtzGeoNames	GeoNames입니다. ApiKey는 GeoNames 사용자 이름입니다.	사용자 이름
mtzGoogle	Google Time Zone API	예
mtzAzure	Azure Maps Timezone	예

선택적인 **aUtcTime** 매개변수(0은 현재를 의미)는 Google과 Azure에서 적용됩니다. TimeAPI와 GeoNames는 항상 현재 시간에 대해 응답합니다.

속성

속성	형식	기본값	설명
공급자	TsgcMapTimeZoneProvider	mtzTimeAPI	시간대 서비스입니다.

메서드

메서드	설명
GetTimeZone(aLat, aLon: Double; aInfo: TsgcMapTimeZoneInfo; aUtcTime: TDateTime = 0): Boolean	aInfo를 좌표의 시간대로 채웁니다.
GetTimeZoneAsync(aLat, aLon: Double; aUtcTime: TDateTime = 0)	비동기 GetTimeZone입니다. 응답은 OnTimeZoneResult로 전달됩니다.
UtcNow: TDateTime	클래스 함수입니다. 현재 UTC 시간입니다.

이벤트

이벤트	설명
OnTimeZoneResult	비동기 요청의 응답이며 메인 스레드에서 전달됩니다. 객체는 이벤트가 반환될 때 해제됩니다.

결과

TsgcMapTimeZoneInfo 에는 **TimeZoneId**('Europe/Madrid'), **Name**(공급자가 반환하면 표시 이름, 그렇지 않으면 TimeZoneId), **UtcOffsetSeconds**(표준 시간에 일광 절약 시간을 더한 값), **DstOffsetSeconds**(일광 절약 시간 부분), **IsDst**, **LocalTime**(오프셋 없는 ISO 8601)이 있습니다.

예제: 좌표의 시간대

참고 항목

- TsgcMapService

TsgcMapRouteCalculator

유닛: sgcMaps_RouteCalculator
지도에서 하는 대화형 경로 편집입니다. 경유지는 **Markers**의 드래그할 수 있는 마커이고 경로는 **Polylines**의 폴리라인이며, 둘 다 지도 컨트롤의 컬렉션을 가리키게 하십시오. **AutoCalculate**가 True이면 경유지를 드래그하거나 추가하거나 제거할 때 경로가 다시 계산됩니다. Delphi XE2 이상이 필요하며, Directions 속성에는 Delphi XE8이 필요합니다.
경로 엔진은 **OnCalculateRoute** 이벤트입니다. 여기서 어떤 엔진이든 경로를 채울 수 있습니다(예: TsgcMapOfflineRouter). 이벤트가 요청을 처리하지 않으면 **Directions**에 할당된 온라인 TsgcMapDirections를 사용합니다.
경유지 마커는 Tag = MarkerTagBase + 인덱스를, 경로 폴리라인은 Tag = MarkerTagBase를 가집니다. ShowAlternatives를 사용하면 대체 경로가 Tag = MarkerTagBase + 1인 더 얇은 회색 폴리라인으로 그려집니다.

속성

속성	형식	기본값	설명
Markers	TsgcMapMarkers		경유지 마커의 컬렉션이며, 예를 들면 Map.Markers입니다.
폴리라인	TsgcMapPolylines		경로 폴리라인의 컬렉션이며, 예를 들면 Map.Polylines입니다.
Directions	TsgcMapDirections		온라인 경로 엔진이며, OnCalculateRoute가 요청을 처리하지 않을 때 사용됩니다.
Active	Boolean	True	False이면 계산기가 Markers와 Polylines에서 분리됩니다. 마커 드래그를 추적하지 않으며 자동 계산도 없습니다.
AutoCalculate	Boolean	True	경유지가 바뀔 때마다 다시 계산합니다.
ShowAlternatives	Boolean	False	대체 경로도 그립니다.
RouteColor, RouteWidth	TColor, Integer	blue, 4	경로 폴리라인의 색상과 너비입니다.
WaypointColor	TColor	red	경유지 마커의 색상입니다.
MarkerTagBase	Integer	SGC_MAP_ROUTE_TAG_BASE	계산기의 마커와 폴리라인에 사용되는 첫 번째 Tag입니다.
Waypoints[Index], WaypointMarkers[Index]	TsgcMapCoord, TsgcMapMarker		경유지와 그 마커입니다. 읽기 전용입니다.
Route, RoutePolyline, Routes	TsgcMapRoute, TsgcMapPolyline, TsgcMapRoutes		주 경로, 그 폴리라인, 마지막 계산의 모든 경로입니다(인덱스 0이

			주 경로). 읽기 전용입니다.

메서드

메서드	설명
AddWaypoint(aLat, aLon: Double): Integer	경유지를 끝에 추가하고 그 인덱스를 반환합니다.
InsertWaypoint(aIndex: Integer; aLat, aLon: Double)	경유지를 삽입합니다.
AddWaypointNearestSegment(aLat, aLon: Double): Integer	해당 점에서 구간이 가장 가까운 두 경유지 사이에 경유지를 삽입하며, 경로선을 드래그할 때와 같습니다.
MoveWaypoint(aIndex: Integer; aLat, aLon: Double)	경유지를 이동합니다.
RemoveWaypoint(aIndex: Integer), ClearWaypoints	경유지 하나 또는 전부를 제거합니다.
WaypointCount: Integer	경유지의 수입니다.
Calculate: Boolean	동기 계산입니다. OnCalculateRoute를 먼저 실행하고 그다음 Directions.GetRoute를 실행합니다.
CalculateAsync	OnCalculateRoute이며, 처리되면 즉시 적용되고 그렇지 않으면 Directions.GetRouteAsync를 사용합니다. 이벤트는 메인 스레드에서 발생합니다.
SelectRoute(aIndex: Integer)	대체 경로를 주 경로로 만들고, 다시 그리며 OnRouteCalculated를 발생시킵니다.

이벤트

이벤트	설명
OnCalculateRoute	모든 계산에서 가장 먼저 발생합니다. aRoute를 채우고 aHandled를 True로 설정하면 자체 엔진을 사용할 수 있습니다.
OnRouteCalculated	새 경로가 계산되고 그려졌습니다.
OnWaypointsChanged	경유지가 변경되었습니다.
OnError	계산에 실패했습니다.

예제: 지도를 클릭하여 경로 만들기

네트워크 연결 없이 경로를 계산하려면 TsgcMapOfflineRouter에 나온 것처럼 OnCalculateRoute를 오프라인 라우터로 처리하십시오.

참고 항목

- TsgcMapDirections
- TsgcMapOfflineRouter

- TsgcVCLMap

TsgcMapOfflineRouter

유닛: sgcMaps_OfflineRouter
OpenStreetMap 데이터로 만든 경로 그래프 파일(**.sgcgraph**)을 사용하여 네트워크 연결 없이 경로를 계산합니다. Delphi XE2 이상이 필요합니다.
같은 유닛에 선언된 **TsgcMapRouteGraphBuilder**나 Tools\Maps의 RouteGraphBuilder 도구로, OpenStreetMap 추출 파일(.osm.pbf 또는 .osm XML)에서 그래프를 한 번 빌드합니다. 라우터는 TsgcMapDirections와 같은 TsgcMapRoute 클래스를 채우므로, 어느 엔진의 경로든 같은 방식으로 그리고 검사할 수 있습니다. 여러 경유지를 지나는 경로에는 연속한 경유지 쌍마다 구간이 하나씩 있습니다.

속성

속성	형식	기본값	설명
GraphFile	string		.sgcgraph 파일입니다.
Profile	TsgcMapOfflineRouterProfile	morCar	morCar, morBike 또는 morFoot입니다.
Active	Boolean	False	그래프를 열거나 닫습니다.
LastError	string		마지막으로 실패한 호출의 오류입니다. 읽기 전용입니다.
VertexCount, EdgeCount	Integer		불러온 그래프의 크기입니다. 읽기 전용입니다.

메서드

메서드	설명
Open, Close	그래프를 불러오거나 해제합니다.
GetRoute(const aWaypoints: array of TsgcMapCoord; aRoute: TsgcMapRoute): Boolean	경로를 계산합니다. 경로를 찾지 못하면 False이며, 이유는 LastError에 있습니다.

예제: 오프라인 경로와 경로 계산기의 오프라인 엔진

참고 항목

- TsgcMapRouteCalculator
- TsgcMapDirections

TsgcMapOfflineGeocoder

유닛: sgcMaps_OfflineGeocoder
TsgcMapGeoIndexBuilder가 GeoNames 덤프로 만든 SQLite 인덱스를 사용하여 네트워크 연결 없이 지오코딩과 역지오코딩을 합니다. FireDAC SQLite 드라이버를 사용하므로 Delphi XE7 이상이 필요합니다.
순방향 검색은 지명, 접두사, 선택적인 국가 또는 지역을 받습니다. 예를 들면 'Madrid', 'mostol', 'Toledo, US', 'Madrid ES', 'Toledo, Ohio'입니다. 결과는 정확히 일치하는 것이 접두사 일치보다 먼저, 그다음 인구순, 그다음 피처 코드순(수도, 행정 중심지, 그 밖의 거주 지역 순)으로 정렬됩니다. 역방향 검색은 50 km 이내에서 가장 가까운 장소를 반환하며, 거주 지역이 먼저 오고 각 그룹은 거리순으로 정렬됩니다.
각 구성 요소는 SQLite 연결을 하나 소유하며 한 번에 하나의 스레드에서만 사용해야 합니다. 같은 인덱스를 여러 스레드에서 질의하려면 스레드마다 구성 요소를 하나씩 만드십시오. 파일은 읽기 전용으로 열립니다.

속성

속성	형식	기본값	설명
DatabaseFile	string		SQLite 인덱스 파일입니다.
Active	Boolean	False	인덱스를 열거나 닫습니다.
MaxResults	Integer	5	최대 결과 수입니다.
CountryCodes	string		모든 검색을 제한하는, 쉼표로 구분한 ISO alpha-2 코드입니다.
Language	string		언어 힌트입니다. GeoNames 덤프에는 대체 이름마다 언어 정보가 없으므로 이 값은 현재 무시됩니다.

메서드

메서드	설명
Open, Close	인덱스를 열거나 닫습니다.
Geocode(const aQuery: string; aResults: TsgcMapGeocodingResults): Boolean	순방향 검색입니다. 장소가 하나 이상 발견되면 True입니다.
ReverseGeocode(aLatitude, aLongitude: Double; aResults: TsgcMapGeocodingResults): Boolean	50 km 이내에서 가장 가까운 장소입니다.
ReverseGeocodeCountry(aLatitude, aLongitude: Double): string	가장 가까운 장소가 속한 국가의 ISO 코드이며, 50 km 이내에 장소가 없으면 비어 있습니다.

결과는 TsgcMapGeocoder와 같은 TsgcMapGeocodingResult 클래스를 사용합니다.

예제: 오프라인 검색

참고 항목

- TsgcMapGeoIndexBuilder
- TsgcMapGeocoder

TsgcMapGeoIndexBuilder

유닛: sgcMaps_OfflineGeocoder
TsgcMapOfflineGeocoder 가 검색하는 SQLite 인덱스를 GeoNames 덤프로 빌드합니다. 덤프는 https://download.geonames.org/export/dump/ 에서 다운로드하며, cities500.zip, cities1000.zip, allCountries.zip, ES.zip 같은 국가 파일, 또는 압축을 푼 .txt 파일입니다. Delphi XE7 이상(FireDAC SQLite)이 필요합니다.
빌드는 호출한 스레드에서 실행됩니다. Tools\Maps의 GeoIndexBuilder 도구는 이 구성 요소를 콘솔 애플리케이션으로 감쌉니다.

속성

속성	형식	기본값	설명
InputFile	string		GeoNames 덤프입니다. 탭으로 구분된 .txt 파일 또는 그것을 담은 .zip입니다.
DatabaseFile	string		생성할 SQLite 파일입니다. 기존 파일은 덮어씁니다.
Admin1CodesFile	string		선택적인 admin1CodesASCII.txt이며, 1단계 지역의 이름에 사용됩니다.
CountryInfoFile	string		선택적인 countryInfo.txt이며, 국가 이름에 사용됩니다.
FeatureClasses	string	PA	가져올 GeoNames 피처 클래스입니다. P는 거주 지역, A는 행정 구역, S는 스팟, T는 지형, L은 지역, H는 수문, R은 도로, U는 해저, V는 식생입니다.
MinPopulation	Int64	0	이 인구보다 적은 거주 지역은 건너뜁니다. 다른 클래스는 필터링되지 않습니다.
IncludeAlternateNames	Boolean	True	짧은 이름과 다른 언어의 이름 같은 대체 이름을 64자까지 인덱싱합니다.
BatchSize	Integer	2000	배치마다 기록하는 행 수입니다.
RowsRead, PlacesWritten, NamesWritten, ElapsedMs	Int64		마지막 빌드의 통계입니다. 읽기 전용입니다.

메서드

메서드	설명
Build	인덱스를 빌드합니다.
RowsPerSecond: Double	마지막 빌드의 속도입니다.

이벤트

이벤트	설명

OnProgress	빌드 중에 읽은 행 수, 기록된 장소, 경과 시간(밀리초)과 함께 발생합니다.

예제: 인덱스 빌드

참고 항목

- TsgcMapOfflineGeocoder

TsgcMapLocation

유닛: sgcMaps_Location
기기의 위치입니다. System.Sensors의 TLocationSensor를 감싸므로 Delphi XE6 이상이 필요하며 VCL과 FireMonkey 애플리케이션에서 동작합니다. Windows는 Windows Location API를 사용하고, Android, iOS, macOS는 각자의 네이티브 위치 서비스를 사용합니다.
이 구성요소는 지도를 알지 못합니다. **OnLocationChanged**는 모든 측정값을 보고하고, **UpdateMarker**와 **UpdateAccuracyCircle**은 지도의 컬렉션에서 위치 마커와 정확도 원을 최신 상태로 유지합니다. **FollowMode**를 mfmCenter로 설정하면 모든 측정값이 **OnFollow**도 발생시키며, 애플리케이션은 여기서 지도의 중심을 맞춥니다.

권한

- Android: 프로젝트 옵션의 Uses Permissions에 ACCESS_FINE_LOCATION을 선언하고, Active를 True로 설정하기 전에 **RequestPermission**(Delphi 10.3 이상) 또는 PermissionsService.RequestPermissions로 런타임에 요청합니다.
- iOS와 macOS: 프로젝트의 버전 정보에 위치 사용 설명 키를 추가합니다.

속성

속성	형식	기본값	설명
Active	Boolean	False	위치 센서를 시작하거나 중지합니다.
Accuracy	Double		센서의 원하는 정확도이며 단위는 미터입니다.
Distance	Double		보고되는 두 측정값 사이의 최소 이동 거리이며 단위는 미터입니다.
FollowMode	TsgcMapLocationFollowMode	mfmNone	mfmCenter는 모든 측정값 뒤에 OnFollow를 발생시킵니다.
Available	Boolean		기기에 위치 센서가 있는지 여부입니다. 읽기 전용입니다.
HasLocation	Boolean		측정값이 도착했는지 여부입니다. 읽기 전용입니다.
LastLatitude, LastLongitude, LastAccuracy, LastHeading, LastSpeed, LastTimestamp			마지막 측정값입니다. 읽기 전용입니다.
TrackCount, TrackIndex, TrackIntervalMs, TrackAccuracy			LoadTrack으로 불러온 시뮬레이션 트랙의 상태입니다. 읽기 전용입니다.

메서드

메서드	설명

UpdateMarker(aMarkers: TsgcMapMarkers; aTag: Integer)	Tag로 식별되는 위치 마커를 추가하거나 이동합니다.
UpdateAccuracyCircle(aCircles: TsgcMapCircles; aTag: Integer)	정확도 반경의 원을 추가하거나 갱신합니다.
RequestPermission	Android에서 위치 권한을 요청합니다. 응답은 OnPermissionResult로 전달됩니다.
SimulateLocation(aLatitude, aLongitude: Double; aAccuracy: Double = 0; aHeading: Double = -1; aSpeed: Double = -1)	센서가 읽은 것처럼 위치를 보고하며, 테스트용입니다.
LoadTrack(const aPoints: array of TsgcMapCoord; aIntervalMs: Integer = 1000; aAccuracy: Double = 10)	점 목록(예: 경로의 점)을 불러와 이동을 시뮬레이션합니다.
StepTrack: Boolean	불러온 트랙의 다음 점을 보고합니다. 트랙의 끝이면 False입니다.
ResetTrack	트랙의 첫 번째 점으로 돌아갑니다.

이벤트

이벤트	설명
OnLocationChanged	새 측정값입니다: 위도, 경도, 정확도, 방향, 속도.
OnFollow	FollowMode = mfmCenter이면 OnLocationChanged 뒤에 발생하며, 중심을 맞출 위치가 전달됩니다.
OnStateChanged	센서가 시작되거나 중지되었습니다.
OnError	센서가 오류를 보고했습니다.
OnPermissionResult	RequestPermission의 응답입니다.

예제: 위치 표시와 따라가기

참고 항목

- TsgcVCLMap
- TsgcFMXMap

도구

Tools\Maps 폴더에는 오프라인 구성 요소의 데이터를 준비하는 콘솔 애플리케이션 두 개가 있습니다. 이 도구는 sgcMaps의 빌더 클래스를 감싸고 있으므로, 같은 작업을 직접 작성한 코드에서도 할 수 있습니다.

도구	입력	출력	사용하는 곳	최소 버전
GeoIndexBuilder	GeoNames 덤프	SQLite 인덱스	TsgcMapOfflineGeocoder	Delphi XE7 (FireDAC SQLite)
RouteGraphBuilder	OpenStreetMap 추출 파일	.sgcgraph 파일	TsgcMapOfflineRouter	Delphi XE2

도구 빌드

각 도구는 하나의 .dpr 파일입니다. IDE에서 열어 컴파일하거나, 유닛 경로와 인클루드 경로에 Source 폴더를 지정하여 명령 프롬프트에서 컴파일합니다. 예를 들면 다음과 같습니다.

```
dcc32 -NSSystem;Data;FireDAC -U..\..\..\Source -I..\..\..\Source sgcMapsGeoIndexBuilder.dpr
```

데이터는 개발 머신에서 한 번 빌드하고, 결과로 나온 파일을 애플리케이션과 함께 배포하십시오.

GeoIndexBuilder

sgcMapsGeoIndexBuilder(Tools\Maps\GeoIndexBuilder)는 GeoNames 덤프에서 오프라인 지오코더의 SQLite 인덱스를 빌드합니다. TsgcMapGeoIndexBuilder의 콘솔 래퍼이며, 인덱스를 FireDAC SQLite로 기록하므로 Delphi XE7 이상이 필요합니다.

입력 데이터

GeoNames 덤프는 무료로 다운로드할 수 있습니다(https://download.geonames.org/export/dump/).

- **cities500.zip**, **cities1000.zip**, **cities5000.zip**, **cities15000.zip**: 해당 인구 이상인 전 세계의 거주 지역입니다.
- **XX.zip**, 예를 들어 ES.zip: 한 국가의 모든 피처입니다.
- **allCountries.zip**: 전 세계의 모든 피처입니다.
- **admin1CodesASCII.txt**(선택 사항): 주, 지역 같은 1단계 행정 구역의 이름입니다.
- **countryInfo.txt**(선택 사항): 국가 이름이며, "Paris, France" 같은 검색을 한정하는 데 사용됩니다.

이 도구는 .zip 파일을 직접 읽거나, 압축을 푼 .txt 파일을 읽습니다.

명령줄

```
sgcMapsGeoIndexBuilder <input.zip|txt> <output.sqlite> [--minpop N]
  [--admin1 file] [--countryinfo file] [--classes PA] [--noalternates]
```

매개변수	속성	의미
input	InputFile	GeoNames 덤프이며 .zip 또는 .txt입니다.
output	DatabaseFile	생성할 SQLite 파일입니다. 기존 파일은 덮어씁니다.
--minpop N	MinPopulation	이 인구보다 적은 거주 지역(피처 클래스 P)을 건너뜁니다. 다른 피처 클래스는 필터링되지 않습니다.
--admin1 file	Admin1CodesFile	admin1CodesASCII.txt 파일입니다.
--countryinfo file	CountryInfoFile	countryInfo.txt 파일입니다.
--classes	FeatureClasses	가져올 GeoNames 피처 클래스입니다. 기본값은 PA이며 거주 지역과 행정 구역입니다. 나머지는 S(스팟), T(지형), L(지역), H(수문), R(도로), U(해저), V(식생)입니다.
--noalternates	IncludeAlternateNames	대체 이름(약칭과 다른 언어의 이름)을 제외하여 인덱스가 더 작아집니다.

```
sgcMapsGeoIndexBuilder cities500.zip world.sqlite --admin1 admin1CodesASCII.txt --countryinfo countryInfo.txt
```

출력

도구는 진행 상황을 출력한 다음, 읽은 행 수, 기록된 장소와 이름, 시간, 인덱스 크기를 출력합니다:

```
Building world.sqlite from cities500.zip
  ... rows read, ... places, ... s
Done: ... rows read, ... places, ... names, ... s, ... rows/s, ... MB
```

성공하면 코드 0으로, 잘못된 매개변수이면 1로, 빌드가 실패하면 오류를 출력한 후 2로 종료합니다.

인덱스 사용

.sqlite 파일을 애플리케이션과 함께 배포하고, DatabaseFile을 설정하여 TsgcMapOfflineGeocoder로 엽니다. 그러면 구성 요소가 네트워크 연결 없이 Geocode, ReverseGeocode, ReverseGeocodeCountry에 응답합니다. 애플리케이션에는 문제 해결에서 설명한 대로 런타임에 FireDAC SQLite 드라이버가 필요합니다. 오프라인 지오코딩 인덱스에서 인덱스의 내용을 설명합니다.

RouteGraphBuilder

sgcMapsRouteGraphBuilder(Tools\Maps\RouteGraphBuilder)는 OpenStreetMap 추출 파일을 오프라인 라우터의 라우팅 그래프로 변환합니다. TsgcMapRouteGraphBuilder(유닛 sgcMaps_OfflineRouter)의 콘솔 래퍼이며 Delphi XE2 이상이 필요합니다.

입력 데이터

.osm.pbf 형식 또는 **.osm** XML로 된 OpenStreetMap 추출 파일입니다. Geofabrik 같은 다운로드 서비스는 모든 대륙, 국가, 지역의 추출 파일을 게시합니다. 빌드의 시간과 메모리는 추출 파일에 따라 늘어나므로, 애플리케이션이 다루는 영역을 포함하는 가장 작은 추출 파일을 선택하십시오.

명령줄

```
sgcMapsRouteGraphBuilder <input.osm.pbf|input.osm> <output.sgcgraph>

sgcMapsRouteGraphBuilder andorra-latest.osm.pbf andorra.sgcgraph
```

출력

빌더는 파일을 두 번 읽습니다. 먼저 웨이를 읽고, 그다음 그 웨이가 사용하는 노드의 좌표를 읽으며, 10퍼센트마다 진행 상황을 출력합니다. 마지막에는 그래프의 통계를 출력합니다:

```
Input : andorra-latest.osm.pbf
Output: andorra.sgcgraph
  reading ways   0%
  ...
  reading nodes 100%
Routable ways : ...
Nodes used    : ... (missing ...)
Vertices      : ...
Edges         : ...
Shape points  : ...
Street names  : ...
Graph size    : ... bytes
Time          : ... ms
```

Nodes used는 라우팅 가능한 웨이가 참조하는 노드의 수이고, missing은 파일에 없는 노드의 수이며, 이는 추출 파일의 경계를 넘는 웨이에서 발생합니다. 도구는 성공하면 코드 0으로, 잘못된 매개변수이면 2로, 빌드가 실패하면 오류를 출력한 후 1로 종료합니다.

그래프 사용

하나의 .sgcgraph 파일에 자동차, 자전거, 도보 프로필이 들어 있습니다. 이 파일을 애플리케이션과 함께 배포하고, TsgcMapOfflineRouter의 GraphFile로 설정하고, Profile(morCar, morBike, morFoot)을 선택한 다음 GetRoute를 호출합니다. 경로는 TsgcMapDirections가 사용하는 것과 같은 TsgcMapRoute 클래스로 반환되므로, sgcMapAddRouteToPolylines로 그리거나 TsgcMapRouteCalculator에 전달할 수 있습니다. 경로 그래프와 오프라인 라우팅에서 그래프와 빌더 클래스를 설명합니다.

Lazarus

sgcMaps에는 Lazarus와 Free Pascal용 패키지 **Packages\Lazarus\sgcMapsLazarus.lpk**가 있습니다. 이 패키지는 LCL에 지도 컨트롤을 설치하며, VCL 컨트롤과 같은 지도 모델과 같은 코드를 사용하지만, Delphi 전용 RTL 유닛이 필요한 부분은 FPC에서 컴파일되지 않으므로 기능이 더 적습니다.
가장 중요한 차이: **Lazarus 컨트롤은 타일을 다운로드할 수 없으므로 폴더의 타일만 표시합니다.**

패키지 설치

1. Lazarus를 시작하고 **Package**, **Open Package File (.lpk)**를 선택합니다.
2. sgcUI 설치본의 **Packages\Lazarus\sgcMapsLazarus.lpk**를 엽니다. 패키지는 상대 경로를 통해 Source 폴더에서 유닛을 찾으므로, 설치본의 폴더 구조를 그대로 유지하십시오.
3. **Compile**을 누른 다음 **Use**, **Install**을 선택하고, Lazarus가 IDE를 다시 빌드하고 다시 시작하도록 합니다.
4. 이제 TsgcVCLMap이 구성 요소 팔레트의 **SGC Maps** 페이지에 있습니다.

이 패키지에는 모든 Lazarus 설치에 있는 LCL, FCL, IDEIntf 패키지가 필요합니다. 지도를 사용하는 프로젝트는 필수 패키지에 sgcMapsLazarus를 나열하며, 컨트롤을 폼에 놓으면 IDE가 이를 추가합니다.

포함되는 항목

포함 항목	유닛
LCL의 지도 컨트롤, TsgcVCLMap	sgcMaps_VCL
지도 모델: 레이어, 마커, 폴리라인, 폴리곤, 직사각형, 원, 레이블, 히트맵, 옵션	sgcMaps_Classes
폴더 타일 소스(z/x/y 파일)	sgcMaps_TileSource
공급자 목록과 sgcMapExpandTileURL	sgcMaps_Providers
지오메트리 함수	sgcMaps_Geometry
소프트웨어 래스터라이저, TsgcMapRasterizer	sgcMaps_Raster

컨트롤은 마커(사용자 지정 PNG 아이콘 포함), 클러스터, 도형, 레이블, 히트맵을 그리며, RenderToBitmap, SaveToBMP, SaveToPNG로 보기를 저장합니다.

포함되지 않는 항목

- **서버의 타일.** mtsHTTP 소스나 Provider가 있는 레이어는 Lazarus에서 타일을 표시하지 않습니다. Delphi에서 TsgcMapTileDownloader로, 또는 z/x/y 타일을 기록하는 아무 도구로 타일 폴더를 준비하여 애플리케이션과 함께 배포하십시오.
- **MBTiles와 벡터 타일.** 이 소스에는 FireDAC과 System.ZLib이 필요합니다.
- **GeoJSON, GPX, KML.** 유닛은 패키지에 나열되어 있지만 FPC에서는 비어 있는 상태로 컴파일됩니다.
- **비시각 구성 요소.** 온라인 서비스, 경로 계산기, 오프라인 라우터, 오프라인 지오코더, OSM 리더, TsgcMapLocation은 Delphi에서만 사용할 수 있습니다.
- **지도 회전.** Options.Bearing은 LCL에서 무시됩니다.
- **불투명도.** 레이어, 폴리라인, 채우기의 불투명도는 LCL에서 적용되지 않습니다.

예제

아래 프로그램은 실행 파일 옆의 타일 폴더에서 지도를 표시합니다. 새 Application 프로젝트를 만들고 필수 패키지에 sgcMapsLazarus를 추가한 다음, 메인 프로그램을 이 코드로 바꾸십시오.

tiles 폴더에는 줌 레벨마다 하위 폴더 하나, 열마다 하위 폴더 하나, 타일마다 이미지 하나가 있으며, 예를 들면 tiles\12\2072\1529.png입니다. 컨트롤이 없는 타일을 찾지 않도록 MaxZoom을 폴더에 있는 가장 높은 줌 레벨로 설정하십시오. 디스크에 있는 사본에도 타일 공급자의 약관이 여전히 적용되므로, 타일의 저작자 표시를 유지하십시오.

참고 항목

- TsgcVCLMap
- 오프라인 타일과 타일 다운로더
- 설치

.NET 개요

sgcMaps for .NET은 Delphi sgcMaps 라이브러리를 감싸는 래퍼가 아니라, 이를 관리 코드인 C#으로 옮긴 포트입니다. 같은 지도 모델, 같은 타일 공급자, 같은 온라인 서비스, 같은 오프라인 라우터와 지오코더를 갖추었으며, 브라우저 컨트롤이나 JavaScript 없이 SkiaSharp로 지도를 네이티브로 그립니다.
두 라이브러리의 파일은 호환됩니다. Delphi로 만든 경로 그래프(.sgcgraph), GeoNames 인덱스, 타일 폴더는 .NET에서 열리며, 그 반대도 마찬가지입니다. .NET QA 스위트는 Delphi 라이브러리가 기록한 경로 그래프와 GeoNames 인덱스로 이를 확인합니다.

패키지

이 포트는 NuGet 패키지 네 개로 나뉘어 있으므로, 프로젝트는 사용하는 것만 가져옵니다. 네 패키지 모두 버전이 **2.0.0**입니다. sgcMaps for .NET은 Delphi 릴리스와 별개로 자체 버전 번호를 유지합니다.

PackageId	포함 내용	대상 프레임워크	의존 대상
esegece.sgcMaps	핵심: 지도 모델(레이어, 마커, 폴리라인, 폴리곤, 직사각형, 원, 레이블, 히트맵, 옵션), 타일 계산과 지오메트리, 타일 공급자, 폴더, MBTiles, HTTP 타일 소스, 타일 다운로더, GeoJSON, GPX, KML 가져오기와 내보내기, 벡터 타일 및 스타일 리더, 그리고 온라인 서비스인 Geocoder, Directions, Places, Elevation, MapTimeZone, RouteCalculator입니다.	net8.0, net9.0	Microsoft.Data.Sqlite
esegece.sgcMaps.Rendering	**MapView**, 지도 엔진입니다. SkiaSharp로 래스터 및 벡터 타일, 오버레이, 클러스터, 히트맵, 팝업을 그리고, 포인터, 휠, 키보드 입력을 처리하며, 창 없이 이미지나 PNG 파일로 렌더링합니다.	net8.0, net9.0	esegece.sgcMaps, SkiaSharp, SkiaSharp.NativeAssets.Linux.NoDependencies
esegece.sgcMaps.WinForms	**SgcMapControl**, MapView를 호스팅하는 Windows Forms 지도 컨트롤입니다.	net8.0-windows, net9.0-windows	esegece.sgcMaps.Rendering
esegece.sgcMaps.Offline	오프라인 서비스: **OsmReader**(.osm.pbf 및 .osm XML), **RouteGraphBuilder**와 **OfflineRouter**, **GeoIndexBuilder**와	net8.0, net9.0	esegece.sgcMaps

	OfflineGeocoder입니다.		

Registered 에디션 전용

sgcMaps for .NET은 Registered(상용) 제품입니다. Community 에디션과 **.Community** 패키지는 없으며, 위의 네 패키지가 전부입니다.

호스트

- **Windows Forms**: Delphi TsgcVCLMap에 대응하는 SgcMapControl입니다.
- **창 없음**: MapView는 Windows, Linux, macOS에서 SkiaSharp 캔버스, 이미지, PNG 파일로 렌더링합니다. 콘솔 애플리케이션, 서비스, 웹 서버가 여기에 해당합니다.
- **자체 호스트**: MapView는 사용자 인터페이스 프레임워크에 의존하지 않습니다. 임의의 SKCanvas에 그리며 PointerDown, PointerMove, PointerUp, Wheel, Pinch, Rotate, KeyDown으로 입력을 받는데, SgcMapControl이 정확히 이 방식으로 MapView를 구동합니다.

WPF, Avalonia, MAUI 컨트롤은 없으며, FireMonkey 컨트롤 TsgcFMXMap에 대응하는 것도 없습니다.

네임스페이스

네임스페이스	내용
esegece.sgcMaps	지도 모델, MapCoord, MapColors, TileMath, MapGeometry, MapProviders, 타일 소스, TileDownloader, GeoJsonFormat, GpxFormat, KmlFormat입니다.
esegece.sgcMaps.Services	온라인 서비스(MapServiceBase, Geocoder, Directions, Places, Elevation, MapTimeZone), RouteCalculator, 그리고 GeocodingResult와 Route 같은 결과 클래스입니다.
esegece.sgcMaps.Vector	Mapbox Vector Tile 리더와 MapLibre 스타일 리더: VectorTile, VectorStyle, StyleEvaluator입니다.
esegece.sgcMaps.Rendering	MapView, VectorTileSource, VectorTileRenderer, 이벤트 인수 클래스입니다.
esegece.sgcMaps.WinForms	SgcMapControl.
esegece.sgcMaps.Offline	OsmReader, RouteGraphBuilder, OfflineRouter, GeoIndexBuilder, OfflineGeocoder입니다.

오프라인 클래스는 온라인 서비스의 결과 클래스를 공유합니다. OfflineRouter는 **Route**를 채우고 OfflineGeocoder는 **GeocodingResult** 객체를 반환하므로, 결과를 표시하는 코드는 둘 다에서 동작합니다. 그래서 오프라인 샘플도 esegece.sgcMaps.Services를 사용합니다.

해제해야 하는 클래스

이 클래스들은 파일, 연결, 스레드, 네이티브 SkiaSharp 객체를 보유하며 IDisposable을 구현합니다. Dispose를 호출하거나 using 문으로 만드십시오:

- **MapView**: 타일 로더와 마커 아이콘의 이미지입니다. **SgcMapControl**은 자신의 MapView, 백 버퍼, 타이머를 해제합니다.
- **Geocoder**, **Directions**, **Places**, **Elevation**, **MapTimeZone**: HttpMessageHandler로 만든 경우 자체 HttpClient를 가집니다.
- **TileDownloader**: 자신의 HttpClient입니다.
- **OfflineRouter**와 **OfflineGeocoder**: Dispose가 Close를 호출하여 경로 그래프나 SQLite 연결을 해제합니다.

참고 항목

- 빠른 시작: WinForms
- 헤드리스 렌더링
- 개요(Delphi)
- .NET 설치
- 클래스 매핑
- .NET과 Delphi의 차이점

.NET 설치

sgcUI 설치 프로그램은 Delphi, C++Builder, Lazarus 패키지를 설치합니다. sgcMaps for .NET은 대신 Registered 에디션 전용으로 자체 NuGet 패키지로 배포됩니다.

패키지 추가

사용하는 호스트의 패키지를 추가합니다. 각 패키지는 자신이 의존하는 패키지를 함께 가져오므로, Windows Forms 애플리케이션에는 esegece.sgcMaps.WinForms만, 헤드리스 렌더러에는 esegece.sgcMaps.Rendering만 필요합니다.

```
dotnet add package esegece.sgcMaps.WinForms --version 2.0.0
dotnet add package esegece.sgcMaps.Rendering --version 2.0.0
dotnet add package esegece.sgcMaps.Offline --version 2.0.0
dotnet add package esegece.sgcMaps --version 2.0.0
```

패키지	추가할 용도	대상 프레임워크
esegece.sgcMaps.WinForms	Windows Forms 폼의 지도 컨트롤입니다.	net8.0-windows, net9.0-windows
esegece.sgcMaps.Rendering	창 없이 렌더링한 지도이거나, 자체 호스트의 지도입니다.	net8.0, net9.0
esegece.sgcMaps.Offline	오프라인 라우팅, 오프라인 지오코딩, OpenStreetMap 파일 읽기입니다.	net8.0, net9.0
esegece.sgcMaps	지도 모델, 형식, 타일 다운로더, 온라인 서비스만 있으며 그리는 기능은 없습니다.	net8.0, net9.0

프로젝트 파일에서는 같은 줄이 다음과 같습니다:

```
<ItemGroup>
  <PackageReference Include="esegece.sgcMaps.WinForms" Version="2.0.0" />
  <PackageReference Include="esegece.sgcMaps.Offline" Version="2.0.0" />
</ItemGroup>
```

패키지가 피드가 아니라 .nupkg 파일로 제공되는 경우에는 먼저 그 폴더를 패키지 소스로 추가하십시오:

```
dotnet nuget add source C:\Packages\sgcMaps --name sgcMaps
```

요구 사항

- .NET 8 또는 .NET 9입니다. 패키지는 .NET Framework나 .NET Standard를 대상으로 하지 않습니다.
- Windows Forms 컨트롤에는 Windows가 필요합니다. 나머지 세 패키지는 Windows, Linux, macOS에서 실행됩니다.
- esegece.sgcMaps는 MBTiles 파일과 GeoNames 인덱스를 읽기 위해 **Microsoft.Data.Sqlite**를 사용합니다. 이를 위해 설치할 것은 없습니다.
- esegece.sgcMaps.Rendering은 **SkiaSharp**를 사용하며 **SkiaSharp.NativeAssets.Linux.NoDependencies**를 함께 가져오므로, 추가 네이티브 라이브러리 없이 Linux 서버나 컨테이너에서도 렌더링합니다.

Community 에디션 없음

sgcMaps for .NET에는 Community(무료) 에디션이 없으며 **.Community** 접미사가 붙은 패키지도 없습니다. 위의 모든 패키지에는 Registered 라이선스가 필요합니다.

참고 항목

- 빠른 시작: WinForms
- 설치(Delphi)
- .NET 개요
- 클래스 매핑
- .NET과 Delphi의 차이점

빠른 시작: WinForms

SgcMapControl은 Windows Forms 지도 컨트롤이며 Delphi TsgcVCLMap의 .NET 대응입니다. net8.0-windows와 net9.0-windows 용 **esegece.sgcMaps.WinForms** 패키지에 포함되어 있으며 esegece.sgcMaps.Rendering 과 esegece.sgcMaps를 함께 가져옵니다.
컨트롤은 지도 엔진인 MapView를 **Map** 속성에서 호스팅합니다. SkiaSharp로 보기를 백 버퍼에 그리고, 마우스, 휠, 키보드를 MapView로 전달합니다. 타일은 백그라운드 스레드에서 로드되므로 사용자 인터페이스가 네트워크를 기다리지 않으며, 타일이 도착하면 컨트롤이 UI 스레드에서 다시 그리고 TileError도 UI 스레드에서 발생시킵니다.

1. 프로젝트 만들기

```
dotnet new winforms -n MapQuickStart
cd MapQuickStart
dotnet add package esegece.sgcMaps.WinForms --version 2.0.0
```

2. 지도 표시

Program.cs를 아래 프로그램으로 바꾸고 Form1.cs와 Form1.Designer.cs를 삭제합니다. 디스크 캐시가 있는 OpenStreetMap 레이어를 추가하고, 지도를 마드리드로 중심 이동하며, 드래그할 수 있는 마커와 선을 추가합니다.

```
using System;
using System.IO;
using System.Windows.Forms;
using esegece.sgcMaps;
using esegece.sgcMaps.WinForms;

namespace QuickStartWinForms
{
    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            ApplicationConfiguration.Initialize();
            Application.Run(new MainForm());
        }
    }

    internal sealed class MainForm : Form
    {
        private readonly SgcMapControl _map = new SgcMapControl { Dock = DockStyle.Fill };

        public MainForm()
        {
            Text = "sgcMaps Quick Start";
            Width = 1000;
            Height = 700;
            Controls.Add(_map);

            // a base layer with the OpenStreetMap tiles, cached on disk
            Layer layer = _map.Layers.Add();
            layer.Name = "OpenStreetMap";
            layer.Provider = TileProvider.OpenStreetMap;
            MapProviders.ApplyProvider(layer);
            layer.CacheFolder = Path.Combine(AppContext.BaseDirectory, "tile_cache");

            _map.CenterLatitude = 40.4168;
            _map.CenterLongitude = -3.7038;
            _map.Zoom = 14;
```

```
            // a marker with a popup and a tooltip, and a line to Retiro Park
            Marker marker = _map.Map.AddMarker(40.4168, -3.7038, "Puerta del Sol", "Drag me");
            marker.Draggable = true;

            Polyline line = _map.Map.AddPolyline();
            line.AddPoint(40.4168, -3.7038);
            line.AddPoint(40.4153, -3.6844);
            line.Color = MapColors.Blue;
            line.Width = 4;

            _map.MapClick += (s, e) => Text = $"Click at {e.Latitude:F5}, {e.Longitude:F5}";
            _map.MarkerDragEnd += (s, e) => Text = $"Dropped at {e.Item.Latitude:F5}, {e.Item.Longitude:F5}";
            _map.TileError += (s, e) => Text = $"Tile {e.Zoom}/{e.X}/{e.Y}: {e.Error}";
        }
    }
}
```

MapProviders.ApplyProvider는 공급자의 설정인 TileSourceType, UrlTemplate, MaxZoom, Attribution을 레이어에 복사합니다. Provider가 Custom이 아닌 레이어는 호출이 없어도 해당 공급자의 타일을 다운로드하므로, 이 호출은 그 값을 읽거나 변경하려는 경우에만 필요합니다.

3. 실행

```
dotnet run
```

지도를 드래그하여 이동하고, 휠을 돌려 줌하고, 마커를 드래그합니다. 폼의 제목에는 마지막 클릭, 마커를 놓은 위치, 또는 실패한 타일이 표시됩니다.

속성

속성	형식	설명
Map	MapView	지도 엔진입니다. AddMarker, AddPolyline, FitToMarkers, SetLayerVisible, RenderToPng 같은 메서드를 여기서 호출합니다.
CenterLatitude, CenterLongitude	double	보기의 중심이며 단위는 도입니다.
Zoom	int	줌 레벨이며 Options.MinZoom과 Options.MaxZoom으로 제한됩니다.
Bearing	double	북쪽에서 시계 방향으로 잰 지도의 회전 각도입니다. Options.RotationEnabled가 true이면 사용자가 지도를 회전할 수 있습니다.
Options	MapOptions	보기의 동작과 장식: 줌 제한, TileSize, 확대/축소 버튼, 축척 막대, 저작자 표시, 배경색, 드래그, 줌, 회전, 레이어 전환기입니다.
MarkerOptions	MarkerOptions	마커의 기본 크기, 색상, 글꼴, 그리고 MarkerOptions.Clustering의 클러스터링 옵션입니다.
Layers	MapItemCollection<Layer>	타일 레이어입니다.
Markers, Polylines, Polygons, Circles, Rectangles, Labels, Heatmaps	MapItemCollection<T>	오버레이입니다. 새 항목은 Add()나 Map의 Add 메서드로 추가합니다.

Dock, Anchor, Enabled, Visible, TabStop 같은 일반적인 Control 속성도 사용할 수 있습니다.

이벤트

이벤트	인수	발생 시점
MapClick	MapClickEventArgs: Latitude, Longitude	지도를 클릭했을 때입니다.
MarkerClick	MapItemEventArgs<Marker>: Item	마커를 클릭했을 때입니다.
MarkerDragStart, MarkerDrag, MarkerDragEnd	MapItemEventArgs<Marker>: Item	Draggable이 true인 마커를 드래그했을 때입니다.
ShapeEdited	MapItemEventArgs<MapItem>: Item	사용자가 편집 가능한 폴리라인이나 폴리곤의 꼭짓점을 이동, 추가 또는 삭제했을 때입니다.
ClusterClick	ClusterClickEventArgs: Count, Latitude, Longitude, Handled	마커 클러스터를 클릭했을 때입니다. 클릭을 직접 처리하려면 Handled를 true로 설정하십시오.
LayerChanged	MapItemEventArgs<Layer>: Item	레이어가 표시되거나 숨겨졌을 때이며, 예를 들어 레이어 전환기에서 발생합니다.
ZoomChanged, BearingChanged	EventArgs	줌 레벨 또는 회전이 바뀔 때입니다.
TileError	TileErrorEventArgs: Layer, Zoom, X, Y, Error	타일을 불러올 수 없을 때이며, 예를 들어 타일 서버의 HTTP 오류입니다.

이것들은 일반적인 C# 이벤트이며, +=로 여러 핸들러를 연결할 수 있습니다.

참고 항목

- 헤드리스 렌더링
- .NET 데모
- 타일과 레이어
- .NET 개요
- .NET 설치
- 클래스 매핑
- .NET과 Delphi의 차이점

헤드리스 렌더링

MapView는 **esegece.sgcMaps.Rendering** 패키지에 있으며 sgcMaps for .NET의 지도 엔진입니다. SgcMapControl이 이를 창에 표시하지만 창은 전혀 필요 없습니다. SkiaSharp 캔버스, 이미지, PNG 파일로 렌더링합니다. 따라서 콘솔 애플리케이션, Windows 서비스, Linux 데몬, 웹 서버가 컨트롤과 같은 레이어와 오버레이로 정적 지도를 만들 수 있습니다. Delphi 컨트롤의 헤드리스 사용에 대응하는 .NET 기능이며, 정적 지도 렌더링에서 설명합니다.

PNG 파일로 렌더링

아래 프로그램은 에펠탑을 줌 15에서 마커와 원과 함께 렌더링하고 1024 x 768 PNG 파일을 기록합니다.

```
using System;
using System.IO;
using esegece.sgcMaps;
using esegece.sgcMaps.Rendering;

namespace StaticMap
{
    internal static class Program
    {
        private static int Main()
        {
            using var map = new MapView();
            map.Options.ShowZoomControls = false;

            Layer layer = map.Layers.Add();
            layer.Name = "osm";
            layer.Provider = TileProvider.OpenStreetMap;
            MapProviders.ApplyProvider(layer);
            layer.CacheFolder = Path.Combine(AppContext.BaseDirectory, "tile_cache");

            map.SetCenter(48.8584, 2.2945); // Eiffel Tower
            map.Zoom = 15;
            map.AddMarker(48.8584, 2.2945, "Eiffel Tower");
            map.AddCircle(48.8584, 2.2945, 300).FillOpacity = 0.2;

            // waits up to 30 seconds for the tiles, then writes the image
            map.RenderToPng("eiffel.png", 1024, 768, includeControls: false, tileTimeout: TimeSpan.FromSeconds(30));
            Console.WriteLine($"Missing tiles: {map.LastRenderMissingTiles}");
            Console.WriteLine($"Attribution: {map.AttributionText()}");
            return map.LastRenderMissingTiles == 0 ? 0 : 1;
        }
    }
}
```

RenderToPng는 보기를 그리고, 아직 로드 중인 타일을 기다린 다음, 모든 타일이 도착하거나 **tileTimeout**이 지날 때까지(지정하지 않으면 30초) 다시 그립니다. 호출 후에는 **LastRenderMissingTiles**가 이미지에서 누락된 타일 수를 알려 줍니다. 이후 보기의 크기가 복원되므로, 하나의 MapView로 여러 크기의 이미지를 렌더링할 수 있습니다.

메모리로 렌더링

RenderToImage는 파일을 쓰는 대신 SKImage를 반환합니다. 필요한 형식으로 인코딩하십시오. 예를 들어 웹 요청에 PNG 바이트로 응답할 수 있습니다:

```
using System;
using esegece.sgcMaps.Rendering;
using SkiaSharp;

public static class MapImages
{
    // renders the current view of a MapView into PNG bytes, for example for a web response
```

```
    public static byte[] RenderPng(MapView map, int width, int height)
    {
        using SKImage image = map.RenderToImage(width, height, includeControls: false,
            tileTimeout: TimeSpan.FromSeconds(10));
        using SKData data = image.Encode(SKEncodedImageFormat.Png, 100);
        return data.ToArray();
    }
}
```

자체 캔버스에 그리기

Render(SKCanvas canvas, int width, int height, bool includeControls)는 임의의 SkiaSharp 캔버스에 보기를 한 번 그리며 기다리지 않습니다. 아직 로드되지 않은 타일은 백그라운드 대기열에 들어가고, 도착하면 **Invalidated** 이벤트가 다음 그리기를 요청합니다. Invalidated는 작업자 스레드에서 발생할 수 있으므로, 호스트는 SgcMapControl이 하듯이 그리기 전에 자신의 UI 스레드로 전환해야 합니다.

그려지는 항목

- 타일 레이어, 벡터 타일 레이블, 히트맵, 폴리곤, 직사각형, 폴리라인, 원, 마커, 레이블 순서로 그려집니다.
- 확대/축소 버튼, 레이어 전환기, 나침반은 **includeControls**가 true일 때만 그려집니다. RenderToImage와 RenderToPng에서는 기본값이 false입니다.
- 축척 막대와 저작자 표시는 includeControls와 관계없이 **Options.ShowScaleBar**와 **Options.ShowAttribution**가 true이면 그려집니다. 저작자 표시는 계속 보이게 두십시오. 타일 공급자가 이를 요구합니다.

헤드리스 사용을 위한 멤버

멤버	설명
SetSize(int width, int height)	렌더링하기 전에 FitBounds, FitToMarkers, 좌표 변환에서 사용하는 보기의 크기를 설정합니다.
SetCenter(double latitude, double longitude), Zoom, Bearing	보기입니다.
FitBounds, FitToMarkers(int paddingPx), FitToPolyline(Polyline polyline, int paddingPx)	영역, 마커, 폴리라인이 보이도록 중심과 줌을 선택합니다.
AddMarker, AddPolyline, AddPolygon, AddCircle, AddRectangle, AddLabel	오버레이를 추가합니다. AddMarker는 팝업과 툴팁 텍스트를 받습니다.
Render, RenderToImage, RenderToPng	위에서 설명한 대로 보기를 그립니다.
WaitForTiles(TimeSpan timeout)	대기 중인 타일이 로드되거나 시간 제한이 지날 때까지 차단하며, 대기 중인 것이 없으면 true를 반환합니다.
PendingTileCount	백그라운드에서 로드 중인 타일의 수입니다.
LastRenderMissingTiles	마지막 렌더링에서 누락된 타일의 수입니다.
DrawnMarkerCount, DrawnClusterCount, DrawnLabelCount	마지막 렌더링이 그린 것입니다.
AttributionText()	보이는 레이어의 저작자 표시이며, 이미지에 그려지지 않을 때 이미지 옆에 출력합니다.
LatLonToScreen, ScreenToLatLon	좌표와 보기의 픽셀 사이를 변환합니다.

Linux와 macOS

Rendering 패키지는 Windows 의존성 없이 net8.0과 net9.0을 대상으로 하며, fontconfig나 다른 시스템 패키지가 필요 없는 Linux용 SkiaSharp 네이티브 라이브러리(SkiaSharp.NativeAssets.Linux.NoDependencies)를 함께 가져옵니다. 따라서 헤드리스 렌더러를 Linux 컨테이너에서 실행할 수 있습니다. 서버의 타일에는 아웃바운드 HTTPS 접근이 필요하며, 폴더나 MBTiles 파일의 타일에는 필요 없습니다.

참고 항목

- .NET 데모
- .NET 오프라인 데이터
- 05 Static Render(Delphi)
- .NET 개요
- .NET 설치
- 클래스 매핑
- .NET과 Delphi의 차이점

.NET 오프라인 데이터

sgcMaps for .NET은 네트워크 연결 없이 지도를 표시하고, 경로를 계산하고, 지오코딩할 수 있습니다. 디스크의 타일, OpenStreetMap으로 만든 경로 그래프, GeoNames 인덱스의 세 가지 로컬 데이터를 사용합니다. 모두 Delphi 라이브러리와 파일 호환이므로, Delphi 도구로 만든 데이터를 여기서 사용할 수 있으며 그 반대도 마찬가지입니다.

데이터	만드는 방법	사용하는 곳	패키지
타일 폴더(z/x/y)	TileDownloader 또는 아무 타일 도구	TileSourceType.Folder를 사용하는 레이어	esegece.sgcMaps
MBTiles 파일, 래스터 또는 벡터	아무 MBTiles 도구	TileSourceType.MBTiles 또는 TileSourceType.VectorMBTiles를 사용하는 레이어	esegece.sgcMaps, 벡터 타일에는 esegece.sgcMaps.Rendering
경로 그래프 (.sgcgraph)	RouteGraphBuilder	OfflineRouter	esegece.sgcMaps.Offline
GeoNames 인덱스 (SQLite)	GeoIndexBuilder	OfflineGeocoder	esegece.sgcMaps.Offline

오프라인 타일

레이어는 **TileSourceType**이 Folder 또는 MBTiles이고 **TileSourcePath**가 그것을 가리키면 폴더나 MBTiles 파일에서 타일을 읽습니다. 벡터 MBTiles 파일은 **VectorStyleFile**의 MapLibre 스타일로 그려지며, 비어 있으면 내장 스타일로 그려집니다. 오프라인 타일과 벡터 타일을 참조하십시오.

```
using esegece.sgcMaps;
using esegece.sgcMaps.Rendering;

public static class OfflineLayers
{
    public static void Add(MapView map)
    {
        // raster tiles from a z/x/y folder, such as the one TileDownloader fills
        Layer streets = map.Layers.Add();
        streets.Name = "Streets";
        streets.TileSourceType = TileSourceType.Folder;
        streets.TileSourcePath = "tiles";
        streets.MaxZoom = 15;

        // raster tiles from an MBTiles file, as a second base layer
        Layer satellite = map.Layers.Add();
        satellite.Name = "Satellite";
        satellite.TileSourceType = TileSourceType.MBTiles;
        satellite.TileSourcePath = "satellite.mbtiles";
        satellite.Visible = false;

        // vector tiles from an MBTiles file, drawn with a MapLibre style
        Layer vector = map.Layers.Add();
        vector.Name = "Vector";
        vector.TileSourceType = TileSourceType.VectorMBTiles;
        vector.TileSourcePath = "spain.mbtiles";
        vector.VectorStyleFile = "style.json";
        vector.Visible = false;

        map.Options.ShowLayerSwitcher = true;
```

```
    }
}
```

타일 팩 다운로드

TileDownloader는 한 영역과 줌 레벨 범위의 타일을 z/x/y 폴더로 다운로드하며, 이미 있는 타일은 건너뜁니다. 의도적으로 기본 UrlTemplate이 없습니다. tile.openstreetmap.org와 대부분의 공개 타일 서버는 대량 다운로드를 금지하므로, 자체 서버나 약관이 허용하는 공급자를 가리키게 하고 DelayMS는 0보다 크게 유지하십시오.

```
using System;
using System.Threading.Tasks;
using esegece.sgcMaps;

public static class TilePack
{
    public static async Task DownloadAsync()
    {
        using var downloader = new TileDownloader();
        // your own tile server, or a provider whose terms allow bulk downloads
        downloader.UrlTemplate = "https://tiles.example.com/{z}/{x}/{y}.png";
        downloader.OutputPath = "tiles";
        downloader.DelayMS = 250;

        // the center of Madrid, zoom 10 to 15
        Console.WriteLine($"{TileDownloader.CountTiles(40.38, -3.75, 40.46, -3.65, 10, 15)} tiles");
        var progress = new Progress<TileDownloadProgress>(p =>
            Console.WriteLine($"zoom {p.CurrentZoom}: {p.Downloaded} downloaded, {p.Skipped} skipped, {p.Failed} failed"));
        TileDownloadProgress result = await downloader.DownloadAsync(40.38, -3.75, 40.46, -3.65, 10, 15, progress);
        Console.WriteLine($"Done: {result.Downloaded} of {result.TotalTiles} tiles downloaded");
    }
}
```

오프라인 라우팅

RouteGraphBuilder는 OpenStreetMap 추출 파일(.osm.pbf 또는 .osm XML)을 읽고 도로망을 자동차, 자전거, 도보 프로필이 있는 .sgcgraph 파일로 기록합니다. **OfflineRouter**는 그 파일을 불러와 네트워크 없이 경로에 응답합니다. 경로는 온라인 Directions 서비스가 반환하는 것과 같은 클래스인 **Route**이며, 거리(미터), 소요 시간(초), 점, 구간, 단계를 가집니다.

```
using System;
using esegece.sgcMaps;
using esegece.sgcMaps.Offline;
using esegece.sgcMaps.Services;

public static class OfflineRouting
{
    public static void Run(string osmFile, string graphFile, MapCoord from, MapCoord to)
    {
        // build the route graph once, from an .osm.pbf extract or an .osm XML file
        var builder = new RouteGraphBuilder();
        builder.Build(osmFile, graphFile);
        Console.WriteLine($"Graph: {builder.VertexCount} vertices, {builder.EdgeCount} edges");

        // then route with no network at all
        using var router = new OfflineRouter(graphFile);
        router.Profile = OfflineRouterProfile.Car;
        router.Open();

        var route = new Route();
        if (router.GetRoute(route, from, to))
            Console.WriteLine($"Route: {route.Distance:F0} m, {route.Duration:F0} s, {route.Points.Count} points");
        else
            Console.WriteLine("No route: " + router.LastError);
```

```
    }
}
```

멤버	설명
RouteGraphBuilder.Build(string inputFile, string outputFile)	그래프를 빌드합니다. 통계는 WayCount, NodeCount, MissingNodes, VertexCount, EdgeCount, ShapeCount, NameCount에 있으며, Progress가 읽은 바이트 수를 보고합니다.
OfflineRouter.GraphFile, Open(), Close(), Active	그래프 파일과 그것을 열고 닫는 것입니다. Active는 읽기 전용이며, 파일 이름을 받는 생성자는 GraphFile만 설정합니다.
OfflineRouter.Profile	OfflineRouterProfile.Car(기본값), Bike 또는 Foot입니다.
OfflineRouter.GetRoute(Route route, params MapCoord[] waypoints)	둘 이상의 경유지를 지나는 경로를 계산합니다. 경로가 없으면 false를 반환하고 LastError를 설정합니다. IReadOnlyList<MapCoord>를 받는 오버로드도 있습니다.

국가 전체의 그래프를 빌드하는 데는 시간과 메모리가 필요하므로, 한 번 빌드하고 .sgcgraph 파일을 배포하십시오. 빌드 단계는 오프라인 라우팅을 참조하십시오.

오프라인 지오코딩

GeoIndexBuilder는 GeoNames 덤프를 SQLite 인덱스로 만들고, **OfflineGeocoder**는 이름이나 위치로 그 인덱스를 검색합니다. GeoNames 내보내기 서버에서 cities1000.txt 같은 덤프와 admin1CodesASCII.txt, countryInfo.txt를 다운로드하십시오. 결과는 온라인 Geocoder와 마찬가지로 **GeocodingResult** 객체입니다.

```
using System;
using esegece.sgcMaps.Offline;
using esegece.sgcMaps.Services;

public static class OfflineGeocoding
{
    public static void Run(string citiesFile, string admin1File, string countryInfoFile, string databaseFile)
    {
        // build the GeoNames index once
        var indexBuilder = new GeoIndexBuilder
        {
            InputFile = citiesFile,
            Admin1CodesFile = admin1File,
            CountryInfoFile = countryInfoFile,
            DatabaseFile = databaseFile,
            MinPopulation = 1000
        };
        indexBuilder.Build();
        Console.WriteLine($"Index: {indexBuilder.PlacesWritten} places, {indexBuilder.NamesWritten} names");

        // then geocode and reverse geocode with no network at all
        using var geocoder = new OfflineGeocoder(databaseFile);
        geocoder.MaxResults = 3;
        geocoder.Open();
        foreach (GeocodingResult r in geocoder.Geocode("Madrid"))
            Console.WriteLine($"{r.DisplayName}: {r.Latitude:F4}, {r.Longitude:F4}");
        foreach (GeocodingResult r in geocoder.ReverseGeocode(41.3874, 2.1686))
            Console.WriteLine($"Nearest place: {r.DisplayName}");
    }
}
```

멤버	기본값	설명
GeoIndexBuilder.InputFile, DatabaseFile		읽을 GeoNames 덤프와 기록할 인덱스입니다.

GeoIndexBuilder.Admin1CodesFile, CountryInfoFile		코드를 지역 및 국가 이름으로 바꾸는 파일입니다.
GeoIndexBuilder.FeatureClasses	PA	가져올 GeoNames 피처 클래스입니다. P는 거주 지역, A는 행정 구역입니다.
GeoIndexBuilder.MinPopulation	0	인구가 더 적은 장소를 건너뜁니다.
GeoIndexBuilder.IncludeAlternateNames	true	다른 언어의 이름 같은 대체 이름도 인덱싱합니다.
GeoIndexBuilder.BatchSize	2000	배치마다 기록하는 행 수입니다.
OfflineGeocoder.MaxResults	5	결과의 최대 수입니다.
OfflineGeocoder.CountryCodes		모든 검색을 제한하는, ES,PT 같은 쉼표로 구분한 ISO alpha-2 국가 코드입니다.
OfflineGeocoder.Geocode, ReverseGeocode, ReverseGeocodeCountry		이름으로 검색하고, 위치에서 가장 가까운 장소를 검색하고, 위치의 국가를 반환합니다.

OfflineGeocoder는 OfflineRouter와 마찬가지로 Open()으로 열고 Close() 또는 Dispose()로 닫습니다.

OpenStreetMap 파일 읽기

OsmReader는 .osm.pbf 또는 .osm XML 파일을 스트리밍하며, 각 요소에 대해 태그와 함께 **Node**, **Way**, **Relation**을 발생시키고 읽은 바이트 수와 함께 **Progress**를 발생시킵니다. ReadNodes, ReadWays, ReadRelations, SkipNodeTags, WayFilter는 필요 없는 것을 건너뜁니다. RouteGraphBuilder가 이를 기반으로 하며, OpenStreetMap 데이터의 다른 가져오기에도 사용할 수 있습니다.

참고 항목

- .NET 데모
- 오프라인 지오코딩
- .NET 개요
- .NET 설치
- 클래스 매핑
- .NET과 Delphi의 차이점

클래스 매핑

표는 sgcMaps의 각 Delphi 구성 요소와 클래스를 .NET 대응 항목에 매핑합니다. .NET 클래스는 Tsgc 접두사를 없애고 일반적인 .NET 이름을 사용하며, 개념이 같은 곳에서는 멤버가 Delphi 이름을 유지합니다.

지도 컨트롤과 모델

Delphi	.NET	패키지
TsgcVCLMap	SgcMapControl(Windows Forms)이며 엔진은 SgcMapControl.Map에 있습니다	esegece.sgcMaps.WinForms
TsgcFMXMap	대응 항목이 없습니다. MapView는 SkiaSharp로 그리는 어떤 프레임워크에서도 호스팅할 수 있습니다.	esegece.sgcMaps.Rendering
창 없이 렌더링하는 지도 컨트롤, TsgcMapRasterizer	MapView: Render, RenderToImage, RenderToPng	esegece.sgcMaps.Rendering
TsgcMapLayer, TsgcMapMarker, TsgcMapPolyline, TsgcMapPolygon, TsgcMapRectangle, TsgcMapCircle, TsgcMapLabel, TsgcMapHeatmap	Layer, Marker, Polyline, Polygon(구멍은 PolygonHoles), Rectangle, Circle, Label, Heatmap	esegece.sgcMaps
TsgcMapLayers, TsgcMapMarkers와 그 밖의 컬렉션	MapItemCollection<T>	esegece.sgcMaps
TsgcMapOptions, TsgcMapMarkerOptions, TsgcMapClusterOptions	MapOptions, MarkerOptions, ClusterOptions	esegece.sgcMaps
TsgcMapCoord, TsgcMapHeatmapPoint	MapCoord, HeatmapPoint(readonly record struct)	esegece.sgcMaps

타일, 공급자, 형식

Delphi	.NET	패키지
TsgcMapTileProvider, TsgcMapTileSourceType	TileProvider, TileSourceType	esegece.sgcMaps
sgcMapApplyProvider, sgcMapGetProviderInfo, sgcMapProviderNames, sgcMapExpandTileURL, TsgcMapProviderInfo	MapProviders.ApplyProvider, GetInfo, ProviderNames, ExpandTileUrl, ProviderInfo	esegece.sgcMaps
폴더, MBTiles, HTTP 타일 소스	FolderTileSource, MBTilesTileSource, HttpTileSource (ITileSource), TileSourceFactory	esegece.sgcMaps
TsgcMapTileSource_MVT	VectorTileSource	esegece.sgcMaps.Rendering

TsgcMapVectorTile, TsgcMapVectorLayer, TsgcMapVectorFeature, TsgcMapVectorStyle	VectorTile, VectorLayer, VectorFeature, VectorStyle(네임스페이스 esegece.sgcMaps.Vector)	esegece.sgcMaps
TsgcMapTileDownloader	TileDownloader	esegece.sgcMaps
TsgcMapGeoJSON, TsgcMapGPX, TsgcMapKML	GeoJsonFormat, GpxFormat, KmlFormat	esegece.sgcMaps
TsgcMapGPXTrackStats, TsgcMapElevationGradient	GpxTrackStats, ElevationGradient	esegece.sgcMaps
sgcMaps_Geometry의 함수 (sgcMapDistance, sgcMapInitialBearing, sgcMapDestinationPoint, sgcMapGeodesicInterpolate, sgcMapPointInPolygon, sgcMapSimplify, sgcMapNearestPointOnPolyline, sgcMapClusterPoints, sgcMapRenderHeatmap)	정적 클래스 MapGeometry: Distance, InitialBearing, DestinationPoint, GeodesicInterpolate, PointInPolygon, Simplify, NearestPointOnPolyline, ClusterPoints, RenderHeatmap	esegece.sgcMaps
sgcMaps_Classes의 타일 계산 (sgcMapLatLonToTileF, sgcMapTileFToLatLon, sgcMapLatLonToPixel, sgcMapPixelToLatLon, sgcMapTileCount)	정적 클래스 TileMath: TryLatLonToTile, TryTileToLatLon, TryLatLonToPixel, TryPixelToLatLon, TileCount	esegece.sgcMaps

온라인 서비스와 라우팅

Delphi	.NET	패키지
TsgcMapService	MapServiceBase	esegece.sgcMaps
TsgcMapGeocoder	Geocoder와 GeocodingResult, GeocodingProvider	esegece.sgcMaps
TsgcMapDirections	Directions와 Route, RouteLeg, RouteStep, DirectionsProvider, TravelMode, PolylineCodec	esegece.sgcMaps
sgcMapAddRouteToPolylines	Route.AddToPolylines	esegece.sgcMaps
TsgcMapPlaces	Places와 Place, PlacesProvider	esegece.sgcMaps
TsgcMapElevation	Elevation과 ElevationResult, ElevationProvider	esegece.sgcMaps
TsgcMapTimeZone	MapTimeZone과 MapTimeZoneInfo, TimeZoneProvider	esegece.sgcMaps
TsgcMapRouteCalculator	RouteCalculator	esegece.sgcMaps

오프라인

Delphi	.NET	패키지

TsgcMapOfflineRouter	OfflineRouter와 OfflineRouterProfile	esegece.sgcMaps.Offline
TsgcMapRouteGraphBuilder, RouteGraphBuilder 도구	RouteGraphBuilder	esegece.sgcMaps.Offline
TsgcMapOfflineGeocoder	OfflineGeocoder	esegece.sgcMaps.Offline
TsgcMapGeoIndexBuilder, GeoIndexBuilder 도구	GeoIndexBuilder	esegece.sgcMaps.Offline
TsgcMapOSMReader, TsgcMapOSMTags, TsgcMapOSMMember	OsmReader, OsmTags, OsmMember	esegece.sgcMaps.Offline

.NET에 대응 항목이 없는 것

- TsgcMapLocation, 기기의 위치입니다.
- FireMonkey 컨트롤과 Lazarus 컨트롤입니다.
- GeoIndexBuilder와 RouteGraphBuilder 명령줄 도구입니다. 같은 이름의 클래스를 호출하거나, 두 파일을 모두 빌드하는 OfflineDemo를 사용하십시오.

참고 항목

- .NET 개요
- .NET 설치
- .NET과 Delphi의 차이점

.NET과 Delphi의 차이점

.NET 포트는 Delphi 라이브러리의 모델, 동작, 파일 형식을 유지합니다. 달라지는 것은 .NET 고유의 관례가 있는 곳의 API 형태입니다.

이름

- 클래스는 Tsgc 접두사를 뗍니다. TsgcMapGeocoder는 Geocoder가 되고 TsgcVCLMap은 SgcMapControl이 됩니다. 클래스 매핑에 전체 목록이 있습니다.
- 열거형도 접두사를 뗍니다. mtpOpenStreetMap은 TileProvider.OpenStreetMap이 되고, mtsMBTiles는 TileSourceType.MBTiles가 됩니다.
- 일부 멤버는 UrlTemplate과 NominatimUrl처럼 .NET 표기법을 따릅니다. 대부분은 Delphi 이름을 유지합니다.

엔진 하나, 컨트롤 하나

Delphi에서는 지도 컨트롤이 지도를 가집니다. .NET에서는 지도가 창 없이 동작하는 **MapView**이고, **SgcMapControl**은 Map 속성에 MapView를 하나 호스팅하는 Windows Forms 컨트롤입니다. 컨트롤은 보기와 컬렉션(Zoom, Layers, Markers...)을 직접 노출하며, AddMarker, FitToMarkers, RenderToPng 같은 메서드는 SgcMapControl.Map에서 호출합니다.

온라인 서비스는 비동기

Delphi에서는 서비스의 각 요청에 동기 메서드와, OnGeocodeResult 같은 결과 이벤트로 응답하는 Async 메서드가 있습니다. .NET에서는 각 요청이 결과를 반환하는 Task 기반 메서드 하나입니다. GeocodeAsync와 ReverseGeocodeAsync, GetRouteAsync, SuggestAsync 와 RetrieveAsync, GetElevationsAsync 와 GetElevationAlongPathAsync, GetTimeZoneAsync, RouteCalculator.CalculateAsync입니다. 모두 선택적인 CancellationToken을 받습니다. 결과 이벤트는 없으며, 콘솔 애플리케이션에서 CheckSynchronize도 필요 없습니다.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using esegece.sgcMaps;
using esegece.sgcMaps.Services;

public static class OnlineServices
{
    public static async Task ShowRouteAsync(MapItemCollection<Polyline> polylines)
    {
        using var geocoder = new Geocoder(); // Nominatim by default
        geocoder.UserAgent = "MyCompany.MyApp/1.0";
        using var directions = new Directions(); // OSRM by default
        try
        {
            IReadOnlyList<GeocodingResult> places = await geocoder.GeocodeAsync("Plaza Mayor, Madrid");
            if (places.Count == 0)
            {
                Console.WriteLine(geocoder.LastError); // "No results"
                return;
            }

            IReadOnlyList<Route> routes = await directions.GetRouteAsync(new[]
            {
                new MapCoord(places[0].Latitude, places[0].Longitude),
                new MapCoord(40.4153, -3.6844)
            });
            routes[0].AddToPolylines(polylines);
            Console.WriteLine($"{routes[0].Distance / 1000:F2} km, {routes[0].Duration / 60:F0} min");
        }
        catch (ServiceException e)
```

```
        {
            // LastError holds the same text, and the Error event was raised
            Console.WriteLine($"HTTP {e.StatusCode}: {e.Message}");
        }
    }
}
```

- 실패한 요청은 HTTP 상태를 StatusCode에 담은 **ServiceException**을 발생시킵니다(HTTP 응답이 없으면 0). 또한 LastError를 설정하고 Delphi의 OnError처럼 Error 이벤트를 발생시킵니다.
- 결과가 없는 요청은 오류가 아닙니다. 목록이 비어 있으며 Geocoder는 LastError를 "No results"로 설정합니다.
- 서비스는 기본적으로 HttpClient 하나를 공유합니다. HttpMessageHandler나 HttpClient를 받는 생성자를 사용하면 프록시나 테스트 핸들러를 사용할 수 있습니다.
- Nominatim, OSRM 데모 서버, OpenTopoData에 대한 요청의 프로세스 전체 간격(초당 한 건)은 Delphi와 같습니다.

이벤트

이벤트는 MapClickEventArgs를 가진 MapClick처럼 EventHandler<T> 형식의 C# 이벤트입니다. Delphi 이벤트는 메서드 하나를 담는 반면, 여러 핸들러를 +=로 연결하고 -=로 제거할 수 있습니다. Delphi 이벤트 매개변수의 데이터는 인수의 속성입니다. 마커는 e.Item, 클릭은 e.Latitude와 e.Longitude, 기본 동작을 중지하려면 e.Handled입니다.

색상

색상은 0xAARRGGBB 순서의 ARGB uint 값이며, 예를 들면 MapColors.Blue(0xFF0000FF)입니다. Delphi 모델은 TColor를 사용합니다. **MapColors.FromTColor**와 **MapColors.ToTColor**가 둘 사이를 변환하고, **MapColors.FromRgb**는 불투명한 색상을 만듭니다.

좌표와 컬렉션

- MapCoord와 HeatmapPoint는 readonly record struct이며 new MapCoord(latitude, longitude)로 만듭니다.
- 오버레이 컬렉션은 MapItemCollection<T>이며, Add(), BeginUpdate, EndUpdate, Changed 이벤트가 있는 Collection<T>입니다.
- 마커 아이콘은 Marker.Icon에 담긴 이미지 파일의 내용이며 바이트 배열이고, LoadIconFromFile 또는 LoadIconFromStream으로 불러옵니다.

오프라인 라우터와 지오코더

OfflineRouter와 OfflineGeocoder는 **Open()**으로 열고 **Close()** 또는 Dispose()로 닫습니다. Active 속성은 읽기 전용이며, Delphi에서는 Active를 True로 설정하면 열립니다. OfflineRouter.GetRoute와 OfflineGeocoder.Geocode는 Delphi와 마찬가지로 동기 방식입니다.

렌더링

엔진은 모든 플랫폼에서 SkiaSharp로 그립니다. RenderToPng는 PNG 파일을 기록하며, 다른 형식은 RenderToImage의 SKImage를 인코딩하십시오. 헤드리스 렌더링을 참조하십시오.

플랫폼과 버전

- 패키지는 .NET 8과 .NET 9를 대상으로 하고, 컨트롤은 net8.0-windows와 net9.0-windows를 대상으로 합니다. .NET Framework용 빌드는 없습니다.
- 패키지의 버전은 2.0.0이며 Delphi 버전과 별개입니다.
- Community 에디션은 없습니다. .NET 설치를 참조하십시오.
- TsgcMapLocation, FireMonkey 컨트롤, 명령줄 도구에는 대응 항목이 없습니다.

참고 항목

- .NET 개요
- .NET 설치
- 클래스 매핑

.NET 데모

sgcMaps for .NET에는 데모 세 개가 있으며 sgcUI의 **.net\demos\Maps** 폴더에 있습니다. 각 데모는 net8.0 또는 net8.0-windows용 작은 SDK 스타일 프로젝트입니다. 라이브러리 소스가 옆(.net\source\Maps)에 있으면 sgcMaps 프로젝트를 참조하고, 그렇지 않으면 버전 2.0.0의 NuGet 패키지를 참조합니다.

데모	형식	패키지	보여 주는 내용
ConsoleRenderDemo	콘솔, net8.0	esegece.sgcMaps, esegece.sgcMaps.Rendering	창 없이 PNG 파일로 렌더링한 지도이며, 마커와 온라인 Directions 서비스의 경로가 있습니다.
OfflineDemo	콘솔, net8.0	esegece.sgcMaps, esegece.sgcMaps.Offline	오프라인 라우팅과 오프라인 지오코딩이며, 경로 그래프와 GeoNames 인덱스의 빌드를 포함합니다.
WinFormsDemo	Windows Forms, net8.0-windows	esegece.sgcMaps.WinForms	기본 레이어 두 개, 마커, 편집 가능한 도형, 원, 히트맵, 이벤트가 있는 지도 컨트롤입니다.

ConsoleRenderDemo

Delphi 데모 05 Static Render의 .NET 버전입니다. 인수 없이 실행하면 줌 13의 마드리드를, 공개 OSRM 서버에서 가져온 푸에르타 델 솔에서 레티로 공원까지의 자동차 경로와 함께 렌더링하고, 현재 폴더에 **madrid.png**를 기록합니다. 타일은 실행 파일 옆의 tile_cache 폴더에 캐시됩니다.

```
dotnet run --project ConsoleRenderDemo
dotnet run --project ConsoleRenderDemo -- --center 48.8584,2.2945 --zoom 15 --size 800x600 --provider carto --marker 48.858
```

인수	설명
--center lat,lon	지도의 중심입니다.
--zoom z	줌 레벨이며 기본값은 13입니다.
--size WxH	이미지 크기이며 기본값은 1024x768입니다.
--provider osm|opentopo|esri|carto	타일 공급자이며 기본값은 osm입니다.
--marker lat,lon	마커를 추가합니다. 반복해서 지정할 수 있습니다.
--route lat,lon;lat,lon[;...]	OSRM 서버에 점들을 지나는 경로를 요청하고 그립니다.
--out file.png	출력 파일이며 기본값은 madrid.png입니다.

데모는 공급자, 보기, 경로, 파일의 전체 경로, 저작자 표시 텍스트를 출력합니다. 타일과 경로를 위해 인터넷 접근이 필요합니다.

OfflineDemo

네트워크 호출 없이 경로를 계산하고 지오코딩합니다. OpenStreetMap 추출 파일에서 경로 그래프를 빌드하거나 .sgcgraph 파일이 있으면 재사용하며, 두 지점 사이를 자동차로 라우팅합니다. GeoNames 인수를 지정하면 GeoNames 인덱스도 빌드하거나 재사용하고 질의를 지오코딩합니다.

```
dotnet run --project OfflineDemo -- --pbf andorra-latest.osm.pbf --from 42.5063,1.5218 --to 42.5350,1.5807
dotnet run --project OfflineDemo -- --pbf andorra-latest.osm.pbf --from 42.5063,1.5218 --to 42.5350,1.5807 --geonames citie
```

인수	설명
--pbf file.osm.pbf	OpenStreetMap 추출 파일입니다. 필수입니다.
--from lat,lon, --to lat,lon	경로의 시작과 끝입니다. 필수입니다.
--graph file.sgcgraph	경로 그래프입니다. 기본값은 추출 파일 이름에 .sgcgraph 확장자를 붙인 것입니다.
--geonames, --admin1, --countryinfo	GeoNames 덤프, admin1CodesASCII.txt, countryInfo.txt입니다.
--geocode "query"	검색할 장소입니다.
--db file.sqlite	GeoNames 인덱스입니다. 기본값은 덤프 이름에 .sqlite 확장자를 붙인 것입니다.

추출 파일은 Geofabrik 같은 OpenStreetMap 다운로드 서버에서, GeoNames 파일은 GeoNames 내보내기 서버에서 받을 수 있습니다. .NET 오프라인 데이터를 참조하십시오.

WinFormsDemo

SgcMapControl과 상태 표시줄이 있는 폼입니다. 다음을 보여 줍니다:

- OpenStreetMap과 OpenTopoMap 두 기본 레이어, 레이어 전환기, 활성화된 회전.
- 마드리드와 바르셀로나의 드래그할 수 있는 마커, 발렌시아의 마커.
- 마드리드에서 사라고사를 거쳐 바르셀로나까지의 편집 가능한 폴리라인과 편집 가능한 폴리곤.
- 발렌시아 주변의 40 km 원, 세비야 주변의 가중치 점 200개로 이루어진 히트맵.
- 상태 표시줄에 기록되는 MarkerClick, MapClick, MarkerDragEnd, ShapeEdited, ZoomChanged, TileError 이벤트.

폼이 표시되면 지도가 마커에 맞춰집니다. 타일은 임시 폴더의 sgcMaps_tiles 폴더에 캐시됩니다.

```
dotnet run --project WinFormsDemo
```

참고 항목

- 빠른 시작: WinForms
- 헤드리스 렌더링
- 데모(Delphi)
- .NET 개요
- .NET 설치
- 클래스 매핑
- .NET과 Delphi의 차이점

문제 해결

아래 문제는 sgcMaps에서 가장 자주 만나는 문제이며, 원인과 해결 방법을 함께 설명합니다.

지도에 타일이 표시되지 않음

- **OnTileError**를 처리하십시오. 이 이벤트는 실패한 타일마다 한 번씩, 레이어, 타일 좌표, 오류(예: HTTP 403 Forbidden)와 함께 발생합니다. 서버에 없는 타일(HTTP 404)은 오류가 아닙니다. 폴더에서 누락된 타일과 마찬가지로 **LastRenderMissingTiles**에서만 집계됩니다.
- 레이어를 전혀 열 수 없으면 지도는 "Tile source not available"이라는 텍스트 뒤에 이유를 보기에 그립니다. 가장 흔한 것은 "Tile source directory not found"입니다. 레이어에 Provider가 없고 TileSourceType이 기본값인 mtsFolder 그대로이며, 존재하지 않는 TileSourcePath가 있는 경우입니다.
- 서버의 타일에는 Delphi 2009 이상이 필요하며, Lazarus 컨트롤은 폴더 타일만 표시합니다(Lazarus 참조). MBTiles와 벡터 MBTiles에는 Delphi XE7이 필요하며, 더 오래된 버전에서는 레이어가 이 컴파일러에서 타일 소스 형식을 사용할 수 없다고 보고합니다.
- Delphi 2009부터 XE7까지는 타일을 Indy로 다운로드하며, https에는 실행 파일 옆에 OpenSSL 1.0.2 라이브러리(libeay32.dll과 ssleay32.dll)가 필요합니다. 이 라이브러리가 없으면 OpenSSL 라이브러리를 불러올 수 없다는 오류가 표시됩니다. Delphi XE8부터는 지도가 THTTPClient를 사용하므로 DLL이 필요 없습니다.
- 실패한 타일은 저장되지 않으므로 지도는 다음 그리기에서 다시 요청합니다. 레이어를 다시 열 때까지는 같은 타일에 대해 OnTileError가 두 번 발생하지 않습니다.

API 키

- MapTiler, Thunderforest, Stadia 타일 공급자에는 레이어의 **ApiKey** 속성에 API 키가 필요합니다. 키가 없으면 OnTileError가 "This tile provider requires an API key (ApiKey)"를 보고하고, 키가 잘못되면 공급자가 HTTP 401 또는 403으로 응답합니다.
- 온라인 서비스는 요청을 보내기 전에 키를 검사하지 않습니다. 공급자의 오류는 **LastError**(동기 메서드) 또는 **OnError**(비동기 메서드)로 전달되며, 예를 들어 공급자의 설명이 포함된 HTTP 403입니다.
- GeoNames를 사용하는 TsgcMapTimeZone은 ApiKey에 키가 아니라 GeoNames 사용자 이름을 받습니다.
- 데모나 공개 소스 코드에 키를 넣어 배포하지 마십시오. sgcMaps 데모에는 대신 키를 입력하는 편집 상자가 있습니다.

사용 정책과 사용자 에이전트

무료 공개 서버는 자원봉사자나 후원자가 운영하며, 각각 사용 정책이 있습니다. 정책을 위반하는 애플리케이션은 경고 없이 차단될 수 있습니다.

- **tile.openstreetmap.org**는 대화형 보기만 허용합니다. 대량 다운로드와 영역 선행 가져오기는 금지되고, 식별 가능한 사용자 에이전트와 보이는 저작자 표시가 필요합니다. 지도 컨트롤의 타일 요청은 sgcMaps/1.0으로 자신을 식별합니다. 운영 애플리케이션에는 상용 공급자, 자체 타일 서버, 또는 오프라인 타일을 사용하십시오.
- **TsgcMapTileDownloader**에는 의도적으로 기본 URLTemplate이 없습니다. 자체 서버나 약관이 대량 다운로드를 허용하는 공급자를 가리키게 하고, 절대 tile.openstreetmap.org를 가리키지 마십시오. DelayMS는 0보다 크게 유지하십시오(기본값 250 ms). 다운로더는 sgcMapsDownloader/1.0으로 자신을 식별합니다.
- 온라인 서비스는 기본적으로 **UserAgent**에 sgcMaps/2.0을 보냅니다. 이를 애플리케이션 이름으로 설정하십시오. Nominatim은 식별 가능한 사용자 에이전트를 요구합니다.
- Nominatim과 OpenTopoData에 대한 요청은 정책에 따라 구성 요소가 전체 프로세스에서 초당 한 건으로 간격을 둡니다. 따라서 메인 스레드에서 동기 호출을 하면 최대 1초까지 기다릴 수 있으므로, 사용자 인터페이스에서는 Async 메서드를 사용하십시오.
- TsgcMapDirections의 기본 OSRM 서버는 공개 데모 서버이며, 가벼운 사용만 가능하고 초당 요청 한 건, 자동차 프로필만 지원합니다. 도보와 자전거에는 자체 OSRM 서버나 다른 공급자를 사용하고, **ValidateRequest** 또는 **IsTravelModeSupported**를 호출하여 공급자에 대해 이동 수단을 확인하십시오.

비동기 이벤트가 도착하지 않음

서비스의 Async 메서드는 요청을 스레드에서 실행하고, TThread.Queue를 통해 메인 스레드에서 결과 이벤트 또는 OnError를 발생시킵니다. VCL 또는 FireMonkey 애플리케이션은 메시지 루프에서 이 이벤트를 전달합니다. 콘솔 애플리케이션, Windows 서비스, 스레드에는 이런 루프가 없으므로 응답이 도착할 때까지 **CheckSynchronize**를 호출해야 하며, 그때까지

구성 요소가 살아 있어야 합니다. 이벤트에 전달되는 결과 객체는 이벤트가 반환될 때 해제되므로, 필요한 것은 핸들러 안에서 복사하십시오.
서비스와 타일 소스의 **Timeout**은 Delphi 10.1 Berlin부터 THTTPClient로 적용됩니다. Delphi XE8과 Delphi 10 Seattle에서는 효과가 없습니다.

FireDAC SQLite

- MBTiles 파일, 벡터 MBTiles 파일, TsgcMapOfflineGeocoder의 인덱스는 Delphi XE7부터 FireDAC으로 읽는 SQLite 데이터베이스입니다. Delphi 에디션에는 sgcMaps의 런타임 패키지가 요구하는 SQLite 드라이버가 있는 FireDAC이 포함되어 있어야 합니다.
- 유닛이 FireDAC SQLite 드라이버 링크를 직접 만들고 파일을 읽기 전용으로 열므로, 폼에는 FireDAC 구성 요소가 필요 없습니다. 배포할 때는 Delphi 버전의 FireDAC 문서에 설명된 대로 각 대상 플랫폼에서 FireDAC SQLite 드라이버에 필요한 것을 포함하십시오.
- 파일이 없으면 경로와 함께 "MBTiles file not found" 또는 "Geocoder index not found"가 발생합니다. 파일을 애플리케이션과 함께 배포하고 경로는 런타임에 만드십시오. 상대 경로는 현재 폴더에 따라 달라지기 때문입니다.
- 스키마가 다른 MBTiles 파일은 오류 없이 빈 타일을 만들므로, 새 파일은 먼저 데모에서 확인하십시오.
- 각 TsgcMapOfflineGeocoder에는 연결이 하나 있으며 한 번에 하나의 스레드에서 사용해야 합니다. 스레드마다 자체 구성 요소를 만드십시오. 온라인 지오코더와 달리 인덱스가 없으면 False를 반환하는 대신 예외를 발생시킵니다.

캐시 폴더

- 온라인 레이어는 타일을 **CacheFolder**에 보관합니다. CacheFolder가 비어 있으면 TileSourcePath를 사용하고, 둘 다 비어 있으면 디스크 캐시가 없고 메모리에 256개 타일의 캐시만 있습니다.
- 폴더는 필요할 때 만들어집니다. 폴더에 쓸 수 없으면 타일은 캐시되지 않으며 오류도 발생하지 않습니다. Program Files 아래와 Android 및 iOS의 애플리케이션 폴더에서 이런 일이 생깁니다. 모바일에서는 TPath.GetCachePath 같은 사용자 폴더를 사용하십시오.
- 캐시 파일은 줌, x, y로만 이름이 정해집니다. 레이어, 공급자, Retina 설정마다 각자의 폴더를 지정하십시오. 그렇지 않으면 한 레이어의 타일이 다른 레이어에 그려집니다.
- 캐시된 타일은 만료되지 않습니다. 다시 다운로드하려면 폴더를 삭제하십시오. 예를 들어 공급자에서 스타일이 바뀐 후에 그렇게 합니다.

고해상도 화면

VCL 컨트롤은 타일을 **Options.TileSize** 픽셀(기본값 256)로 그리며, 모니터의 DPI에 맞춰 스스로 크기를 조정하지 않습니다. 고해상도 화면에서는 TileSize를 인치당 픽셀 수(64에서 1024)에 맞춰 설정하고 레이어에 **Retina**를 설정하십시오. 그러면 이를 지원하는 공급자가 512픽셀 타일을 보내어 확대해도 선명하게 유지됩니다. Retina를 사용하는 벡터 레이어는 타일을 512픽셀로 그립니다. 레티나 타일을 지원하는 공급자는 타일, 레이어, 공급자에 나열되어 있습니다.

모바일 기기의 위치

- Android에서는 프로젝트 옵션에서 **Access fine location** 권한을 활성화하고, Active를 True로 설정하기 전에 TsgcMapLocation의 **RequestPermission**을 호출하십시오(Delphi 10.3 이상). 디자인 타임에 Active를 설정하면 사용자가 권한을 허용하기 전에 센서가 시작됩니다.
- iOS와 macOS에서는 프로젝트의 버전 정보에 NSLocationWhenInUseUsageDescription을 추가하십시오. Windows에서는 개인 정보 설정에서 위치 서비스가 활성화되어 있어야 합니다.
- 오류는 "Location sensor not available", "Location sensor access denied or in error state", "Location permission denied"입니다. FireMonkey 데모처럼 센서 없이 테스트하려면 **SimulateLocation** 또는 **LoadTrack**을 사용하십시오.

오프라인 라우팅

- TsgcMapOfflineRouter의 **Open**은 GraphFile이 이 버전의 완전한 .sgcgraph 파일이 아니면 "Invalid routing graph file", "Unsupported routing graph version" 또는 "Truncated routing graph file"을 발생시킵니다. RouteGraphBuilder 도구로 다시 빌드하십시오.
- "Waypoint n is not near a routable road"는 경유지가 그래프의 영역 밖에 있거나 선택한 프로필의 어떤 도로에서도 멀다는 뜻입니다. "No route found between waypoints"는 그래프에서 경유지 사이에 연결이 없다는 뜻이며, 추출 파일의 경계에서 발생합니다.
- 그래프 전체가 메모리에 로드되므로, 사용 영역을 포함하는 가장 작은 추출 파일을 선택하십시오.

- OpenStreetMap 리더는 일반적인 추출 서비스의 형식인 압축되지 않은 .osm.pbf 파일과 zlib으로 압축된 .osm.pbf 파일을 지원합니다. LZMA, LZ4, Zstandard, bzip2로 압축된 파일은 오류를 발생시킵니다.

Lazarus

Lazarus 컨트롤은 타일을 다운로드할 수 없으므로 폴더의 타일만 표시하며, 회전과 불투명도가 없습니다. GeoJSON, GPX, KML은 FPC에서 컴파일되지 않습니다. Lazarus를 참조하십시오.

소개

sgcDashboard는 VCL과 FireMonkey를 지원하는 Delphi 및 C++Builder용 네이티브 대시보드입니다. 대시보드는 타일의 그리드입니다. 각 타일은 드롭된 모든 컨트롤을 호스팅하는 자식 컨트롤이므로, 차트, 지도, PDF 뷰어, 보고서 미리 보기, 메모, 목록을 같은 테마가 적용된 화면에 나란히 배치할 수 있습니다. KPI 카드 컨트롤은 하나의 수치를 추세와 스파크라인과 함께 표시합니다.
sgcDashboard는 eSeGeCe 비주얼 스위트의 일부입니다. 타일은 **sgcCharts**(실시간 데이터 차트), **sgcMaps**(지도), **sgcPDF**(PDF 뷰어), **sgcReports**(보고서 미리 보기)의 뷰어를 호스팅하며, 화면 표시에는 **sgcUICore**를 사용합니다. sgcUICore는 캔버스, 래스터라이저, 글꼴, SVG 작성기, 라이트 및 다크 테마를 갖춘 공유 런타임 패키지입니다. sgcDashboard에는 sgcUICore가 필요하며, 함께 설치됩니다. 다른 제품은 선택 사항입니다. 설치되어 있는 경우에만 타일이 해당 제품을 호스팅합니다.
대시보드의 **Theme** 속성 하나로 대시보드, 타일, 호스팅된 컨트롤을 라이트와 다크 사이에서 전환합니다. Registered 빌드에서는 사용자가 실행 중에 타일을 끌어서 크기를 조정할 수 있고 레이아웃을 JSON으로 저장할 수 있으며, **sgcDashboardWS** 브리지는 sgcWebSockets의 WebSocket, MQTT 또는 서버 전송 이벤트(Server-Sent Events) 연결에서 KPI 카드로 데이터를 공급합니다.

에디션

sgcDashboard는 Registered, Trial, Community 에디션으로 제공됩니다. Community 에디션에는 두 가지 기능이 빠져 있습니다. SaveLayout과 LoadLayout을 이용한 타일의 실행 중 끌기 및 크기 조정, 그리고 sgcDashboardWS 실시간 데이터 브리지입니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 레이아웃 엔진, 타일, sgcUICore 캔버스와 테마, VCL과 FMX.
- 플랫폼: Delphi 버전, C++Builder, 지원 플랫폼.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder: 몇 분 만에 만드는 첫 대시보드.
- 구성 요소: 대시보드, 타일, KPI 카드, 브리지 소스 구성 요소.
- 레이아웃, 테마, 영속성, 실시간 데이터, 헤드리스 KPI 이미지.
- 데모: Delphi 및 C++Builder 데모 프로젝트.

아키텍처

sgcDashboard는 계층으로 구성되어 있습니다. 각 계층은 아래 계층에만 의존하므로, 레이아웃 엔진과 KPI 렌더러는 VCL이나 FireMonkey 없이 컴파일됩니다.

계층	유닛	역할
레이아웃 엔진	sgcDashboard_Layout	TsgcDashboardLayout: 모든 타일의 셀(Col, Row, ColSpan, RowSpan)을 사각형으로 바꾸는 프레임워크 중립 그리드입니다. Columns, RowHeight, Gap, Padding은 장치 독립 픽셀 단위이며, 자동 배치, 충돌 해결, 끌기와 크기 조정을 위한 스냅 계산을 포함합니다. 레이아웃을 참조하십시오.
대시보드와 타일	sgcDashboard_VCL, sgcDashboard_FMX	TsgcVCLDashboard와 TsgcFMXDashboard는 레이아웃 엔진을 소유하고 그 위에 타일을 배치합니다. TsgcVCLDashboardTile과 TsgcFMXDashboardTile은 TPageControl 안의 TTabSheet처럼 자식 컨트롤이며, 타일에 드롭된 컨트롤은 그 타일이 호스팅합니다.
KPI 카드	sgcDashboard_KPI, sgcDashboard_KPI_VCL, sgcDashboard_KPI_FMX	TsgcDashboardKPI는 모델이자 렌더러입니다. TsgcVCLKPICard와 TsgcFMXKPICard는 모델 하나를 소유하고 이를 그리는 얇은 컨트롤입니다.
sgcUICore	sgcUICore_Canvas, sgcUICore_VCL, sgcUICore_FMX, sgcUICore_Theme, sgcUICore_JSON	공유 런타임입니다. 모든 타일과 카드가 그리는 벡터 캔버스와 래스터라이저, 프레임 조절(MaxFPS)과 스레드에 안전한 다시 그리기 요청을 갖춘 VCL 및 FMX 컨트롤 기반 클래스, 라이트 및 다크 테마 토큰과 테마 적용기, 그리고 레이아웃 영속성에 사용되는 JSON 판독기와 작성기가 포함됩니다.
실시간 데이터 브리지	sgcDashboardWS_Source, _WebSocket, _MQTT, _SSE	TsgcDashboardSource와 그 하위 클래스는 sgcWebSockets 클라이언트가 받은 모든 JSON 메시지에서 값을 읽어 KPI 카드에 넣습니다. 브리지는 sgcDashboard와 sgcWebSockets가 필요한 별도의 패키지 sgcDashboardWS입니다.

타일은 자식 컨트롤입니다

타일은 대시보드 안의 실제 컨트롤입니다. 디자인 타임에는 대시보드 구성 요소 편집기의 "Add Tile" 동사가 타일을 만들고, 팔레트에서 모든 컨트롤을 타일에 드롭할 수 있습니다. 디자이너에서 타일을 이동하거나 크기를 조정하면 그리드에 맞춰집니다. 실행 중에는 타일을 코드에서 만들고 **Parent**를 대시보드로 설정합니다. 타일은 테두리, 둥근 모서리, **Title**이 있는 선택적 헤더를 갖춘 카드 표면을 그리며, 클라이언트 영역은 호스팅하는 컨트롤에 맡깁니다.

그리기와 테마

타일과 KPI 카드는 sgcCharts가 사용하는 것과 같은 sgcUICore 캔버스를 통해 그려지므로, 라이트 또는 다크 테마는 두 프레임워크와 헤드리스 PNG 및 SVG 출력에서 동일하게 보입니다. 타일은 호스팅하는 컨트롤을 직접 그리지 않습니다. 컨트롤의 게시된 속성을 통해 테마를 전달합니다. 테마를 참조하십시오.

VCL과 FMX

VCL 클래스와 FMX 클래스는 속성, 메서드, 이벤트가 같습니다. VCL 대시보드는 TScrollingWinControl을, FMX 대시보드는 TVertScrollBox를 상속하므로, 보기보다 큰 그리드는 세로로 스크롤됩니다. 레이아웃의 모든 크기는 장치 독립 픽셀 단위이며, VCL에서는 모니터 PPI로, FMX에서는 폼 배율로 조정됩니다.

참고 항목

- 플랫폼
- 레이아웃
- 테마

에디션

sgcDashboard는 모든 에디션에서 같은 소스로 빌드됩니다. 에디션은 빌드에 컴파일되는 기능만 바꿉니다.

에디션	포함 내용
Registered	모든 기능: VCL 및 FMX 대시보드와 타일, KPI 카드와 헤드리스 KPI 이미지, 테마, SaveLayout과 LoadLayout을 이용한 타일의 실행 중 끌기 및 크기 조정, sgcDashboardWS 실시간 데이터 브리지.
Trial	Registered와 같은 기능입니다. SGC_TRIAL로 컴파일된 빌드는 빌드에 지정된 릴리스 기간이 지나면 체험판임을 알리는 안내를 표시합니다. 그 외에는 달라지는 것이 없습니다.
Community	대시보드와 타일, 자동 배치를 갖춘 그리드 레이아웃, 스파크라인이 있는 KPI 카드와 헤드리스 PNG 및 SVG 출력, 라이트 및 다크 테마. SaveLayout과 LoadLayout을 이용한 실행 중 끌기 및 크기 조정과 sgcDashboardWS 브리지는 빠져 있습니다.

Community에서 빠지는 기능

Community 에디션은 SGC_DASHBOARD_PERSIST를 정의하지 않고 컴파일됩니다. 다음 멤버는 Community 빌드에 존재하지 않습니다.

- 대시보드의 **AllowDrag**, **AllowResize**, **OnLayoutChanged**.
- 대시보드의 **SaveLayout**, **LoadLayout**, **SaveLayoutToFile**, **LoadLayoutFromFile**, **SaveLayoutToStream**, **LoadLayoutFromStream**.
- VCL 타일의 **Locked**. FMX에서 Locked는 표준 FMX 속성이므로 계속 사용할 수 있으며, 막을 끌기가 없을 뿐입니다.

Community 에디션에는 sgcDashboardWS 유닛과 패키지도 포함되지 않으므로 TsgcDashboardSource_WebSocket, TsgcDashboardSource_MQTT, TsgcDashboardSource_SSE를 사용할 수 없습니다. KPI 카드에는 여전히 코드에서 **Value**와 **AddValue**로 데이터를 공급할 수 있습니다.
에디션은 런타임 스위치가 아니라 sgcVer.inc의 컴파일 타임 선택(SGC_COMMUNITY이며 SGC_TRIAL도 함께 끕니다)입니다. 위의 멤버 중 하나를 사용하는 코드는 Community 빌드에서 컴파일되지 않습니다. 문제 해결을 참조하십시오.

참고 항목

- 영속성
- 실시간 데이터
- 플랫폼

설치

sgcDashboard는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcDashboard**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcDashboard에 포함된 항목

- **sgcUICore**는 레이아웃 제품과 sgcCharts의 공통 기반입니다. 자체 확인란은 없습니다. sgcDashboard를 선택하면 설치 프로그램이 자동으로 설치합니다.
- **sgcDashboard for sgcWebSockets**는 **sgcWebSockets bridges** 아래에 표시됩니다. WebSocket, MQTT, SSE 소스(실시간 데이터 참조)가 포함된 sgcDashboardWS 패키지를 설치합니다. 설치 프로그램은 sgcWebSockets가 이미 컴퓨터에 설치되어 있고 라이선스에 sgcDashboard가 포함된 경우에만 이 항목을 제공합니다. 그렇지 않으면 회색으로 표시됩니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드입니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Dashboard`	sgcDashboard 데모입니다. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcDashboard도 포함됩니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 sgcDashboard가 포함되며, sgcCharts, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard도 함께 포함됩니다. sgcWebSockets 브리지는 Community 설치 프로그램에 포함되지 않으며, sgcPDF와 sgcMaps도 Community에 포함되지 않습니다. Community 빌드에서 빠지는 기능은 에디션을 참조하십시오.

이전 독립 실행형 설치

설치 프로그램이 이전 독립 실행형 sgcUICore 설치(예: 독립 실행형 sgcCharts 설치 프로그램이 만든 것)를 발견하면, 그 설치의 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcDashboardD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcDashboard<V>` / `dclsgcDashboard<V>`	sgcDashboard 컴포넌트(런타임/디자인 타임).
`sgcDashboardWS<V>` / `dclsgcDashboardWS<V>`	sgcWebSockets 브리지(런타임/디자인 타임). sgcWebSockets가 필요합니다.

의존 순서대로 빌드합니다. `sgcUICore`, 그다음 `sgcDashboard`, 그다음 `dclsgcDashboard`, 마지막으로 브리지 패키지입니다. 그런 다음 `dcl*` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder의 경우 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcDashboardC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

.NET

.NET에서는 달라지는 것이 없습니다. sgcDashboard는 여전히 자체 NuGet 패키지(`esegece.sgcDashboard`와 그 보조 패키지)로 배포됩니다. .NET 설치를 참조하십시오.

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows(Win32 및 Win64)의 TsgcVCLDashboard, TsgcVCLDashboardTile, TsgcVCLKPICard 이며, 모든 Delphi 버전에서 지원됩니다.
FireMonkey	Delphi XE6부터 TsgcFMXDashboard, TsgcFMXDashboardTile, TsgcFMXKPICard를 사용할 수 있으며(SGC_DASHBOARD_FMX), Win32, Win64, macOS, iOS, Android에서 지원됩니다.
Linux64	프레임워크 중립 유닛, 레이아웃 엔진(sgcDashboard_Layout), 헤드리스 PNG 및 SVG 출력을 갖춘 KPI 모델(sgcDashboard_KPI)이 Delphi 10.2부터 지원됩니다(SGC_DASHBOARD_LINUX). Linux의 FMX 컨트롤에는 FMX for Linux(FMXLinux)가 필요합니다.
실시간 데이터 브리지	sgcDashboardWS 패키지이며, 같은 IDE용 sgcWebSockets 패키지가 설치된 환경에서 사용할 수 있습니다.
.NET	WinForms, WPF, Avalonia용으로 같은 API를 제공하는 C# 포트이며, 헤드리스 PNG 및 SVG 출력과 sgcWebSockets 브리지를 포함합니다. .NET Framework 4.7.2와 .NET 8, 9를 지원합니다. .NET 개요를 참조하십시오.

패키지

sgcDashboard에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcDashboardD13.dpk, C++Builder 13에서는 sgcDashboardC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcDashboard	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, sgcUICore
sgcDashboardVCL	런타임, VCL	sgcDashboard_VCL 및 sgcDashboard_KPI_VCL (TsgcVCLDashboard, TsgcVCLDashboardTile, TsgcVCLKPICard)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcDashboard
sgcDashboardFMX	런타임, FireMonkey, XE6부터	sgcDashboard_FMX 및 sgcDashboard_KPI_FMX (TsgcFMXDashboard, TsgcFMXDashboardTile, TsgcFMXKPICard)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcDashboard
dclsgcDashboard	디자인 타임, VCL	VCL 컴포넌트를 **SGC Dashboard** 팔레트 페이지에 등록하고 대시보드 컴포넌트 편집기에 "Add Tile" 동사를 추가합니다.	designide, rtl, vcl, sgcDashboard, sgcDashboardVCL
dclsgcDashboardFMX	디자인 타임, FireMonkey, XE6부터	FireMonkey 컴포넌트를 **SGC Dashboard** 팔레트 페이지에 등록하	designide, rtl, fmx, sgcDashboard,

		며, 같은 "Add Tile" 동사를 제공합니다.	sgcDashboardFMX, dclsgcDashboard
sgcDashboardWS, dclsgcDashboardWS	런타임 및 디자인 타임, sgcWebSockets 브리지	라이브 데이터 브리지, 프레임워크 독립.	rtl, sgcDashboard, sgcWebSockets

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcDashboardFMX에는 dclsgcDashboard가 필요하므로 dclsgcDashboard를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcDashboard	sgcDashboardVCL	sgcDashboardFMX	dclsgcDashboard, dclsgcDashboardFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcDashboard, sgcDashboardVCL	sgcDashboardFMX	dclsgcDashboard, dclsgcDashboardFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcDashboard만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcDashboard만)

XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcDashboardC13.cbproj, sgcDashboardVCLC13.cbproj, sgcDashboardFMXC13.cbproj, dclsgcDashboardC13.cbproj, dclsgcDashboardFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXDashboard, TsgcFMXDashboardTile 및 TsgcFMXKPICard)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcDashboardVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcDashboard_FMX 및 sgcDashboard_KPI_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcDashboard는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 레이아웃 엔진과, 헤드리스 PNG 및 SVG 출력을 갖춘 KPI 모델입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

PaintTo로 VCL 폼 캡처하기

VCL 폼 또는 그 위의 대시보드를 PaintTo로 비트맵에 캡처하려면 호출 전후로 비트맵 캔버스를 잠그십시오. 잠그지 않은 캔버스에 PaintTo를 하면 캡처 중에 폼의 네이티브 컨트롤이 메시지를 처리할 때 디바이스 컨텍스트를 잃을 수 있으며, 그러면 비트맵이 비어 있거나 일부만 나옵니다. FireMonkey 폼에서는 대신 MakeScreenshot을 사용합니다.

참고 항목

- 에디션
- 아키텍처
- 문제 해결

빠른 시작 VCL

디자인 타임

- **SGC Dashboard** 팔레트 페이지에서 **TsgcVCLDashboard**를 폼에 놓고 **Align**을 alClient로 설정합니다.
- 대시보드를 마우스 오른쪽 버튼으로 클릭하고 **Add Tile**을 선택합니다. 타일마다 반복합니다. 새 타일은 Col과 Row가 -1이므로 자동 배치가 그리드의 첫 번째 빈 슬롯에 놓습니다. 타일에 고정 셀을 지정하려면 오브젝트 인스펙터에서 Col, Row, ColSpan, RowSpan을 설정하거나 디자이너에서 타일을 이동하고 크기를 조정하십시오.
- 타일을 선택하고 그 안에 컨트롤을 드롭합니다. 예를 들어 sgcCharts의 TsgcVCLChart나 TMemo이며, 해당 컨트롤의 **Align**을 alClient로 설정합니다.
- 다른 타일에 **TsgcVCLKPICard**를 드롭하고 Align을 alClient로 설정한 다음 **Title**, **Value**, **Trend**, **TrendValue**를 채웁니다. 카드가 이미 자체 제목을 표시하므로 해당 타일의 **ShowHeader**를 False로 설정합니다.
- 대시보드의 **Theme**을 utDark로 설정합니다. 대시보드, 타일, KPI 카드, 호스팅된 컨트롤이 다크 팔레트로 전환됩니다.

코드에서

같은 대시보드를 실행 중에 만듭니다. 검색 경로에 `sgcUI\Source`가 있으면 이 프로그램은 그대로 컴파일됩니다.

참고 항목

- TsgcVCLDashboard
- TsgcVCLDashboardTile
- TsgcVCLKPICard
- 레이아웃
- 테마

빠른 시작 FMX

FireMonkey 컨트롤에는 Delphi XE6 이상이 필요합니다. VCL 컨트롤과 똑같이 사용합니다. **TsgcFMXDashboard**를 놓고, **Align**을 Client로 설정하고, **Add Tile** 동사를 사용한 다음, 컨트롤이나 **TsgcFMXKPICard**를 타일에 드롭합니다.

코드에서

참고 항목

- TsgcFMXDashboard
- TsgcFMXDashboardTile
- TsgcFMXKPICard
- 플랫폼

빠른 시작 C++Builder

사용 중인 C++Builder 버전용 sgcDashboard 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 경로와 라이브러리 경로에 추가한 다음, Delphi와 똑같이 **SGC Dashboard** 팔레트 페이지의 구성 요소를 사용합니다. **Add Tile** 동사도 같은 방식으로 동작합니다. 설치 프로그램이 두 경로를 IDE에 추가합니다. sgcDashboard와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있으므로, sgcUICore에는 별도의 경로가 필요하지 않습니다.

코드에서

참고 항목

- 빠른 시작 VCL
- 데모
- 플랫폼

데모

데모는 설치 폴더의 **Demos** 폴더에 있으며, Delphi 프로젝트는 Demos\Delphi에, C++Builder용 동일한 데모는 Demos\CBuilder에 있습니다. 모든 데모 폴더에는 데모가 하는 일과 동작 방식을 설명하는 readme.html이 있습니다. 데모는 Win32 및 Win64용 VCL 애플리케이션입니다.

데모	구성 요소	보여 주는 내용
Basic	TsgcVCLDashboard, TsgcVCLDashboardTile, TsgcVCLKPICard	12열 그리드에 타일 여섯 개가 있는 대시보드입니다. 타이머로 움직이는 실시간 KPI 카드 네 개, 이벤트 로그, 서버 목록이 있으며, TMemo와 TListBox에도 테마를 적용하는 라이트 및 다크 테마 전환이 있습니다.
Suite	TsgcVCLDashboard, TsgcVCLKPICard, TsgcVCLChart, TsgcVCLMap, TsgcPDFViewerControl, TsgcReportPreviewVCL	타일이 다른 eSeGeCe VCL 제품을 호스팅하는 대시보드 하나입니다. 실시간 sgcCharts 꺾은선 차트, sgcMaps 지도, sgcPDF 뷰어, sgcReports 미리 보기가 KPI 카드 두 개와 함께 있으며, 모두 같은 테마를 따릅니다. 검색 경로에 sgcCharts, sgcMaps, sgcPDF, sgcReports 소스가 있어야 합니다.
LiveKPI	TsgcVCLKPICard, TsgcDashboardSource_WebSocket, TsgcWebSocketClient, TsgcWebSocketServer	TsgcDashboardSource_WebSocket을 통해 WebSocket 피드에 연결된 KPI 카드 세 개입니다. 데모 안의 TsgcWebSocketServer가 500 ms마다 JSON 메시지를 브로드캐스트하고, 각 소스가 그 안에서 자신의 ValuePath를 읽습니다. sgcWebSockets 소스가 필요하며 Community 에디션에는 포함되지 않습니다.
SaveLayout	TsgcVCLDashboard, TsgcVCLDashboardTile, TsgcVCLKPICard	사용자가 AllowDrag와 AllowResize로 실행 중에 재배치하는 대시보드이며, SaveLayoutToFile과 LoadLayoutFromFile로 layout.json에 저장하고 불러오는 레이아웃, 그리고 OnLayoutChanged를 보여 줍니다. Community 에디션에는 포함되지 않습니다.

참고 항목

- 빠른 시작 VCL
- 영속성
- 실시간 데이터

TsgcVCLDashboard

유닛: sgcDashboard_VCL
TsgcVCLDashboard는 Windows 전용 VCL 대시보드입니다. 자식 TsgcVCLDashboardTile을 **Columns**개의 열과 **RowHeight** 픽셀 높이의 행으로 이루어진 그리드에 배치하며, 타일 사이에는 **Gap**, 타일 주위에는 **Padding**을 두고, 타일이 컨트롤보다 높으면 세로로 스크롤합니다. 타일은 구성 요소 편집기의 **Add Tile** 동사로 디자인 타임에 추가합니다. 셀이 배치되는 방식은 레이아웃을 참조하십시오.

속성

속성	형식	설명
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom입니다. 대시보드 배경을 그리며, ParentTheme이 True인 모든 타일이 이를 따르고, 타일은 이를 호스팅된 컨트롤에 전달합니다. 테마를 참조하십시오.
Columns	Integer	그리드 열의 개수입니다. 기본값은 12입니다.
RowHeight	Integer	그리드 행 하나의 높이이며, 장치 독립 픽셀 단위입니다. 기본값은 120입니다.
Gap	Integer	타일 사이의 간격이며, 장치 독립 픽셀 단위입니다. 기본값은 12입니다.
Padding	Integer	대시보드의 테두리와 타일 사이의 간격이며, 장치 독립 픽셀 단위입니다. 기본값은 12입니다.
AutoFlow	Boolean	True(기본값)이면 Col 또는 Row가 -1인 타일이 그리드의 첫 번째 빈 슬롯을 채웁니다. False이면 -1은 0으로 읽습니다.
AllowDrag	Boolean	타일의 헤더를 끌어서 실행 중에 이동합니다. 기본값은 False입니다. Community 에디션에는 없습니다.
AllowResize	Boolean	타일 오른쪽 아래 모서리의 그립을 끌어서 실행 중에 크기를 조정합니다. 기본값은 False입니다. Community 에디션에는 없습니다.
Layout	TsgcDashboardLayout (public, 읽기 전용)	대시보드의 그리드 레이아웃 엔진입니다. 레이아웃을 참조하십시오.

일반적인 TScrollingWinControl 멤버도 게시되어 있습니다. **Align**, **Anchors**, **Constraints**, **Enabled**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **TabOrder**, **TabStop**, **Visible**, **OnClick**, **OnDblClick**, **OnMouseDown**, **OnMouseMove**, **OnMouseUp**, **OnResize**입니다.

메서드

메서드	반환 값	설명
Arrange		지금 타일을 그리드에 배치합니다. 대시보드는 타일이나 그리드 속성이 바뀔 때 스스로 배치하므로 호출할 필요는 거의 없습니다.
ApplyTheme		각 타일의 테마를 타일이 호스팅하는 컨트롤에 다시 적용합니다. 예를 들어 코드에서 컨트롤을 추가하거나 변경한 후에 사용합니다.
SaveLayout	string	그리드 메트릭과, 이름이 있는 모든 타일의 셀 및 표시 여부를 JSON으로 반환합니다. Community 에디션에는 없습니다.

LoadLayout(const aJSON: string)		SaveLayout 텍스트를 적용합니다. 타일은 Name으로 대응되며, 알 수 없는 이름은 무시되고 텍스트에 없는 타일은 기존 셀을 유지합니다. 배치를 한 번 수행한 뒤 OnLayoutChanged가 발생합니다. Community 에디션에는 없습니다.
SaveLayoutToFile(const aFileName: string)		SaveLayout을 파일에 기록합니다. BOM 없는 UTF-8입니다. Community 에디션에는 없습니다.
LoadLayoutFromFile(const aFileName: string)		SaveLayoutToFile로 기록한 레이아웃 파일을 읽습니다. BOM은 건너뜁니다. Community 에디션에는 없습니다.
SaveLayoutToStream(aStream: TStream)		SaveLayout을 스트림에 기록합니다. BOM 없는 UTF-8입니다. Community 에디션에는 없습니다.
LoadLayoutFromStream(aStream: TStream)		스트림에서 레이아웃을 읽습니다. BOM은 건너뜁니다. Community 에디션에는 없습니다.

이벤트

이벤트	형식	설명
OnLayoutChanged	TNotifyEvent	실행 중의 끌기 또는 크기 조정으로 타일의 셀이 바뀌었거나 LoadLayout이 적용되었습니다. Community 에디션에는 없습니다.

예제

참고 항목

- TsgcVCLDashboardTile
- TsgcFMXDashboard
- 레이아웃
- 영속성
- 테마

TsgcVCLDashboardTile

유닛: sgcDashboard_VCL
TsgcVCLDashboardTile은 TsgcVCLDashboard의 타일 하나입니다. 테두리, 둥근 모서리, 선택적 헤더가 있는 테마 적용 카드를 그리며, 그 안에 드롭된 모든 VCL 컨트롤을 호스팅합니다. 차트, 지도, PDF 뷰어, 보고서 미리 보기, KPI 카드, 표준 컨트롤이 해당됩니다. 타일은 자신에게 적용된 유효 테마를 호스팅하는 컨트롤에 전달합니다. 호스팅된 컨트롤은 타일이 직접 그리지 않습니다.

속성

속성	형식	설명
Title	string	타일 헤더의 텍스트입니다.
ShowHeader	Boolean	Title이 표시되는 헤더 띠를 그립니다. 기본값은 True입니다. 타일의 클라이언트 영역은 헤더 아래에서 시작합니다.
Col	Integer	타일의 0부터 시작하는 그리드 열입니다. 기본값은 -1이며 자동 배치로 위치가 정해집니다.
Row	Integer	타일의 0부터 시작하는 그리드 행입니다. 기본값은 -1이며 자동 배치로 위치가 정해집니다.
ColSpan	Integer	타일이 차지하는 열의 개수입니다. 기본값은 3이며 1에서 Columns 사이로 제한됩니다.
RowSpan	Integer	타일이 차지하는 행의 개수입니다. 기본값은 2이며 최소 1입니다.
ParentTheme	Boolean	타일과 그 안에 호스팅된 컨트롤이 대시보드의 Theme을 따릅니다. 기본값은 True입니다.
Theme	TsgcUITheme	타일의 테마입니다. 설정하면 ParentTheme이 해제됩니다. 대시보드 밖에서는 ParentTheme이 있는 타일이 utLight를 사용합니다.
Locked	Boolean	대시보드가 허용하더라도 타일은 실행 중에 이동하거나 크기가 조정되지 않습니다. 기본값은 False입니다. Community 에디션에는 없습니다.
Dashboard	TsgcVCLDashboard (public, 읽기 전용)	타일이 속한 대시보드입니다. 부모가 대시보드가 아니면 nil입니다.

일반적인 컨트롤 멤버도 게시되어 있습니다. **Constraints**, **Enabled**, **Hint**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **Visible**, **OnClick**, **OnContextPopup**, **OnDblClick**, **OnMouseDown**, **OnMouseMove**, **OnMouseUp**, **OnResize**입니다. 숨겨진 타일(Visible이 False)은 자신의 셀을 비워 둡니다.

메서드

메서드	반환값	설명
GetCell	TsgcDashboardCell	타일이 요청한 셀이며, Col, Row, ColSpan, RowSpan을 담은 레코드입니다.
SetCell(const aCell: TsgcDashboardCell)		Col, Row, ColSpan, RowSpan을 한 번에 설정하며, 대시보드의 배치는 한 번만 수행합니다.

이벤트

이벤트	형식	설명
OnResize	TNotifyEvent	대시보드의 배치 또는 실행 중의 크기 조정으로 타일의 크기가 조정되었습니다.

예제

타일은 자체 테마를 설정하지 않는 한 대시보드 테마를 따릅니다.

참고 항목

- TsgcVCLDashboard
- TsgcFMXDashboardTile
- 레이아웃
- 테마

TsgcFMXDashboard

유닛: sgcDashboard_FMX
TsgcFMXDashboard는 FireMonkey 대시보드이며 Delphi XE6부터 사용할 수 있습니다. TsgcVCLDashboard와 같은 그리드, 테마, 영속성 멤버를 갖추고 있습니다. TVertScrollBox를 상속하므로 타일이 세로로 스크롤되며, **Padding**은 FMX 컨트롤의 TBounds Padding을 대체하는, 장치 독립 픽셀 단위의 Integer입니다. 구성 요소 편집기의 **Add Tile** 동사는 Windows IDE에서 사용할 수 있습니다.

속성

속성	형식	설명
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom입니다. 대시보드 배경을 그리며, ParentTheme이 True인 모든 타일이 이를 따르고, 타일은 이를 호스팅된 컨트롤에 전달합니다. 테마를 참조하십시오.
Columns	Integer	그리드 열의 개수입니다. 기본값은 12입니다.
RowHeight	Integer	그리드 행 하나의 높이이며, 장치 독립 픽셀 단위입니다. 기본값은 120입니다.
Gap	Integer	타일 사이의 간격이며, 장치 독립 픽셀 단위입니다. 기본값은 12입니다.
Padding	Integer	대시보드의 테두리와 타일 사이의 간격이며, 장치 독립 픽셀 단위입니다. 기본값은 12입니다.
AutoFlow	Boolean	True(기본값)이면 Col 또는 Row가 -1인 타일이 그리드의 첫 번째 빈 슬롯을 채웁니다. False이면 -1은 0으로 읽습니다.
AllowDrag	Boolean	타일의 헤더를 끌어서 실행 중에 이동합니다. 기본값은 False입니다. Community 에디션에는 없습니다.
AllowResize	Boolean	타일 오른쪽 아래 모서리의 그립을 끌어서 실행 중에 크기를 조정합니다. 기본값은 False입니다. Community 에디션에는 없습니다.
Layout	TsgcDashboardLayout (public, 읽기 전용)	대시보드의 그리드 레이아웃 엔진입니다. 레이아웃을 참조하십시오.

메서드

메서드	반환 값	설명
Arrange		지금 타일을 그리드에 배치합니다. 대시보드는 타일이나 그리드 속성이 바뀔 때 스스로 배치하므로 호출할 필요는 거의 없습니다.
ApplyTheme		각 타일의 테마를 타일이 호스팅하는 컨트롤에 다시 적용합니다. 예를 들어 코드에서 컨트롤을 추가하거나 변경한 후에 사용합니다.
SaveLayout	string	그리드 메트릭과, 이름이 있는 모든 타일의 셀 및 표시 여부를 JSON으로 반환합니다. Community 에디션에는 없습니다.
LoadLayout(const aJSON: string)		SaveLayout 텍스트를 적용합니다. 타일은 Name으로 대응되며, 알 수 없는 이름은 무시되고 텍스트에 없는 타일은 기존 셀을 유지합니

		다. 배치를 한 번 수행한 뒤 OnLayoutChanged가 발생합니다. Community 에디션에는 없습니다.
SaveLayoutToFile(const aFileName: string)		SaveLayout을 파일에 기록합니다. BOM 없는 UTF-8입니다. Community 에디션에는 없습니다.
LoadLayoutFromFile(const aFileName: string)		SaveLayoutToFile로 기록한 레이아웃 파일을 읽습니다. BOM은 건너뜁니다. Community 에디션에는 없습니다.
SaveLayoutToStream(aStream: TStream)		SaveLayout을 스트림에 기록합니다. BOM 없는 UTF-8입니다. Community 에디션에는 없습니다.
LoadLayoutFromStream(aStream: TStream)		스트림에서 레이아웃을 읽습니다. BOM은 건너뜁니다. Community 에디션에는 없습니다.

이벤트

이벤트	형식	설명
OnLayoutChanged	TNotifyEvent	실행 중의 끌기 또는 크기 조정으로 타일의 셀이 바뀌었거나 LoadLayout이 적용되었습니다. Community 에디션에는 없습니다.

예제

참고 항목

- TsgcFMXDashboardTile
- TsgcVCLDashboard
- 빠른 시작 FMX

TsgcFMXDashboardTile

유닛: sgcDashboard_FMX
TsgcFMXDashboardTile은 TsgcFMXDashboard의 타일 하나이며, TsgcVCLDashboardTile과 같은 멤버를 갖고 있습니다. 표준 FMX **Locked** 속성이 True인 타일은 실행 중의 끌기나 크기 조정을 시작하지 않습니다.

속성

속성	형식	설명
Title	string	타일 헤더의 텍스트입니다.
ShowHeader	Boolean	Title이 표시되는 헤더 띠를 그립니다. 기본값은 True입니다. 타일의 클라이언트 영역은 헤더 아래에서 시작합니다.
Col	Integer	타일의 0부터 시작하는 그리드 열입니다. 기본값은 -1이며 자동 배치로 위치가 정해집니다.
Row	Integer	타일의 0부터 시작하는 그리드 행입니다. 기본값은 -1이며 자동 배치로 위치가 정해집니다.
ColSpan	Integer	타일이 차지하는 열의 개수입니다. 기본값은 3이며 1에서 Columns 사이로 제한됩니다.
RowSpan	Integer	타일이 차지하는 행의 개수입니다. 기본값은 2이며 최소 1입니다.
ParentTheme	Boolean	타일과 그 안에 호스팅된 컨트롤이 대시보드의 Theme을 따릅니다. 기본값은 True입니다.
Theme	TsgcUITheme	타일의 테마입니다. 설정하면 ParentTheme이 해제됩니다. 대시보드 밖에서는 ParentTheme이 있는 타일이 utLight를 사용합니다.
Locked	Boolean	표준 FMX 속성입니다. 잠긴 타일은 실행 중에 이동하거나 크기를 조정할 수 없습니다. 기본값은 False입니다.
Dashboard	TsgcFMXDashboard (public, 읽기 전용)	타일이 속한 대시보드입니다. 부모가 대시보드가 아니면 nil입니다.

일반적인 FMX 컨트롤 멤버도 게시되어 있으며, **Anchors**, **Enabled**, **Hint**, **Margins**, **Opacity**, **PopupMenu**, **Visible**, 그리고 마우스, 그리기, 크기 조정 이벤트가 포함됩니다.

메서드

메서드	반환값	설명
GetCell	TsgcDashboardCell	타일이 요청한 셀이며, Col, Row, ColSpan, RowSpan을 담은 레코드입니다.
SetCell(const aCell: TsgcDashboardCell)		Col, Row, ColSpan, RowSpan을 한 번에 설정하며, 대시보드의 배치는 한 번만 수행합니다.

이벤트

이벤트	형식	설명
OnResize	TNotifyEvent	타일의 크기가 조정되었습니다.
OnResized	TNotifyEvent	타일의 크기 조정이 완료되었습니다.

예제

참고 항목

- TsgcFMXDashboard
- TsgcVCLDashboardTile
- 테마

TsgcDashboardKPI

유닛: sgcDashboard_KPI
TsgcDashboardKPI는 KPI 카드의 프레임워크 중립 모델이자 렌더러입니다. 제목, 큰 수치, 선택적 아이콘, 텍스트가 있는 추세 화살표, 바닥글 텍스트, 마지막 MaxPoints개 값의 스파크라인으로 구성됩니다. **Render**는 임의의 sgcUICore 캔버스에 카드를 그리므로, 같은 코드가 TsgcVCLKPICard, TsgcFMXKPICard, 그리고 헤드리스 PNG 및 SVG 이미지를 그립니다. Linux64를 포함한 모든 플랫폼에서 빌드됩니다.
AddValue와 ClearValues는 어느 스레드에서든 호출할 수 있으며, 값은 잠금으로 보호되고 OnChange는 그 스레드에서 호출됩니다. 그 밖의 모든 속성은 메인 스레드에서 설정하도록 되어 있습니다.

속성

속성	형식	설명
Title	string	카드 상단의 캡션입니다.
Value	string	카드의 큰 수치이며, 이미 서식이 지정된 상태입니다.
Icon	string	강조 원 안에 그려지는 짧은 텍스트 글리프입니다. 예를 들어 '$' 또는 '%'입니다. 비워 두면 아이콘을 그리지 않습니다.
Trend	TsgcDashboardKPITrend	ktNone(기본값), ktUp, ktDown 또는 ktNeutral입니다. 추세 화살표를 그리며, 상승은 녹색, 하락은 빨간색입니다.
TrendValue	string	추세 화살표 옆의 텍스트입니다. 예를 들어 '+4.2 %'입니다.
FooterText	string	카드 하단의 작은 텍스트입니다.
AccentColor	TsgcUIColor	아이콘 원과 스파크라인의 $AARRGGBB 색상입니다. 0(기본값)이면 테마의 강조 색상을 사용합니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark입니다. 대시보드 타일 안에서는 타일이 이를 설정합니다.
MaxPoints	Integer	스파크라인이 유지하는 값의 개수입니다. 기본값은 50입니다. 새 값이 도착하면 가장 오래된 값이 버려집니다.
ValueCount	Integer (public, 읽기 전용)	스파크라인 값의 개수입니다.
Values[Index: Integer]	Double (public, 읽기 전용)	스파크라인 값 하나이며, 가장 오래된 값이 먼저입니다.

메서드

메서드	반환값	설명
AddValue(aValue: Double)		스파크라인에 값을 추가합니다. 스레드에 안전합니다. NaN과 무한대 값은 무시됩니다.
ClearValues		스파크라인의 모든 값을 제거합니다.
GetValueArray	TsgcDashboardKPIValues	스파크라인 값의 복사본이며, 가장 오래된 값이 먼저입니다.

Render(aCanvas: TsgcUICanvas; aWidth, aHeight: Integer; aScale: Double = 1)		카드를 aCanvas에 aWidth x aHeight 장치 픽셀로 그립니다. aScale은 모든 크기에 적용되는 HiDPI 배율입니다.
SaveToPNG(const aFileName: string; aWidth, aHeight: Integer; aScale: Double = 1)		카드를 PNG 파일로 렌더링합니다. aWidth와 aHeight는 논리 픽셀이며, 이미지는 Round(aWidth * aScale) x Round(aHeight * aScale) 픽셀입니다. 오버로드 하나는 TStream에 기록합니다.
SaveToSVG(const aFileName: string; aWidth, aHeight: Integer; aScale: Double = 1)		카드를 SVG 파일로 렌더링합니다. 오버로드 하나는 TStream에 기록합니다.
Assign(Source: TPersistent)		다른 TsgcDashboardKPI의 속성과 스파크라인 값을 복사합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	속성 또는 스파크라인이 변경되었습니다. 변경을 수행한 스레드에서 호출됩니다.

예제

참고 항목

- 헤드리스 KPI 이미지
- TsgcVCLKPICard
- TsgcFMXKPICard

TsgcVCLKPICard

유닛: sgcDashboard_KPI_VCL
TsgcVCLKPICard는 Windows 전용 VCL KPI 카드입니다. 이 컨트롤은 TsgcDashboardKPI를 소유하고, 그 속성을 게시하며, sgcUICore VCL 컨트롤 기반 클래스를 통해 그립니다. 변경은 새 프레임을 예약할 뿐이므로 AddValue는 어느 스레드에서든 호출할 수 있고, 카드는 메인 스레드에서 초당 최대 MaxFPS번 다시 그려집니다.

속성

속성	형식	설명
Title	string	카드 상단의 캡션입니다.
Value	string	카드의 큰 수치이며, 이미 서식이 지정된 상태입니다.
Icon	string	강조 원 안에 그려지는 짧은 텍스트 글리프입니다. 예를 들어 '$' 또는 '%'입니다. 비워 두면 아이콘을 그리지 않습니다.
Trend	TsgcDashboardKPITrend	ktNone(기본값), ktUp, ktDown 또는 ktNeutral입니다. 추세 화살표를 그리며, 상승은 녹색, 하락은 빨간색입니다.
TrendValue	string	추세 화살표 옆의 텍스트입니다. 예를 들어 '+4.2 %'입니다.
FooterText	string	카드 하단의 작은 텍스트입니다.
AccentColor	TsgcUIColor	아이콘 원과 스파크라인의 $AARRGGBB 색상입니다. 0(기본값)이면 테마의 강조 색상을 사용합니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark입니다. 대시보드 타일 안에서는 타일이 이를 설정합니다.
MaxPoints	Integer	스파크라인이 유지하는 값의 개수입니다. 기본값은 50입니다. 새 값이 도착하면 가장 오래된 값이 버려집니다.
KPI	TsgcDashboardKPI (public, 읽기 전용)	카드 모델이며, 원하는 크기로 헤드리스 PNG 및 SVG 출력에도 사용할 수 있습니다.
MaxFPS	Integer (public)	컨트롤의 초당 최대 프레임 수이며, sgcUICore 컨트롤 기반 클래스에서 상속됩니다.
FrameCount	Cardinal (public, 읽기 전용)	컨트롤이 생성된 이후 렌더링된 프레임의 수입니다.

일반적인 컨트롤 멤버도 게시되어 있습니다. **Align**, **Anchors**, **Color**, **Constraints**, **Enabled**, **ParentColor**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **Visible**, 그리고 클릭, 마우스, 크기 조정 이벤트입니다.

메서드

메서드	반환값	설명
AddValue(aValue: Double)		스파크라인에 값을 추가합니다. 스레드에 안전합니다. NaN과 무한대 값은 무시됩니다.
ClearValues		스파크라인의 모든 값을 제거합니다.

ContentChanged		새 프레임을 예약합니다. 스레드에 안전합니다. 카드는 변경이 있을 때마다 이를 스스로 호출합니다.

이벤트

이벤트	형식	설명
OnClick	TNotifyEvent	카드를 클릭했습니다.
OnResize	TNotifyEvent	카드의 크기가 조정되었습니다.

예제

카드를 새 측정값으로 갱신하는 방법입니다.

참고 항목

- TsgcDashboardKPI
- TsgcFMXKPICard
- 실시간 데이터

TsgcFMXKPICard

유닛: sgcDashboard_KPI_FMX
TsgcFMXKPICard는 Delphi XE6부터 사용할 수 있는 FireMonkey KPI 카드입니다. TsgcVCLKPICard와 같은 멤버를 갖추고 있으며, 같은 TsgcDashboardKPI 모델을 그립니다.

속성

속성	형식	설명
Title	string	카드 상단의 캡션입니다.
Value	string	카드의 큰 수치이며, 이미 서식이 지정된 상태입니다.
Icon	string	강조 원 안에 그려지는 짧은 텍스트 글리프입니다. 예를 들어 '$' 또는 '%'입니다. 비워 두면 아이콘을 그리지 않습니다.
Trend	TsgcDashboardKPITrend	ktNone(기본값), ktUp, ktDown 또는 ktNeutral입니다. 추세 화살표를 그리며, 상승은 녹색, 하락은 빨간색입니다.
TrendValue	string	추세 화살표 옆의 텍스트입니다. 예를 들어 '+4.2 %'입니다.
FooterText	string	카드 하단의 작은 텍스트입니다.
AccentColor	TsgcUIColor	아이콘 원과 스파크라인의 $AARRGGBB 색상입니다. 0(기본값)이면 테마의 강조 색상을 사용합니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark입니다. 대시보드 타일 안에서는 타일이 이를 설정합니다.
MaxPoints	Integer	스파크라인이 유지하는 값의 개수입니다. 기본값은 50입니다. 새 값이 도착하면 가장 오래된 값이 버려집니다.
KPI	TsgcDashboardKPI (public, 읽기 전용)	카드 모델이며, 헤드리스 PNG 및 SVG 출력에도 사용할 수 있습니다.
MaxFPS	Integer (public)	컨트롤의 초당 최대 프레임 수이며, sgcUICore 컨트롤 기반 클래스에서 상속됩니다.

일반적인 FMX 컨트롤 멤버도 게시되어 있으며, **Align**, **Anchors**, **Margins**, **Opacity**, **RotationAngle**, **Scale**, **Visible**, 그리고 마우스, 그리기, 크기 조정 이벤트가 포함됩니다.

메서드

메서드	반환값	설명
AddValue(aValue: Double)		스파크라인에 값을 추가합니다. 스레드에 안전합니다. NaN과 무한대 값은 무시됩니다.
ClearValues		스파크라인의 모든 값을 제거합니다.

이벤트

이벤트	형식	설명
OnClick	TNotifyEvent	카드를 클릭했습니다.
OnResize	TNotifyEvent	카드의 크기가 조정되었습니다.

예제

참고 항목

- TsgcDashboardKPI
- TsgcVCLKPICard
- 빠른 시작 FMX

TsgcDashboardSource

유닛: sgcDashboardWS_Source
TsgcDashboardSource는 KPI 카드에 데이터를 공급하는 sgcWebSockets 브리지 소스의 기반 클래스입니다. TsgcDashboardSource_WebSocket, TsgcDashboardSource_MQTT, TsgcDashboardSource_SSE가 이를 상속합니다. 하위 클래스가 받는 모든 메시지는 JSON으로 해석됩니다. **ValuePath**에 있는 숫자가 카드의 스파크라인에 추가되고 카드의 Value가 되며, **ValueFormat**으로 서식이 지정됩니다. 카드의 Trend는 같은 소스의 이전 값에서 도출됩니다. 더 높으면 ktUp, 더 낮으면 ktDown, 같으면 ktNeutral입니다. 이 브리지는 Community 에디션에 포함되지 않습니다. sgcDashboardWS 패키지와 sgcWebSockets가 필요합니다.
스레딩은 실시간 데이터에서 설명합니다.

속성

속성	형식	설명
Active	Boolean	False이면 소스가 모든 메시지를 무시하고 클라이언트의 이전 핸들러를 복원합니다. 기본값은 True입니다.
Card	TComponent	데이터를 공급할 KPI 카드입니다. TsgcVCLKPICard 또는 TsgcFMXKPICard이며, IsgcDashboardKPIHost를 구현하는 모든 구성 요소를 사용할 수 있습니다. 그 밖의 구성 요소는 거부됩니다.
ValuePath	string	모든 메시지에서 숫자를 가리키는 JSON 경로입니다. 예를 들어 'price' 또는 'data.items[0].value'입니다. 숫자 또는 숫자 문자열을 사용할 수 있으며, 로캘과 관계없이 소수점 구분 기호는 '.'입니다. 비워 두면 메시지 전체를 숫자로 읽습니다.
ValueFormat	string	카드의 Value 텍스트에 사용하는 서식 문자열이며, 숫자와 함께 Format에 전달됩니다. 기본값은 '%.2f'입니다. 출력에는 애플리케이션 로캘의 소수점 구분 기호가 사용됩니다.
TitlePath	string	설정하면 해당 JSON 값을 카드의 Title로 복사합니다.
TrendValuePath	string	설정하면 해당 JSON 값을 카드의 TrendValue로 복사합니다.

이벤트

이벤트	형식	설명
OnValue	TsgcDashboardSourceValueEvent	procedure(Sender: TObject; aValue: Double). 값을 읽어 카드를 갱신했습니다. Delphi 2010 이상에서는 메인 스레드에서 발생합니다.

예제

참고 항목

- 실시간 데이터
- TsgcVCLKPICard

TsgcDashboardSource_WebSocket

유닛: sgcDashboardWS_Source_WebSocket
TsgcDashboardSource_WebSocket은 TsgcWebSocketClient가 받은 텍스트 메시지로 KPI 카드에 데이터를 공급합니다. 소스가 Active이고 Client가 할당된 동안에는(실행 중에만), 소스가 클라이언트에 자체 OnMessage 핸들러를 설치하고 이전에 있던 핸들러를 유지합니다. 모든 텍스트 메시지는 먼저 카드로 읽혀 들어간 다음 이전 핸들러로 전달되므로 애플리케이션 핸들러도 그대로 실행됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 소멸시키면, 그 사이에 애플리케이션이 핸들러를 교체하지 않은 한 이전 핸들러가 복원됩니다. 이 브리지는 Community 에디션에 포함되지 않습니다. sgcDashboardWS 패키지와 sgcWebSockets가 필요합니다.

속성

속성	형식	설명
Client	TsgcWebSocketClient	텍스트 메시지를 읽어 올 WebSocket 클라이언트입니다.
Active	Boolean	False이면 소스가 모든 메시지를 무시하고 클라이언트의 이전 핸들러를 복원합니다. 기본값은 True입니다.
Card	TComponent	데이터를 공급할 KPI 카드입니다. TsgcVCLKPICard 또는 TsgcFMXKPICard이며, IsgcDashboardKPIHost를 구현하는 모든 구성 요소를 사용할 수 있습니다. 그 밖의 구성 요소는 거부됩니다.
ValuePath	string	모든 메시지에서 숫자를 가리키는 JSON 경로입니다. 예를 들어 'price' 또는 'data.items[0].value'입니다. 숫자 또는 숫자 문자열을 사용할 수 있으며, 로캘과 관계없이 소수점 구분 기호는 '.'입니다. 비워 두면 메시지 전체를 숫자로 읽습니다.
ValueFormat	string	카드의 Value 텍스트에 사용하는 서식 문자열이며, 숫자와 함께 Format에 전달됩니다. 기본값은 '%.2f'입니다. 출력에는 애플리케이션 로캘의 소수점 구분 기호가 사용됩니다.
TitlePath	string	설정하면 해당 JSON 값을 카드의 Title로 복사합니다.
TrendValuePath	string	설정하면 해당 JSON 값을 카드의 TrendValue로 복사합니다.

메서드

메서드	반환값	설명
ProcessMessage(const aText: string)	Boolean	메시지 하나를 Client에서 온 것처럼 읽습니다. 올바른 JSON이 아니거나 ValuePath가 숫자가 아니면 False입니다. 테스트나 다른 경로로 도착하는 메시지에 유용합니다.

이벤트

이벤트	형식	설명
OnValue	TsgcDashboardSourceValueEvent	procedure(Sender: TObject; aValue: Double). 값을 읽어 카드를 갱신했습니다. Delphi 2010 이상에서는 메인 스레드에서 발생합니다.

예제

참고 항목

- TsgcDashboardSource
- TsgcDashboardSource_MQTT
- 실시간 데이터

TsgcDashboardSource_MQTT

유닛: sgcDashboardWS_Source_MQTT
TsgcDashboardSource_MQTT는 TsgcWSPClient_MQTT가 받은 publish 메시지로 KPI 카드에 데이터를 공급합니다. 소스가 Active이고 Client가 할당된 동안에는(실행 중에만), 소스가 클라이언트에 자체 OnMQTTPublish 핸들러를 설치하고 이전에 있던 핸들러를 유지합니다. 모든 publish는 먼저 카드로 읽혀 들어간 다음 이전 핸들러로 전달되므로 애플리케이션 핸들러도 그대로 실행됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 소멸시키면, 그 사이에 애플리케이션이 핸들러를 교체하지 않은 한 이전 핸들러가 복원됩니다. 소스를 연결한 뒤에 Client.OnMQTTPublish를 할당하는 애플리케이션은, 둘 다 유지하려면 Client를 설정하기 전에 할당하거나 Active를 전환해야 합니다.
Topic은 읽을 publish를 선택하며, MQTT 와일드카드 + (한 수준)와 # (해당 수준과 모든 하위 수준)를 사용할 수 있습니다. Topic이 비어 있으면 모든 publish를 읽습니다. 소스는 구독하지 않으며, 구독은 애플리케이션이 MQTT 클라이언트를 통해 수행합니다. 단순 숫자 페이로드는 ValuePath를 비워 두고 읽습니다. 이 브리지는 Community 에디션에 포함되지 않습니다. sgcDashboardWS 패키지와 sgcWebSockets가 필요합니다.

속성

속성	형식	설명
Client	TsgcWSPClient_MQTT	publish를 읽어 올 MQTT 클라이언트입니다.
Topic	string	읽을 publish의 MQTT 토픽 필터입니다. 비어 있으면 모든 publish를 읽습니다.
Active	Boolean	False이면 소스가 모든 메시지를 무시하고 클라이언트의 이전 핸들러를 복원합니다. 기본값은 True입니다.
Card	TComponent	데이터를 공급할 KPI 카드입니다. TsgcVCLKPICard 또는 TsgcFMXKPICard이며, IsgcDashboardKPIHost를 구현하는 모든 구성 요소를 사용할 수 있습니다. 그 밖의 구성 요소는 거부됩니다.
ValuePath	string	모든 메시지에서 숫자를 가리키는 JSON 경로입니다. 예를 들어 'price' 또는 'data.items[0].value'입니다. 숫자 또는 숫자 문자열을 사용할 수 있으며, 로캘과 관계없이 소수점 구분 기호는 '.'입니다. 비워 두면 메시지 전체를 숫자로 읽습니다.
ValueFormat	string	카드의 Value 텍스트에 사용하는 서식 문자열이며, 숫자와 함께 Format에 전달됩니다. 기본값은 '%.2f'입니다. 출력에는 애플리케이션 로캘의 소수점 구분 기호가 사용됩니다.
TitlePath	string	설정하면 해당 JSON 값을 카드의 Title로 복사합니다.
TrendValuePath	string	설정하면 해당 JSON 값을 카드의 TrendValue로 복사합니다.

메서드

메서드	반환값	설명
ProcessPublish(const aTopic, aText: string)	Boolean	publish 하나를 Client에서 온 것처럼 읽습니다. 토픽이 Topic과 일치하지 않거나 페이로드를 읽을 수 없으면 False입니다.

이벤트

이벤트	형식	설명

OnValue	TsgcDashboardSourceValueEvent	procedure(Sender: TObject; aValue: Double). 값을 읽어 카드를 갱신했습니다. Delphi 2010 이상에서는 메인 스레드에서 발생합니다.

예제

참고 항목

- TsgcDashboardSource
- TsgcDashboardSource_WebSocket
- 실시간 데이터

TsgcDashboardSource_SSE

유닛: sgcDashboardWS_Source_SSE
TsgcDashboardSource_SSE는 TsgcHTTP1Client가 받은 서버 전송 이벤트(Server-Sent Events)로 KPI 카드에 데이터를 공급합니다. 애플리케이션은 **Client.GetSSE(URL)**로 스트림을 시작합니다. 이 호출은 스트림이 끝날 때까지 차단되므로 보통 작업자 스레드에서 호출합니다. 소스가 Active이고 Client가 할당된 동안에는(실행 중에만), 소스가 클라이언트에 자체 OnSSEMessage 핸들러를 설치하고 이전에 있던 핸들러를 유지합니다. 모든 청크는 먼저 카드로 읽혀 들어간 다음 이전 핸들러로 전달되므로 애플리케이션 핸들러도 그대로 실행됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 소멸시키면, 그 사이에 애플리케이션이 핸들러를 교체하지 않은 한 이전 핸들러가 복원됩니다. 애플리케이션 핸들러는 여전히 스트림을 취소할 수 있습니다.
OnSSEMessage는 스트림의 원시 바이트를 도착하는 대로 전달하므로, 청크 하나에 이벤트가 여러 개 들어 있거나 이벤트의 일부만 들어 있을 수 있습니다. 소스는 청크를 버퍼링하고, 모든 빈 줄(LF, CRLF 또는 CR 줄 끝)에서 이벤트로 나누며, 이벤트의 data: 줄을 LF로 결합하고, 주석과 다른 필드는 무시한 뒤, 모든 이벤트의 데이터를 ValuePath로 읽습니다. 데이터가 없는 이벤트는 무시됩니다. 이 브리지는 Community 에디션에 포함되지 않습니다. sgcDashboardWS 패키지와 sgcWebSockets가 필요합니다.

속성

속성	형식	설명
Client	TsgcHTTP1Client	서버 전송 이벤트를 읽어 올 HTTP 클라이언트입니다.
Active	Boolean	False이면 소스가 모든 메시지를 무시하고 클라이언트의 이전 핸들러를 복원합니다. 기본값은 True입니다.
Card	TComponent	데이터를 공급할 KPI 카드입니다. TsgcVCLKPICard 또는 TsgcFMXKPICard이며, IsgcDashboardKPIHost를 구현하는 모든 구성 요소를 사용할 수 있습니다. 그 밖의 구성 요소는 거부됩니다.
ValuePath	string	모든 메시지에서 숫자를 가리키는 JSON 경로입니다. 예를 들어 'price' 또는 'data.items[0].value'입니다. 숫자 또는 숫자 문자열을 사용할 수 있으며, 로캘과 관계없이 소수점 구분 기호는 '.'입니다. 비워 두면 메시지 전체를 숫자로 읽습니다.
ValueFormat	string	카드의 Value 텍스트에 사용하는 서식 문자열이며, 숫자와 함께 Format에 전달됩니다. 기본값은 '%.2f'입니다. 출력에는 애플리케이션 로캘의 소수점 구분 기호가 사용됩니다.
TitlePath	string	설정하면 해당 JSON 값을 카드의 Title로 복사합니다.
TrendValuePath	string	설정하면 해당 JSON 값을 카드의 TrendValue로 복사합니다.

메서드

메서드	반환값	설명
ProcessChunk(const aText: string)	Integer	스트림 청크 하나에 들어 있는 완전한 이벤트를 Client에서 온 것처럼 읽습니다. 불완전한 이벤트는 다음 청크를 기다립니다. 데이터에서 값을 얻은 이벤트의 개수를 반환합니다.
ResetStream		버퍼에 남은 불완전한 이벤트를 버립니다. 예를 들어 새 스트림을 시작하기 전에 사용합니다.

이벤트

이벤트	형식	설명
OnValue	TsgcDashboardSourceValueEvent	procedure(Sender: TObject; aValue: Double). 값을 읽어 카드를 갱신했습니다. Delphi 2010 이상에서는 메인 스레드에서 발생합니다.

예제

참고 항목

- TsgcDashboardSource
- TsgcDashboardSource_WebSocket
- 실시간 데이터

레이아웃

대시보드는 타일을 그리드에 배치합니다. 그리드와 셀은 VCL과 FMX에서 동일하며, 프레임워크 중립 레이아웃 엔진 TsgcDashboardLayout(유닛 sgcDashboard_Layout)이 계산하고, 대시보드의 **Layout** 속성으로 노출됩니다.

그리드 메트릭

속성	기본 값	의미
Columns	12	열의 개수입니다. 열 너비는 대시보드의 너비에서 양쪽의 Padding과 간격을 뺀 값을 Columns로 나눈 것입니다.
RowHeight	120	행 하나의 높이이며, 장치 독립 픽셀 단위입니다. 대시보드에는 타일에 필요한 만큼의 행이 있으며 세로로 스크롤됩니다.
Gap	12	가로와 세로로 두 타일 사이의 간격입니다.
Padding	12	대시보드의 테두리와 그리드 사이의 간격입니다.
AutoFlow	True	Col 또는 Row가 -1인 타일을 첫 번째 빈 슬롯에 배치합니다.

모든 크기는 장치 독립 픽셀입니다. VCL 대시보드는 모니터 PPI를 96으로 나눈 값으로, FMX 대시보드는 폼 배율로 크기를 조정합니다.

타일 셀

모든 타일은 **Col**, **Row**, **ColSpan**, **RowSpan**으로 셀을 요청합니다. Col과 Row는 0부터 시작합니다. 새 타일은 Col과 Row가 -1이고 3열 x 2행을 차지합니다. 엔진은 타일을 배치하기 전에 셀을 보정합니다. ColSpan은 1부터 Columns까지, RowSpan은 최소 1로, Col은 타일이 마지막 열을 넘지 않도록 조정합니다.

배치와 충돌 규칙

- **1단계, 명시적 셀.** Col과 Row가 모두 설정된 타일이 컨트롤 순서대로 배치됩니다. 셀이 이미 배치된 타일과 겹치면, 셀이 비워질 때까지 타일을 한 행씩 아래로 밀어냅니다. 열은 바뀌지 않습니다.
- **2단계, 자동 배치.** AutoFlow가 True인 동안 Col 또는 Row가 -1인 타일이 첫 번째 빈 슬롯을 채우며, 위쪽 행부터 왼쪽 열부터 스캔합니다. 지정된 좌표는 스캔을 제한합니다. Col이 6이고 Row가 -1인 타일은 6열에서 첫 번째 빈 행으로 이동합니다.
- AutoFlow가 False이면 Col 또는 Row의 -1은 0으로 읽히고, 타일은 1단계에서 배치됩니다.
- 숨겨진 타일(Visible이 False)은 셀을 차지하지 않습니다.

요청한 셀은 Col, Row, ColSpan, RowSpan에 남아 있습니다. 보정과 충돌 해결을 거쳐 배치된 셀은 **Layout.PlacedCell[Index]**에, 그 사각형은 **Layout.Rect[Index]**에 있습니다.

예제

참고 항목

- TsgcVCLDashboard
- TsgcVCLDashboardTile
- 영속성

테마

대시보드의 **Theme** 속성은 sgcUICore_Theme의 TsgcUITheme 형식이며, **utLight**(기본값), **utDark**, **utCustom** 중에서 선택합니다. 대시보드는 테마로 배경을 그리며, **ParentTheme**이 True인 모든 타일이 이를 따릅니다. 타일은 자체 **Theme**을 설정할 수 있으며, 그러면 ParentTheme이 해제됩니다. 테마는 변경될 때와 컨트롤이 타일에 삽입될 때 적용됩니다.

테마 적용기가 호스팅된 컨트롤에 설정하는 항목

타일은 호스팅하는 모든 컨트롤에 대해, 컨트롤이 노출하는 게시된 속성을 통해 테마를 전달하며, 호스팅되는 제품에 대한 컴파일 타임 의존성은 없습니다. 처음 일치하는 규칙이 적용됩니다.

1. TsgcUITheme 형식의 쓰기 가능한 게시된 **Theme**이 설정됩니다. TsgcVCLKPICard와 TsgcFMXKPICard가 여기에 해당합니다.
2. 그렇지 않으면 그러한 Theme을 가진 게시된 **Chart** 객체에 테마가 적용됩니다. sgcCharts의 TsgcVCLChart 또는 TsgcFMXChart의 **Chart.Theme**입니다.
3. 그렇지 않으면 쓰기 가능한 게시된 **BackgroundColor**가 테마의 표면 색상을 받습니다. 예를 들어 sgcPDF 뷰어입니다.
4. 그렇지 않으면 쓰기 가능한 게시된 **Color**가 표면 색상을 받습니다. 예를 들어 sgcMaps 지도, TMemo, TListBox입니다.

규칙 3 또는 4에 도달하면 어두운 표면에서도 텍스트가 읽히도록 테마의 텍스트 색상도 설정됩니다. VCL에서는 **Font.Color**, FMX에서는 **TextSettings.FontColor** 또는 게시된 **FontColor**입니다. 호스팅된 컨트롤의 게시된 **Viewer** 객체에도 같은 규칙으로 테마가 한 단계 깊이까지 적용됩니다.
utCustom에서는 적용기가 호스팅된 모든 컨트롤을 건드리지 않으므로, 애플리케이션이 색상을 완전히 제어합니다. 코드에서 컨트롤을 변경한 후 테마를 다시 전달하려면 대시보드의 **ApplyTheme**을 호출하십시오.

알려진 제한 사항

- sgcReports 미리 보기는 색상 속성을 게시하지 않으므로, 두 테마 모두에서 자체 색상을 유지합니다.
- 위의 속성을 하나도 게시하지 않는 컨트롤은 변경되지 않습니다.
- 타일의 직접 자식 컨트롤과 그 Viewer 객체에만 테마가 적용됩니다. 타일의 패널 안에 중첩된 컨트롤은 색상을 유지합니다.

예제

참고 항목

- TsgcVCLDashboardTile
- TsgcVCLKPICard
- 데모

영속성

Registered 또는 Trial 빌드에서는 사용자가 실행 중에 타일을 재배치할 수 있으며, 그 배치를 JSON으로 저장합니다. 이 멤버들은 SGC_DASHBOARD_PERSIST로 컴파일되며 Community 에디션에는 존재하지 않습니다.

실행 중 끌기 및 크기 조정

- **AllowDrag**: 타일의 헤더에서 마우스 왼쪽 버튼을 누르면 타일이 이동합니다. ShowHeader가 False인 타일은 이동할 수 없습니다.
- **AllowResize**: 모든 타일의 오른쪽 아래 모서리에 그립이 그려지며, 이를 끌면 타일의 크기가 조정됩니다.
- 마우스를 움직이는 동안 대상 셀이 타일 아래에 그려지고 타일은 그리드에 맞춰집니다. Esc 키를 누르면 끌기가 취소됩니다.
- **Locked**: 잠긴 타일은 이동하거나 크기를 조정할 수 없습니다. FMX에서는 표준 Locked 속성입니다.
- **OnLayoutChanged**는 끌기 또는 크기 조정으로 셀이 바뀐 후, 그리고 LoadLayout 이후에 발생합니다.

SaveLayout과 LoadLayout

SaveLayout은 그리드 메트릭과, Name이 있는 모든 타일의 셀 및 표시 여부를 반환합니다. **LoadLayout**은 이러한 텍스트를 적용합니다. 타일은 Name으로 대응되고, 알 수 없는 이름은 무시되며, 텍스트에 없는 타일은 기존 셀을 유지합니다. 파일 및 스트림 변형은 BOM 없는 UTF-8을 사용하며 불러올 때 BOM을 건너뜁니다.

JSON 형식

이름이 있는 타일 두 개가 있는 대시보드에 대한 SaveLayout의 출력이며, 여기서는 읽기 쉽게 서식을 적용했습니다. SaveLayout은 이를 한 줄로 기록합니다.

```
{
  "version":1,
  "columns":12,
  "rowHeight":120,
  "gap":12,
  "padding":12,
  "tiles": [
    {"name":"tileSales","col":0,"row":0,"colSpan":6,"rowSpan":2,"visible":true},
    {"name":"tileLog","col":6,"row":0,"colSpan":6,"rowSpan":3,"visible":true}
  ]
}
```

키	의미
version	형식 버전이며 1입니다.
columns, rowHeight, gap, padding	대시보드의 그리드 메트릭입니다.
tiles	이름이 있는 타일마다 객체 하나이며, name, col, row, colSpan, rowSpan, visible이 들어 있습니다.

참고 항목

- TsgcVCLDashboard
- 레이아웃
- 데모
- 에디션

실시간 데이터

KPI 카드에는 코드에서 **Value**, **Trend**, **AddValue**로 데이터를 공급하거나, sgcDashboardWS 브리지를 통해 실시간 연결에서 공급할 수 있습니다. 브리지 소스는 sgcWebSockets 클라이언트가 받은 모든 JSON 메시지에서 숫자를 읽어 카드에 넣습니다. 이 브리지는 Community 에디션에 포함되지 않습니다.

소스	Client	읽는 대상
TsgcDashboardSource_WebSocket	TsgcWebSocketClient	모든 텍스트 메시지.
TsgcDashboardSource_MQTT	TsgcWSPClient_MQTT	토픽이 Topic과 일치하는 모든 publish.
TsgcDashboardSource_SSE	TsgcHTTP1Client	모든 서버 전송 이벤트의 데이터.

경로와 서식

- **ValuePath**는 숫자를 가리키는 JSON 경로입니다. 예를 들어 price 또는 data.items[0].value입니다. 숫자 또는 숫자 문자열을 사용할 수 있으며, 로캘과 관계없이 소수점 구분 기호는 .입니다. ValuePath가 비어 있으면 메시지 전체를 숫자로 읽습니다.
- **ValueFormat**은 Format으로 카드의 Value 텍스트에 사용할 숫자의 서식을 지정합니다. 기본값은 %.2f입니다. 텍스트에는 애플리케이션 로캘의 소수점 구분 기호가 사용됩니다.
- **TitlePath**와 **TrendValuePath**는 설정하면 해당 JSON 값을 카드의 Title과 TrendValue로 복사합니다.
- 카드의 Trend는 같은 소스의 이전 값을 따릅니다. 더 높으면 ktUp, 더 낮으면 ktDown, 같으면 ktNeutral입니다.
- 올바른 JSON이 아니거나 ValuePath가 숫자가 아닌 메시지는 무시됩니다.

스레딩

- 숫자는 메시지가 도착한 스레드에서 즉시 스파크라인에 추가됩니다. TsgcDashboardKPI.AddValue는 스레드에 안전합니다.
- Delphi 2010 이상에서는 텍스트 갱신(Value, Title, TrendValue, Trend)이 TThread.Queue를 통해 메인 스레드에서 적용되며, 그다음 OnValue가 발생합니다. 클라이언트는 어떤 NotifyEvents 모드든 사용할 수 있습니다.
- 콘솔 애플리케이션과 서비스에는 큐를 실행하는 메시지 루프가 없으므로 CheckSynchronize를 호출해야 합니다.
- Delphi 7부터 2009까지는 TThread.Queue가 없습니다. 이 버전에서는 갱신과 OnValue가 메시지가 도착한 스레드에서 실행되며, OnValue 핸들러는 컨트롤을 건드리기 전에 직접 동기화해야 합니다.

예제

참고 항목

- TsgcDashboardSource
- TsgcVCLKPICard
- 데모

헤드리스 KPI 이미지

TsgcDashboardKPI는 창도 VCL이나 FMX도 없이 KPI 카드를 렌더링하므로, 콘솔 애플리케이션, 서비스, 웹 서버가 같은 카드를 이미지로 만들 수 있습니다. 예를 들어 보고서, 이메일, 웹 페이지에 사용할 수 있습니다. 이 유닛은 Linux64를 포함한 모든 플랫폼에서 빌드됩니다.

- **SaveToPNG**는 PNG 파일 또는 스트림을 기록합니다. aWidth와 aHeight는 논리 픽셀이며 aScale이 이를 곱합니다. 배율 2에서 320 x 160은 640 x 320 이미지가 됩니다.
- **SaveToSVG**는 같은 카드의 SVG 파일 또는 스트림을 기록합니다.
- 컨트롤의 카드는 **KPI** 속성으로 사용할 수 있으므로, **Card.KPI.SaveToPNG**는 컨트롤이 표시하는 내용을 원하는 크기로 저장합니다.

예제

참고 항목

- TsgcDashboardKPI
- 플랫폼

.NET 개요

sgcDashboard .NET은 Delphi sgcDashboard 라이브러리를 관리 코드 C#으로 이식한 것이며, Delphi 라이브러리를 감싸는 래퍼가 아닙니다. C# 클래스는 **esegece.sgcDashboard** 네임스페이스에 있고 Delphi 이름을 그대로 유지하므로, **TsgcDashboardLayout**, **TsgcDashboardCell**, **TsgcDashboardKPI**, **TsgcDashboardKPITrend**, **IsgcDashboardKPIHost**는 두 언어에서 똑같이 읽히며, 각 호스트의 대시보드, 타일, KPI 카드도 Delphi의 명명 규칙을 따릅니다. 이들은 **sgcUICore**의 .NET 포트로 그려지므로 eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 라이트 및 다크 테마를 사용하며, 대시보드는 WinForms, WPF, Avalonia, 헤드리스 PNG 및 SVG 출력에서 모두 같게 보입니다.
대시보드는 열과 행으로 이루어진 그리드에 타일을 배치합니다. 각 타일에는 셀(열, 행, 그리고 두 방향의 스팬), 제목이 있는 선택적 헤더가 있으며, KPI 카드, sgcCharts 차트, sgcPDF 뷰어, sgcReports 미리 보기, 일반 텍스트 상자 등 모든 컨트롤을 호스팅합니다. 대시보드 테마는 타일과 타일이 호스팅하는 컨트롤에도 적용됩니다. Registered 에디션에서는 사용자가 실행 중에 타일을 끌고 크기를 조정할 수 있고, 레이아웃은 JSON으로 저장 및 로드되며, sgcWebSockets 브리지가 라이브 연결에서 KPI 카드에 데이터를 공급합니다.

패키지

이 포트는 여섯 개의 NuGet 패키지로 나뉘어 있으므로, 프로젝트에는 사용하는 호스트만 포함하면 됩니다.

PackageId	내용	대상 프레임워크
esegece.sgcDashboard	레이아웃 엔진 **TsgcDashboardLayout**과 JSON 영속성, KPI 모델 **TsgcDashboardKPI**, 추세 열거형, 헤드리스 SVG 출력. 네이티브 종속성은 없습니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcDashboard.Rendering	SkiaSharp를 통한 KPI 카드의 헤드리스 PNG 출력(**SaveToPNG**).	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcDashboard.WinForms	**TsgcWinFormsDashboard**, **TsgcWinFormsDashboardTile**, **TsgcWinFormsKPICard**.	net472, net8.0-windows, net9.0-windows
esegece.sgcDashboard.Wpf	**TsgcWpfDashboard**, **TsgcWpfDashboardTile**, **TsgcWpfKPICard**.	net472, net8.0-windows, net9.0-windows
esegece.sgcDashboard.Avalonia	**TsgcAvaloniaDashboard**, **TsgcAvaloniaDashboardTile**, **TsgcAvaloniaKPICard**(Avalonia 11.3).	net8.0, net9.0
esegece.sgcDashboard.sgcWebSockets	실시간 데이터 브리지: **TsgcDashboardSource**, **TsgcDashboardSource_WebSocket**, **TsgcDashboardSource_MQTT**, **TsgcDashboardSource_SSE**. esegece.sgcWebSockets 패키지에 종속됩니다. Registered 에디션 전용입니다.	netstandard2.0, net8.0, net9.0

각 호스트 패키지는 esegece.sgcDashboard.Rendering과 같은 호스트의 sgcUICore 패키지에 종속되므로, 호스트 패키지만 추가하면 충분합니다.

에디션

에디션	패키지 ID	포함 내용

Registered	esegece.sgcDashboard, .Rendering, .WinForms, .Wpf, .Avalonia, .sgcWebSockets	모든 기능: 그리드와 타일, KPI 카드, 테마, 헤드리스 출력, 실행 중 끌기 및 크기 조정, **SaveLayout**과 **LoadLayout**, **OnLayoutChanged** sgcWebSockets 브리지.
Community	같은 ID에 .Community 접미사가 붙습니다. 예: esegece.sgcDashboard.WinForms.Community. Community용 브리지 패키지는 없습니다.	그리드와 타일, KPI 카드, 테마, 헤드리스 출력. 실행 중 끌기 및 크기 조정, 레이아웃 영속성 (**SaveLayout**, **LoadLayout**, 엔진의 **SaveToJSON**과 **LoadFromJSON**), 브리지는 제외됩니다.

이는 Delphi 에디션과 같은 구분입니다. 에디션을 참조하십시오.

호스트

호스트	Delphi 대응 요소	플랫폼
WinForms	TsgcVCLDashboard, TsgcVCLDashboardTile, TsgcVCLKPICard	Windows. .NET Framework 4.7.2와 .NET 8, 9를 지원합니다.
WPF	TsgcFMXDashboard, TsgcFMXDashboardTile, TsgcFMXKPICard	Windows. .NET Framework 4.7.2와 .NET 8, 9를 지원합니다.
Avalonia	TsgcFMXDashboard, TsgcFMXDashboardTile, TsgcFMXKPICard	Windows, Linux, macOS. .NET 8과 9를 지원합니다.
헤드리스	TsgcDashboardKPI와 레이아웃 엔진	Linux 서버를 포함한 .NET Standard 2.0의 모든 플랫폼.

WPF와 Avalonia 타일은 FireMonkey 타일을 따릅니다. 하나의 요소를 호스팅하고, 그 주변 공간은 **ContentPadding**으로 관리합니다. WinForms 타일은 VCL 타일을 따르며 원하는 수만큼 컨트롤을 호스팅합니다.

참고 항목

- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET 라이브 피드
- .NET 데모

NuGet에서 설치

패키지는 nuget.org에 있습니다. 호스트 패키지를 추가하면 코어 패키지와 렌더링 패키지가 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcDashboard.WinForms
dotnet add package esegece.sgcDashboard.Wpf
dotnet add package esegece.sgcDashboard.Avalonia
```

sgcWebSockets에서 KPI 카드에 데이터를 공급하려면 브리지를 추가합니다. esegece.sgcWebSockets 패키지도 함께 설치됩니다.

```
dotnet add package esegece.sgcDashboard.sgcWebSockets
```

Community 에디션

```
dotnet add package esegece.sgcDashboard.WinForms.Community
dotnet add package esegece.sgcDashboard.Wpf.Community
dotnet add package esegece.sgcDashboard.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcDashboard.Wpf" Version="2026.10.0" />
</ItemGroup>
```

한 프로젝트에서는 Registered 패키지나 Community 패키지 중 하나만 참조하고, 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있기 때문입니다. **AllowDrag**, **AllowResize**, **SaveLayout**, **LoadLayout**, **OnLayoutChanged** 또는 타일의 **Locked** 속성을 사용하는 코드는 Community 패키지에서 컴파일되지 않습니다. 한 프로젝트에서 두 에디션을 모두 빌드하는 경우 `#if !SGC_COMMUNITY`로 해당 코드를 감싸십시오.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`가 있는 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`가 있는 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서 네임스페이스는 `xmlns:dash="clr-namespace:esegece.sgcDashboard;assembly=esegece.sgcDashboard.Wpf"`로 매핑합니다.
- Avalonia: 일반 Avalonia 데스크톱 애플리케이션처럼 Avalonia 11.3의 **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 패키지를 사용하는 `net8.0` 또는 `net9.0` 프로젝트. XAML에서 네임스페이스는 `xmlns:dash="using:esegece.sgcDashboard"`로 매핑합니다.
- 헤드리스: 레이아웃 엔진과 SVG에는 esegece.sgcDashboard만으로 충분합니다. PNG에는 esegece.sgcDashboard.Rendering을 추가하십시오. SkiaSharp와 Windows 및 Linux용 네이티브 자산이 함께 설치됩니다.
- 브리지: esegece.sgcDashboard.sgcWebSockets는 netstandard2.0, net8.0, net9.0을 대상으로 하므로 .NET Framework 4.7.2를 포함한 모든 호스트에서 동작합니다.

C#에서는 모든 호스트에 `using esegece.sgcDashboard;`가 필요하고, **TsgcUITheme**과 테마의 다른 형식에는 `using esegece.sgcUICore;`가 필요합니다. 브리지에는 `using esegece.sgcWebSockets;`도 필요합니다.

참고 항목

- .NET 개요
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 라이브 피드

- .NET 데모

WinForms 빠른 시작

이 예제는 타일 세 개, 즉 KPI 카드 두 개와 일반 텍스트 상자에 표시하는 이벤트 로그로 구성된 대시보드를 WinForms 폼에 만듭니다. 타이머가 매출 카드에 데이터를 공급하고, 확인란으로 대시보드의 라이트 테마와 다크 테마를 전환합니다.

단계

- `net8.0-windows` WinForms 프로젝트를 만들고 **esegece.sgcDashboard.WinForms** 패키지를 추가합니다. NuGet에서 설치를 참조하십시오.
- **TsgcWinFormsDashboard**를 만들어 도킹하고 **Columns**와 **RowHeight**를 설정합니다. 그리드는 항상 대시보드의 너비에 맞춰지고 세로로 스크롤됩니다.
- 타일마다 **TsgcWinFormsDashboardTile**을 만들고 **Name**, **Title**, 셀(**Col**, **Row**, **ColSpan**, **RowSpan**)을 지정합니다. 콘텐츠는 타일의 **Controls**에, 타일은 대시보드의 **Controls**에 추가합니다. 도킹된 컨트롤은 헤더 아래 영역을 채웁니다.
- **TsgcWinFormsKPICard**의 **Value**, **Trend**, **TrendValue**, **FooterText**를 설정하고, **AddValue**를 호출하여 스파크라인에 점을 추가합니다.
- 대시보드의 **Theme**을 설정합니다. 타일, 카드, 텍스트 상자가 이를 따릅니다.

예제

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcDashboard;
using esegece.sgcUICore;

namespace DashboardQuickStartWinForms
{
    public class MainForm : Form
    {
        private readonly TsgcWinFormsDashboard Dashboard;
        private readonly TsgcWinFormsKPICard kpiRevenue;
        private readonly TsgcWinFormsKPICard kpiUsers;
        private readonly TextBox memoLog;
        private readonly Timer tmrUpdate;
        private readonly Random FRandom = new Random();
        private double FRevenue = 12500;

        public MainForm()
        {
            Text = "sgcDashboard WinForms";
            ClientSize = new Size(900, 520);

            // a 12 column grid, each row 110 pixels high
            Dashboard = new TsgcWinFormsDashboard();
            Dashboard.Dock = DockStyle.Fill;
            Dashboard.Columns = 12;
            Dashboard.RowHeight = 110;
            // runtime drag and resize (Registered edition)
            Dashboard.AllowDrag = true;
            Dashboard.AllowResize = true;

            kpiRevenue = new TsgcWinFormsKPICard();
            kpiRevenue.Title = "Revenue";
            kpiRevenue.Icon = "$";
            kpiRevenue.FooterText = "Today, USD";
            AddTile("tileRevenue", "Revenue", 0, 0, 6, 2, kpiRevenue);

            kpiUsers = new TsgcWinFormsKPICard();
            kpiUsers.Title = "Active users";
```

```
            kpiUsers.Icon = "U";
            kpiUsers.Value = "340";
            kpiUsers.FooterText = "Signed in now";
            AddTile("tileUsers", "Users", 6, 0, 6, 2, kpiUsers);

            // a tile hosts any control, here an ordinary TextBox
            memoLog = new TextBox();
            memoLog.Multiline = true;
            memoLog.ReadOnly = true;
            memoLog.BorderStyle = BorderStyle.None;
            AddTile("tileLog", "Event log", 0, 2, 12, 2, memoLog);

            CheckBox chkDark = new CheckBox();
            chkDark.Text = "Dark theme";
            chkDark.Dock = DockStyle.Top;
            // the dashboard theme reaches every tile and the controls they host
            chkDark.CheckedChanged += (s, e) =>
                Dashboard.Theme = chkDark.Checked ? TsgcUITheme.utDark : TsgcUITheme.utLight;

            Controls.Add(Dashboard);
            Controls.Add(chkDark);

            tmrUpdate = new Timer();
            tmrUpdate.Interval = 1000;
            tmrUpdate.Tick += (s, e) => UpdateRevenue();
            tmrUpdate.Enabled = true;
        }

        private void AddTile(string aName, string aTitle, int aCol, int aRow, int aColSpan, int aRowSpan,
            Control aContent)
        {
            TsgcWinFormsDashboardTile oTile = new TsgcWinFormsDashboardTile();
            oTile.Name = aName;
            oTile.Title = aTitle;
            oTile.Col = aCol;
            oTile.Row = aRow;
            oTile.ColSpan = aColSpan;
            oTile.RowSpan = aRowSpan;
            aContent.Dock = DockStyle.Fill;
            oTile.Controls.Add(aContent);
            Dashboard.Controls.Add(oTile);
        }

        private void UpdateRevenue()
        {
            double vChange = FRandom.Next(-200, 400);
            FRevenue += vChange;
            kpiRevenue.Value = FRevenue.ToString("N0");
            kpiRevenue.TrendValue = vChange.ToString("+0;-0");
            kpiRevenue.Trend = vChange >= 0 ? TsgcDashboardKPITrend.ktUp : TsgcDashboardKPITrend.ktDown;
            kpiRevenue.AddValue(FRevenue);
            memoLog.AppendText(DateTime.Now.ToLongTimeString() + " revenue " + kpiRevenue.Value + Environment.NewLine);
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new MainForm());
        }
    }
}
```

AllowDrag와 **AllowResize**를 사용하면 사용자가 타일을 헤더로 이동하고 오른쪽 아래 그립으로 크기를 조정할 수 있으며, 둘 다 그리드에 맞춰집니다. 이 기능은 Registered 에디션에 포함됩니다. Community 패키지에서는 두 줄을 제거하십시오.

레이아웃 저장 및 복원

SaveLayout은 그리드 메트릭과 이름이 있는 각 타일의 셀 및 표시 여부를 JSON으로 반환하고, **LoadLayout**은 그 텍스트를 다시 적용합니다. **SaveLayoutToFile** 과 **LoadLayoutFromFile** 은 파일로, **SaveLayoutToStream** 과 **LoadLayoutFromStream**은 UTF-8 스트림으로 같은 작업을 합니다. **OnLayoutChanged**는 끌기로 셀이 바뀐 후와 로드 후에 발생합니다. 레이아웃 영속성은 Registered 에디션에 포함됩니다.

```
using System;
using System.IO;
using esegece.sgcDashboard;

namespace DashboardLayoutRoundTrip
{
    internal static class Program
    {
        private static TsgcWinFormsDashboardTile AddTile(TsgcWinFormsDashboard aDashboard, string aName,
            int aCol, int aRow, int aColSpan, int aRowSpan)
        {
            TsgcWinFormsDashboardTile oTile = new TsgcWinFormsDashboardTile();
            oTile.Name = aName;
            oTile.Title = aName;
            oTile.SetCell(sgcDashboard_Layout.sgcDashboardCell(aCol, aRow, aColSpan, aRowSpan));
            aDashboard.Controls.Add(oTile);
            return oTile;
        }

        private static string CellText(TsgcWinFormsDashboardTile aTile)
        {
            return aTile.Name + " at " + aTile.Col + "," + aTile.Row + " size " + aTile.ColSpan + "x" + aTile.RowSpan;
        }

        [STAThread]
        private static int Main()
        {
            using (TsgcWinFormsDashboard oDashboard = new TsgcWinFormsDashboard())
            {
                oDashboard.SetBounds(0, 0, 800, 600);
                TsgcWinFormsDashboardTile oSales = AddTile(oDashboard, "tileSales", 0, 0, 6, 2);
                TsgcWinFormsDashboardTile oOrders = AddTile(oDashboard, "tileOrders", 6, 0, 6, 2);
                int vChanges = 0;
                oDashboard.OnLayoutChanged = (Sender) => vChanges++;

                // the tile names are the keys of the saved layout
                string vLayout = oDashboard.SaveLayout();
                Console.WriteLine(vLayout);

                // the user rearranges the tiles
                oSales.SetCell(sgcDashboard_Layout.sgcDashboardCell(0, 2, 12, 3));
                oOrders.SetCell(sgcDashboard_Layout.sgcDashboardCell(0, 0, 4, 2));

                // back to the saved layout
                oDashboard.LoadLayout(vLayout);
                Console.WriteLine(CellText(oSales));
                Console.WriteLine(CellText(oOrders));

                // the same text through a file
                string vFile = Path.Combine(Path.GetTempPath(), "dashboard_layout.json");
                oDashboard.SaveLayoutToFile(vFile);
                oDashboard.LoadLayoutFromFile(vFile);
                bool vSame = oDashboard.SaveLayout() == vLayout;
                Console.WriteLine("OnLayoutChanged: " + vChanges + ", round trip: " + (vSame ? "ok" : "FAILED"));
                return vSame ? 0 : 1;
            }
```

```
        }
    }
}
```

프로그램은 다음을 출력합니다.

```
{"version":1,"columns":12,"rowHeight":120,"gap":12,"padding":12,"tiles":[{"name":"tileSales","col":0,"row":0,"colSpan":6,"r
tileSales at 0,0 size 6x2
tileOrders at 6,0 size 6x2
OnLayoutChanged: 2, round trip: ok
```

이 JSON은 Delphi의 **SaveLayout**이 쓰는 텍스트와 같으므로, Delphi 애플리케이션에서 저장한 레이아웃을 .NET 애플리케이션에서 로드할 수 있고 그 반대도 가능합니다. 타일은 대소문자를 구분하지 않고 **Name**으로 대응되며, 이름이 없는 타일은 저장되지 않습니다.

참고 항목

- NuGet에서 설치
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- 영속성
- TsgcVCLDashboard
- .NET 데모

WPF 빠른 시작

같은 대시보드를 WPF로 만듭니다. 타일과 타일이 호스팅하는 요소는 XAML로 선언합니다. 타일은 대시보드의 콘텐츠이고, 각 타일은 하나의 요소를 콘텐츠로 받습니다.

단계

- `net8.0-windows` WPF 프로젝트를 만들고 **esegece.sgcDashboard.Wpf** 패키지를 추가합니다. NuGet에서 설치를 참조하십시오.
- 창에서 네임스페이스를 `xmlns:dash="clr-namespace:esegece.sgcDashboard;assembly=esegece.sgcDashboard.Wpf"`로 매핑합니다.
- **TsgcWpfDashboard**를 선언하고, 그 안에 타일마다 **TsgcWpfDashboardTile** 하나를 **Title**, **Col**, **Row**, **ColSpan**, **RowSpan**과 함께 선언합니다. 타일의 콘텐츠는 **TsgcWpfKPICard** 또는 다른 모든 요소입니다.
- 타일에 `x:Name`을 지정합니다. 이 이름이 **SaveLayout**에 저장됩니다.
- 카드에 데이터를 공급하고 **Theme**을 전환하는 작업은 코드 비하인드에서 합니다.

예제

MainWindow.xaml:

```
<Window x:Class="DashboardQuickStartWpf.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:dash="clr-namespace:esegece.sgcDashboard;assembly=esegece.sgcDashboard.Wpf"
        Title="sgcDashboard WPF" Width="900" Height="560">
    <DockPanel>
        <CheckBox x:Name="chkDark" DockPanel.Dock="Top" Margin="8" Content="Dark theme"
                  Checked="OnThemeChanged" Unchecked="OnThemeChanged" />
        <dash:TsgcWpfDashboard x:Name="Dashboard" Columns="12" RowHeight="110"
                               AllowDrag="True" AllowResize="True">
            <dash:TsgcWpfDashboardTile x:Name="tileRevenue" Title="Revenue" Col="0" Row="0" ColSpan="6" RowSpan="2">
                <dash:TsgcWpfKPICard x:Name="kpiRevenue" Title="Revenue" Icon="$" FooterText="Today, USD" />
            </dash:TsgcWpfDashboardTile>
            <dash:TsgcWpfDashboardTile x:Name="tileUsers" Title="Users" Col="6" Row="0" ColSpan="6" RowSpan="2">
                <dash:TsgcWpfKPICard x:Name="kpiUsers" Title="Active users" Icon="U" Value="340"
                                     FooterText="Signed in now" />
            </dash:TsgcWpfDashboardTile>
            <dash:TsgcWpfDashboardTile x:Name="tileLog" Title="Event log" Col="0" Row="2" ColSpan="12" RowSpan="2">
                <TextBox x:Name="memoLog" IsReadOnly="True" AcceptsReturn="True" BorderThickness="0"
                         VerticalScrollBarVisibility="Auto" />
            </dash:TsgcWpfDashboardTile>
        </dash:TsgcWpfDashboard>
    </DockPanel>
</Window>
```

MainWindow.xaml.cs:

```
using System;
using System.Windows;
using System.Windows.Threading;
using esegece.sgcDashboard;
using esegece.sgcUICore;

namespace DashboardQuickStartWpf
{
    public partial class MainWindow : Window
    {
        private readonly DispatcherTimer tmrUpdate = new DispatcherTimer();
        private readonly Random FRandom = new Random();
        private double FRevenue = 12500;
```

```
        public MainWindow()
        {
            InitializeComponent();
            tmrUpdate.Interval = TimeSpan.FromSeconds(1);
            tmrUpdate.Tick += (s, e) => UpdateRevenue();
            tmrUpdate.Start();
        }

        private void OnThemeChanged(object sender, RoutedEventArgs e)
        {
            // the dashboard theme reaches every tile and the element it hosts
            Dashboard.Theme = chkDark.IsChecked == true ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        }

        private void UpdateRevenue()
        {
            double vChange = FRandom.Next(-200, 400);
            FRevenue += vChange;
            kpiRevenue.Value = FRevenue.ToString("N0");
            kpiRevenue.TrendValue = vChange.ToString("+0;-0");
            kpiRevenue.Trend = vChange >= 0 ? TsgcDashboardKPITrend.ktUp : TsgcDashboardKPITrend.ktDown;
            kpiRevenue.AddValue(FRevenue);
            memoLog.AppendText(DateTime.Now.ToLongTimeString() + " revenue " + kpiRevenue.Value + Environment.NewLine);
        }
    }
}
```

App.xaml은 새 WPF 프로젝트의 것과 같으며 `StartupUri="MainWindow.xaml"`이 지정되어 있습니다. **AllowDrag**와 **AllowResize**는 Registered 에디션에 포함됩니다. Community 패키지에서는 두 특성을 제거하십시오.

참고 항목

- NuGet에서 설치
- WinForms 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- TsgcFMXDashboard
- .NET 데모

Avalonia 빠른 시작

같은 대시보드를 Windows, Linux, macOS용 Avalonia로 만듭니다. WPF와 마찬가지로 타일과 타일의 콘텐츠는 XAML로 선언합니다.

단계

- **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 11.3을 사용하는 `net8.0` Avalonia 데스크톱 프로젝트를 만들고 **esegece.sgcDashboard.Avalonia** 패키지를 추가합니다. NuGet에서 설치를 참조하십시오.
- 창에서 네임스페이스를 `xmlns:dash="using:esegece.sgcDashboard"`로 매핑합니다.
- **TsgcAvaloniaDashboard**를 선언하고, 그 안에 타일마다 **TsgcAvaloniaDashboardTile** 하나를 선언합니다. 타일의 콘텐츠는 **TsgcAvaloniaKPICard** 또는 다른 모든 컨트롤입니다.
- 카드에 데이터를 공급하고 **Theme**을 전환하는 작업은 코드 비하인드에서 합니다.

예제

MainWindow.axaml:

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:dash="using:esegece.sgcDashboard"
        x:Class="DashboardQuickStartAvalonia.MainWindow"
        Title="sgcDashboard Avalonia" Width="900" Height="560">
    <DockPanel>
        <CheckBox x:Name="chkDark" DockPanel.Dock="Top" Margin="8" Content="Dark theme"
                  IsCheckedChanged="OnThemeChanged" />
        <dash:TsgcAvaloniaDashboard x:Name="Dashboard" Columns="12" RowHeight="110"
                                    AllowDrag="True" AllowResize="True">
            <dash:TsgcAvaloniaDashboardTile x:Name="tileRevenue" Title="Revenue" Col="0" Row="0" ColSpan="6" RowSpan="2">
                <dash:TsgcAvaloniaKPICard x:Name="kpiRevenue" Title="Revenue" Icon="$" FooterText="Today, USD" />
            </dash:TsgcAvaloniaDashboardTile>
            <dash:TsgcAvaloniaDashboardTile x:Name="tileUsers" Title="Users" Col="6" Row="0" ColSpan="6" RowSpan="2">
                <dash:TsgcAvaloniaKPICard x:Name="kpiUsers" Title="Active users" Icon="U" Value="340"
                                          FooterText="Signed in now" />
            </dash:TsgcAvaloniaDashboardTile>
            <dash:TsgcAvaloniaDashboardTile x:Name="tileLog" Title="Event log" Col="0" Row="2" ColSpan="12" RowSpan="2">
                <TextBox x:Name="memoLog" IsReadOnly="True" AcceptsReturn="True" TextWrapping="Wrap" />
            </dash:TsgcAvaloniaDashboardTile>
        </dash:TsgcAvaloniaDashboard>
    </DockPanel>
</Window>
```

MainWindow.axaml.cs:

```
using System;
using Avalonia.Controls;
using Avalonia.Interactivity;
using Avalonia.Markup.Xaml;
using Avalonia.Threading;
using esegece.sgcDashboard;
using esegece.sgcUICore;

namespace DashboardQuickStartAvalonia
{
    public partial class MainWindow : Window
    {
        private readonly DispatcherTimer tmrUpdate = new DispatcherTimer();
        private readonly Random FRandom = new Random();
        private double FRevenue = 12500;

        public MainWindow()
```

```
        {
            AvaloniaXamlLoader.Load(this);
            tmrUpdate.Interval = TimeSpan.FromSeconds(1);
            tmrUpdate.Tick += (s, e) => UpdateRevenue();
            tmrUpdate.Start();
        }

        private void OnThemeChanged(object sender, RoutedEventArgs e)
        {
            // Theme is declared new on the dashboard, so it is the sgc theme,
            // not the Avalonia ControlTheme of StyledElement
            TsgcAvaloniaDashboard oDashboard = this.FindControl<TsgcAvaloniaDashboard>("Dashboard");
            CheckBox oDark = this.FindControl<CheckBox>("chkDark");
            oDashboard.Theme = oDark.IsChecked == true ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        }

        private void UpdateRevenue()
        {
            TsgcAvaloniaKPICard oRevenue = this.FindControl<TsgcAvaloniaKPICard>("kpiRevenue");
            TextBox oLog = this.FindControl<TextBox>("memoLog");
            double vChange = FRandom.Next(-200, 400);
            FRevenue += vChange;
            oRevenue.Value = FRevenue.ToString("N0");
            oRevenue.TrendValue = vChange.ToString("+0;-0");
            oRevenue.Trend = vChange >= 0 ? TsgcDashboardKPITrend.ktUp : TsgcDashboardKPITrend.ktDown;
            oRevenue.AddValue(FRevenue);
            oLog.Text += DateTime.Now.ToLongTimeString() + " revenue " + oRevenue.Value + Environment.NewLine;
        }
    }
}
```

Program.cs:

```
using System;
using Avalonia;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;

namespace DashboardQuickStartAvalonia
{
    internal class App : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new MainWindow();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main(string[] args)
        {
            BuildAvaloniaApp().StartWithClassicDesktopLifetime(args);
        }

        public static AppBuilder BuildAvaloniaApp()
        {
```

```
            return AppBuilder.Configure<App>().UsePlatformDetect().LogToTrace();
        }
    }
}
```

타일은 **ContentPadding**과 **ShowHeader**로부터 자신의 **Padding**을 관리하므로, XAML에서 설정한 **Padding**은 대체됩니다. 터치 화면에서는 타일 헤더를 누르면 타일을 끌고, 다른 곳을 누르면 그리드를 이동(팬)합니다. Delphi와 .NET의 차이점을 참조하십시오.

참고 항목

- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- TsgcFMXDashboard
- .NET 데모

.NET 클래스

모든 클래스는 **esegece.sgcDashboard** 네임스페이스에 있습니다. 호스트 클래스는 프레임워크와 무관한 엔진과 모델 위에 얹힌 얇은 컨트롤이며, 세 호스트와 헤드리스 출력이 이를 공유합니다.

호스트 클래스

호스트	클래스	기본 클래스	Delphi
WinForms	TsgcWinFormsDashboard	ScrollableControl	TsgcVCLDashboard
WinForms	TsgcWinFormsDashboardTile	TsgcUIWinFormsControl	TsgcVCLDashboardTile
WinForms	TsgcWinFormsKPICard	TsgcUIWinFormsControl	TsgcVCLKPICard
WPF	TsgcWpfDashboard	FrameworkElement	TsgcFMXDashboard
WPF	TsgcWpfDashboardTile	TsgcUIWpfControl	TsgcFMXDashboardTile
WPF	TsgcWpfKPICard	TsgcUIWpfControl	TsgcFMXKPICard
Avalonia	TsgcAvaloniaDashboard	TsgcUIAvaloniaControl	TsgcFMXDashboard
Avalonia	TsgcAvaloniaDashboardTile	TsgcUIAvaloniaControl	TsgcFMXDashboardTile
Avalonia	TsgcAvaloniaKPICard	TsgcUIAvaloniaControl	TsgcFMXKPICard

프레임워크와 무관한 형식

형식	설명
TsgcDashboardLayout	그리드 엔진: **Columns**, **RowHeight**, **Gap**, **Padding**, **AutoFlow**, 타일 셀을 사각형으로 변환(**Arrange**, **Rect**, **ContentHeight**), 자동 흐름, 충돌 시 아래로 밀기, 끌기와 크기 조정의 스냅 계산(**SnapMove**, **SnapResize**), 레이아웃 JSON(**SaveToJSON**, **LoadFromJSON**, Registered 에디션).
TsgcDashboardCell	**Col**, **Row**, **ColSpan**, **RowSpan**을 가진 구조체. `sgcDashboard_Layout.sgcDashboardCell` 로 만듭니다.
TsgcDashboardLayoutTile	레이아웃 JSON에 있는 타일 하나의 이름, 셀, 표시 여부. Registered 에디션.
TsgcDashboardKPI	KPI 모델과 렌더러. TsgcDashboardKPI를 참조하십시오.
TsgcDashboardKPITrend	`ktNone` , `ktUp` , `ktDown` , `ktNeutral` 중 하나.
IsgcDashboardKPIHost	세 KPI 카드 컨트롤이 구현합니다. 브리지는 이를 구현하는 모든 카드를 받아들입니다.

대시보드

TsgcWinFormsDashboard, **TsgcWpfDashboard**, **TsgcAvaloniaDashboard**에 공통된 멤버.

멤버	형식	설명

Columns	int	그리드의 열 수. 기본값은 12입니다.
RowHeight	int	그리드 행의 높이(장치 독립 픽셀). 기본값은 120입니다.
Gap	int	타일 사이의 간격. 기본값은 12입니다.
Padding	int	그리드 주변의 공간. 기본값은 12입니다. WinForms에서는 `Control.Padding`을 숨깁니다.
AutoFlow	bool	셀이 없는(Col 또는 Row가 -1인) 타일을 처음으로 비어 있는 셀에 배치합니다. 기본값은 True입니다.
Theme	TsgcUITheme	대시보드, 대시보드의 타일, 타일이 호스팅하는 컨트롤의 테마.
Layout	TsgcDashboardLayout	그리드 엔진(읽기 전용). WinForms에서는 `Control.Layout` 이벤트를 숨깁니다.
Tiles	컬렉션	WPF와 Avalonia: 타일 컬렉션이며 XAML 콘텐츠 속성입니다.
VerticalOffset	double	Avalonia: 그리드의 스크롤 위치.
AllowDrag	bool	실행 중에 타일을 헤더로 이동합니다. Registered 에디션.
AllowResize	bool	실행 중에 타일을 오른쪽 아래 그립으로 크기 조정합니다. Registered 에디션.
OnLayoutChanged	TsgcUINotifyEvent	끌기로 셀이 바뀐 후와 로드 후에 발생합니다. Registered 에디션.
SaveLayout, LoadLayout	메서드	JSON 문자열로 표현한 레이아웃. Registered 에디션.
SaveLayoutToStream, LoadLayoutFromStream	메서드	UTF-8 스트림으로 표현한 레이아웃. Registered 에디션.
SaveLayoutToFile, LoadLayoutFromFile	메서드	파일로 표현한 레이아웃. Registered 에디션.
Arrange	메서드	타일을 다시 배치합니다.
ApplyTheme	메서드	테마를 타일과 타일의 콘텐츠에 다시 적용합니다. 예를 들어 콘텐츠가 자체 색을 변경한 후에 사용합니다.

타일

TsgcWinFormsDashboardTile, **TsgcWpfDashboardTile**, **TsgcAvaloniaDashboardTile**에 공통된 멤버.

멤버	형식	설명
대시보드	dashboard	타일을 포함하는 대시보드이며, 없으면 null입니다.
Title	문자열	헤더의 텍스트.
ShowHeader	bool	헤더 영역을 표시합니다. 기본값은 true입니다.
Col, Row	int	요청한 셀. 기본값은 -1이며, 이 경우 자동 흐름이 타일을 배치합니다.

ColSpan, RowSpan	int	타일의 크기(열 수와 행 수). 기본값은 3과 2입니다.
GetCell, SetCell	메서드	네 값을 하나의 **TsgcDashboardCell**로 가져오거나 설정합니다.
ParentTheme	bool	대시보드의 테마를 따릅니다. 기본값은 true입니다.
Theme	TsgcUITheme	**ParentTheme**이 false일 때 타일의 테마.
Locked	bool	타일이 끌기나 크기 조정을 시작하지 않습니다. Registered 에디션.
Content	요소	WPF와 Avalonia: 호스팅되는 요소이며 XAML 콘텐츠 속성입니다. WinForms에서 타일은 **Controls** 컬렉션의 컨트롤을 호스팅합니다.
ContentPadding	int	WPF와 Avalonia: 콘텐츠 주변의 공간. 기본값은 8이며, **ShowHeader**가 true인 동안에는 위쪽에 헤더 높이가 더해집니다.

KPI 카드

TsgcWinFormsKPICard, **TsgcWpfKPICard**, **TsgcAvaloniaKPICard**는 **KPI** 속성으로 제공되는 TsgcDashboardKPI를 감싸며 그 멤버를 다시 제공합니다: **Title**, **Value**, **Icon**, **Trend**, **TrendValue**, **FooterText**, **AccentColor**, **Theme**, **MaxPoints**, **AddValue**.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET 라이브 피드
- 레이아웃
- 테마

Delphi와 .NET의 차이점

이 .NET 포트는 Delphi의 이름, 기본값, 레이아웃 규칙을 그대로 유지합니다. 아래 차이점은 호스트 프레임워크에서 비롯됩니다.

타일은 자식 컨테이너입니다

호스트	타일이 콘텐츠를 호스팅하는 방식
WinForms	타일은 VCL 타일처럼 `Control` 컨테이너입니다. 타일의 **Controls**에 원하는 수만큼 컨트롤을 추가할 수 있습니다. 컨트롤은 헤더 아래에 8픽셀 여백을 두고 배치되므로 `Dock = DockStyle.Fill`을 지정한 컨트롤이 콘텐츠 영역을 채웁니다. 타일에는 ParentControlDesigner가 있으므로 디자이너에서 컨트롤을 타일에 놓을 수 있습니다.
WPF	타일은 FireMonkey 타일처럼 XAML 콘텐츠 속성인 **Content**(`[ContentProperty("Content")]`)를 통해 하나의 요소를 받습니다. 대시보드는 타일의 논리적 부모이고 **Tiles**가 콘텐츠 속성이므로, 타일은 대시보드 요소의 자식으로 선언합니다.
Avalonia	WPF와 같은 모델이며, 타일의 **Content**와 대시보드의 **Tiles**에 `[Content]` 특성을 사용합니다. 대시보드는 모든 타일의 논리적 부모이자 비주얼 부모이고, 타일은 자신의 콘텐츠에 대해 그렇습니다.

WPF와 Avalonia에서 타일은 **ContentPadding**과 **ShowHeader**로부터 자신의 **Padding**을 계산하므로, XAML에서 설정한 **Padding**은 대체됩니다. 숨겨진 타일의 경우: WPF에서 `Hidden` 타일은 셀을 유지하고 `Collapsed` 타일은 그리드에서 빠지며, Avalonia에서는 `IsVisible`이 false인 타일이 그리드에서 빠집니다.

호스팅된 컨트롤로의 테마 전파

대시보드의 **Theme**이 바뀔 때, 타일이 대시보드에 들어갈 때, 타일에 콘텐츠가 추가될 때, **ApplyTheme**이 호출될 때, 호스팅된 컨트롤은 sgcUICore 도우미 `sgcUIApplyThemeWinForms`, `sgcUIApplyThemeWpf`, `sgcUIApplyThemeAvalonia`를 통해 타일의 테마를 가져옵니다. 이들은 Delphi의 `sgcUIApplyThemeVCL`과 `sgcUIApplyThemeFMX`에 대응합니다. 따라서 타일 안의 sgcCharts 차트, sgcPDF 뷰어, sgcReports 미리 보기는 대시보드와 함께 라이트와 다크를 전환합니다. `utCustom`은 호스팅된 컨트롤을 그대로 둡니다.

레이아웃 JSON은 엔진에 있습니다

TsgcDashboardLayout의 **SaveToJSON**과 **LoadFromJSON**이 레이아웃 텍스트를 쓰고 읽습니다. 세 호스트의 **SaveLayout**과 **LoadLayout** 메서드는 자신의 타일을 모아 이를 호출할 뿐입니다. 이 텍스트는 Delphi 대시보드가 쓰는 것과 같으므로 레이아웃을 Delphi, WinForms, WPF, Avalonia 사이에서 자유롭게 옮길 수 있습니다. 타일의 키는 WinForms에서는 `Control.Name`, WPF에서는 `FrameworkElement.Name`(`x:Name`), Avalonia에서는 `StyledElement.Name`입니다. 호스트가 없는 예제는 .NET 헤드리스 렌더링을 참조하십시오.

WinForms의 Padding과 Layout

TsgcWinFormsDashboard는 `new`를 붙여 두 멤버를 선언합니다. **Padding**은 그리드 주변의 공간(`int`)이며 `Control.Padding`을 숨기고, **Layout** 속성은 그리드 엔진을 반환하며 `Control.Layout` 이벤트를 숨깁니다. 숨겨진 멤버에 접근하려면 대시보드를 `Control`로 캐스팅하십시오.

Avalonia의 Theme

모든 Avalonia 컨트롤에는 이미 **Theme** 속성, 즉 `StyledElement`의 `ControlTheme`이 있습니다. 대시보드, 타일, KPI 카드는 **TsgcUITheme** 형식의 자체 **Theme**을 `new`로 선언합니다. sgc 형식의 참조를 통하면 sgc 테마이고, `Control` 또는 `StyledElement` 형식의 참조를 통하면 Avalonia 컨트롤 테마입니다.

Avalonia에서의 터치 끌기와 팬

세로로 8단위보다 많이 움직이는 터치는 그리드를 이동(팬)합니다. **AllowDrag** 또는 **AllowResize**가 있을 때 타일의 헤더나 그립을 터치하면 즉시 타일 끌기가 시작되고 팬은 일어나지 않습니다. 이는 FireMonkey와 같습니다. 다른 곳을 터치하면 여전히 그리드를 이동합니다. 마우스 끌기는 팬을 하지 않습니다. WPF에서는 터치와 펜이 마우스 입력으로 승격되며, WinForms는 마우스만 사용합니다.

끌기 윤곽선

타일을 끌거나 크기를 조정하는 동안 대상 셀은 강조 색과 25퍼센트 채우기로 표시됩니다. WinForms는 VCL 대시보드처럼 타일 아래에 그립니다. WPF는 어도너 레이어에, Avalonia는 오버레이 비주얼에 그리며, 둘 다 타일 위에 표시됩니다. FireMonkey 자리 표시자는 타일 아래에 있습니다. **Escape**는 모든 호스트에서 끌기를 취소하며, WPF와 Avalonia에서는 캡처를 잃어도 취소됩니다.

스크롤

그리드는 항상 너비에 맞춰지고 세로로 스크롤되며, 모든 호스트에서 휠 한 칸에 반 행씩 이동합니다. WinForms는 `ScrollableControl`로 스크롤합니다. WPF는 그리드를 `ScrollViewer`에 배치합니다. Avalonia는 자체 `ScrollBar`를 표시하고 위치를 **VerticalOffset**으로 노출하며, 이 값이 타일을 위로 옮기고 대시보드 영역으로 잘라냅니다.

이벤트

OnLayoutChanged는 다른 sgc .NET 제품과 마찬가지로 Delphi의 명명된 대리자 형식 **TsgcUINotifyEvent**의 속성입니다. `=`로 할당하며 처리기를 하나만 보유합니다. 브리지 소스는 sgcWebSockets .NET 이벤트를 `+=`로 구독하고 자신의 처리기만 제거하므로, 애플리케이션의 처리기는 계속 실행됩니다. .NET 라이브 피드를 참조하십시오.

디자이너 동사와 지도 타일 없음

Delphi 구성 요소 편집기의 "Add Tile" 동사에는 WinForms 대응 기능이 없습니다. 타일은 코드로 추가하거나, WPF와 Avalonia에서는 XAML로 선언하십시오. 이동한 타일의 디자인 타임 스냅도 이 포트에 포함되지 않습니다. Delphi Suite 데모에는 sgcMaps 타일이 있지만 sgcMaps에는 .NET 포트가 없으므로, .NET Suite 데모에는 차트, PDF 뷰어, 보고서 미리 보기가 있고 지도는 없습니다.

에디션

Community 패키지는 `SGC_COMMUNITY`로 빌드됩니다. 실행 중 끌기 및 크기 조정, **Locked**, 레이아웃 영속성, **OnLayoutChanged**는 컴파일에서 제외되고 브리지 패키지는 만들어지지 않으며, 이는 Delphi의 `SGC_DASHBOARD_PERSIST` 블록과 같습니다. 이를 사용하는 코드는 `#if !SGC_COMMUNITY`로 감싸십시오.

데이터 바인딩

WPF와 Avalonia에서 사용자용 속성은 종속성 속성과 Avalonia 속성이므로 데이터 바인딩, 스타일 지정, 애니메이션이 가능합니다. .NET 데이터 바인딩을 참조하십시오.

참고 항목

- .NET 개요
- .NET 클래스
- .NET 헤드리스 렌더링
- .NET 라이브 피드
- 레이아웃
- 테마
- 영속성

.NET 데이터 바인딩

WPF 및 Avalonia 호스트에서 대시보드, 해당 타일, KPI 카드의 사용자용 속성은 종속성 속성(WPF)과 Avalonia 속성이므로 뷰 모델에 데이터 바인딩하거나 Style로 설정하거나 애니메이션을 적용할 수 있습니다. 이 항목은 WPF와 Avalonia에만 적용됩니다. WinForms 컨트롤은 변경되지 않았으며, 해당 속성은 이전과 같은 일반 CLR 속성입니다.

바인딩 가능한 속성

표의 모든 속성에는 접미사 Property가 붙은 정적 필드가 있습니다. 예를 들어 `TsgcWpfDashboardTile.ColProperty`입니다. WPF에서 이 필드는 DependencyProperty입니다. Avalonia에서는 StyledProperty이며, Col, Row, ColSpan, RowSpan만 DirectProperty입니다. **TwoWay**로 표시된 속성은 런타임 끌기 또는 크기 조정과 LoadLayout이 변경하므로 기본적으로 TwoWay로 바인딩되며, 따라서 Mode가 없는 바인딩도 뷰 모델을 최신 상태로 유지합니다. 나머지 속성은 기본적으로 OneWay로 바인딩됩니다.

WPF 클래스	Avalonia 클래스	바인딩 가능한 속성
TsgcWpfDashboard	TsgcAvaloniaDashboard	Columns, RowHeight, Gap, Padding, AutoFlow, Theme, AllowDrag, AllowResize
TsgcWpfDashboardTile	TsgcAvaloniaDashboardTile	Title, **Col (TwoWay)**, **Row (TwoWay)**, **ColSpan (TwoWay)**, **RowSpan (TwoWay)**, ShowHeader, ContentPadding, ParentTheme, Locked
TsgcWpfKPICard	TsgcAvaloniaKPICard	Title, Value, Icon, Trend, TrendValue, FooterText, AccentColor, MaxPoints

AllowDrag와 AllowResize는 Community 에디션에 없습니다. Padding은 장치 독립 단위로 표현한 그리드 주변의 여백이며 int입니다. Avalonia에서 Theme은 sgcUICore 테마(TsgcUITheme)이며 StyledElement의 Avalonia 컨트롤 테마를 가립니다. 후자는 `StyledElement.ThemeProperty`를 통해 계속 접근할 수 있습니다.

동작

- **끌기는 셀을 다시 기록합니다.** 런타임 끌기 또는 크기 조정과 LoadLayout은 타일의 셀을 변경하며, 타일은 맞춰진 셀을 Col, Row, ColSpan, RowSpan에 복사합니다. 기본 TwoWay 바인딩에서 뷰 모델은 타일이 놓인 셀을 받습니다.
- **바인딩된 셀은 레이아웃 변경입니다.** 데이터 바인딩을 통해 도착한 새 셀은 끌기와 같이 처리되므로 대시보드의 OnLayoutChanged가 발생합니다(Community 에디션 제외). 코드(CLR setter 또는 SetCell)에서 설정하면 셀 속성은 이전과 같이 동작합니다.
- **거부된 값은 다시 기록됩니다.** 대시보드나 타일이 받아들이지 않는 값(1보다 작은 Columns 또는 RowHeight, 음수 Gap 또는 Padding, 1보다 작은 span, 음수 ContentPadding)은 적용되지 않으며, 사용 중인 값이 WPF와 Avalonia 모두에서 속성과 TwoWay 소스에 다시 기록됩니다. KPI 카드가 수정하는 값(1보다 작은 MaxPoints, null 텍스트)에도 같은 규칙이 적용됩니다. WPF에서는 디스패처가 대기 중인 작업을 실행한 뒤에 소스가 사용 중인 값을 받습니다. Avalonia에서 다시 쓰기는 바인딩이 실행 중일 때 소스가 보낸 값에 적용됩니다. 바인딩이 시작될 때 소스가 이미 가지고 있는 값은 컨트롤에서 수정되지만 소스에는 다시 기록되지 않습니다.
- **LoadLayout은 속성을 업데이트합니다.** LoadLayout 후 그리드 치수와 타일의 셀이 속성으로 다시 복사되므로 바인딩은 복원된 레이아웃을 봅니다.
- **스타일.** Style setter는 바인딩 가능한 모든 속성을 설정할 수 있습니다. Style로 설정한 대시보드의 Theme은 ParentTheme이 true인 모든 타일에 적용됩니다.

CLR 속성으로 남는 속성

속성	이유
Tiles (대시보드)	타일 컬렉션이며 XAML에 선언된 타일로 채워집니다. 바인딩할 값이 아닙니다.

Layout (대시보드)	레이아웃 엔진 개체이며 읽기 전용입니다. 그리드 치수는 Columns, RowHeight, Gap, Padding, AutoFlow를 통해 바인딩합니다.
VerticalOffset (Avalonia 대시보드)	마지막 레이아웃 패스의 스크롤 위치이며 설정이 아닙니다.
Theme (타일)	ParentTheme이 true인 동안 타일은 대시보드의 테마를 읽으며, Theme을 설정하면 ParentTheme이 지워집니다. 대신 대시보드의 Theme 또는 ParentTheme을 바인딩하십시오.
Theme (KPI 카드)	카드를 호스팅하는 타일이 설정합니다.

예제

아래 뷰 모델은 두 예제에서 공유됩니다. Revenue는 KPI 카드의 Value에 바인딩되고, RevenueCol과 RevenueRow는 Mode 없이 해당 타일의 Col과 Row에 바인딩되므로 TwoWay로 바인딩됩니다. 사용자가 타일을 끌면 뷰 모델이 새 셀을 받습니다. 사용자가 타일을 끌 수 있게 하려면 대시보드의 AllowDrag를 True로 설정하십시오.

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace DashboardBinding
{
    public class MainViewModel : INotifyPropertyChanged
    {
        private string FRevenue = "12,400";
        private int FRevenueCol;
        private int FRevenueRow;

        public event PropertyChangedEventHandler PropertyChanged;

        public string Revenue
        {
            get { return FRevenue; }
            set { DoSet(ref FRevenue, value); }
        }

        // bound to Col and Row of the tile, TwoWay by default
        public int RevenueCol
        {
            get { return FRevenueCol; }
            set { DoSet(ref FRevenueCol, value); }
        }

        public int RevenueRow
        {
            get { return FRevenueRow; }
            set { DoSet(ref FRevenueRow, value); }
        }

        private void DoSet<T>(ref T aField, T aValue, [CallerMemberName] string aName = null)
        {
            if (EqualityComparer<T>.Default.Equals(aField, aValue))
                return;
            aField = aValue;
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(aName));
        }
    }
}
```

WPF

MainWindow.xaml은 뷰 모델을 창의 DataContext로 만듭니다. 코드 비하인드와 App.xaml은 WPF 템플릿의 것입니다.

```
<Window x:Class="DashboardBinding.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:DashboardBinding"
        xmlns:dash="clr-namespace:esegece.sgcDashboard;assembly=esegece.sgcDashboard.Wpf"
        Title="sgcDashboard binding" Width="900" Height="560">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <TextBlock DockPanel.Dock="Top" Margin="8" Text="{Binding RevenueCol}" />
        <dash:TsgcWpfDashboard x:Name="Dashboard" Columns="12" RowHeight="110">
            <!-- Col and Row bind TwoWay by default: a drag writes the snapped cell back -->
            <dash:TsgcWpfDashboardTile x:Name="tileRevenue" Title="Revenue" ColSpan="6" RowSpan="2"
                                       Col="{Binding RevenueCol}" Row="{Binding RevenueRow}">
                <dash:TsgcWpfKPICard x:Name="kpiRevenue" Title="Revenue" Icon="$" Value="{Binding Revenue}" />
            </dash:TsgcWpfDashboardTile>
            <dash:TsgcWpfDashboardTile x:Name="tileUsers" Title="Users" Col="0" Row="2" ColSpan="6" RowSpan="2">
                <dash:TsgcWpfKPICard x:Name="kpiUsers" Title="Active users" Icon="U" Value="340" />
            </dash:TsgcWpfDashboardTile>
        </dash:TsgcWpfDashboard>
    </DockPanel>
</Window>
```

Avalonia

MainWindow.axaml은 `x:DataType`을 뷰 모델로 지정하므로 바인딩은 컴파일된 바인딩입니다. 코드 비하인드는 InitializeComponent를 호출하고 App 클래스는 Avalonia 빠른 시작의 것입니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:DashboardBinding"
        xmlns:dash="using:esegece.sgcDashboard"
        x:Class="DashboardBinding.MainWindow"
        x:DataType="local:MainViewModel"
        Title="sgcDashboard binding" Width="900" Height="560">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <TextBlock DockPanel.Dock="Top" Margin="8" Text="{Binding RevenueCol}" />
        <dash:TsgcAvaloniaDashboard x:Name="Dashboard" Columns="12" RowHeight="110">
            <!-- Col and Row bind TwoWay by default: a drag writes the snapped cell back -->
            <dash:TsgcAvaloniaDashboardTile x:Name="tileRevenue" Title="Revenue" ColSpan="6" RowSpan="2"
                                            Col="{Binding RevenueCol}" Row="{Binding RevenueRow}">
                <dash:TsgcAvaloniaKPICard x:Name="kpiRevenue" Title="Revenue" Icon="$" Value="{Binding Revenue}" />
            </dash:TsgcAvaloniaDashboardTile>
            <dash:TsgcAvaloniaDashboardTile x:Name="tileUsers" Title="Users" Col="0" Row="2" ColSpan="6" RowSpan="2">
                <dash:TsgcAvaloniaKPICard x:Name="kpiUsers" Title="Active users" Icon="U" Value="340" />
            </dash:TsgcAvaloniaDashboardTile>
        </dash:TsgcAvaloniaDashboard>
    </DockPanel>
</Window>
```

참고 항목

- Delphi와 .NET의 차이점
- .NET 클래스
- WPF 빠른 시작
- Avalonia 빠른 시작

.NET 헤드리스 렌더링

레이아웃 엔진과 KPI 카드는 UI 프레임워크가 필요 없으므로, 서버, 콘솔 애플리케이션, 테스트에서 타일을 배치하고 카드를 이미지 파일로 렌더링할 수 있습니다. 헤드리스 KPI 이미지의 .NET 대응 기능입니다.

메서드

메서드	패키지	설명
TsgcDashboardKPI.SaveToSVG(stream 또는 file, width, height, scale)	esegece.sgcDashboard	카드를 SVG 텍스트로 씁니다. Delphi 카드의 SVG와 바이트 단위로 같습니다.
TsgcDashboardKPI.SaveToPNG(stream 또는 file, width, height, scale)	esegece.sgcDashboard.Rendering	투명한 배경의 PNG를 쓰는 확장 메서드입니다. 이미지는 width 곱하기 scale, height 곱하기 scale 픽셀입니다.
TsgcDashboardKPI.Render(canvas, width, height, scale)	esegece.sgcDashboard	카드를 모든 sgcUICore 캔버스에 그립니다.
TsgcDashboardLayout.Arrange(width, scale)	esegece.sgcDashboard	항목을 배치합니다. 결과는 **PlacedCell**, **Rect**, **ContentHeight**가 반환합니다.
TsgcDashboardLayout.SaveToJSON, LoadFromJSON	esegece.sgcDashboard	대시보드의 레이아웃 텍스트. Registered 에디션.

예제

```
using System;
using System.Collections.Generic;
using esegece.sgcDashboard;

namespace DashboardHeadless
{
    internal static class Program
    {
        private static int Main()
        {
            // a KPI card without any UI framework
            TsgcDashboardKPI oKPI = new TsgcDashboardKPI();
            oKPI.Title = "Revenue";
            oKPI.Icon = "$";
            oKPI.Value = "12,480";
            oKPI.Trend = TsgcDashboardKPITrend.ktUp;
            oKPI.TrendValue = "+4.2%";
            oKPI.FooterText = "Today, USD";
            double[] vHistory = { 11200, 11650, 11400, 11900, 12100, 11980, 12480 };
            foreach (double vValue in vHistory)
                oKPI.AddValue(vValue);

            // SVG from the core package, PNG from esegece.sgcDashboard.Rendering
            oKPI.SaveToSVG("revenue.svg", 280, 200);
            oKPI.SaveToPNG("revenue.png", 280, 200, 2);
```

```
            // the layout engine places tiles on the grid without a host
            TsgcDashboardLayout oLayout = new TsgcDashboardLayout();
            oLayout.Columns = 12;
            oLayout.RowHeight = 100;
            oLayout.AddItem("revenue", sgcDashboard_Layout.sgcDashboardCell(0, 0, 4, 2));
            oLayout.AddItem("orders", sgcDashboard_Layout.sgcDashboardCell(2, 0, 4, 2));
            oLayout.Arrange(1200, 1);
            for (int i = 0; i < oLayout.Count; i++)
            {
                TsgcDashboardCell vCell = oLayout.PlacedCell(i);
                Console.WriteLine(oLayout.Keys(i) + ": column " + vCell.Col + ", row " + vCell.Row);
            }

            // the layout JSON lives in the engine, the hosts only add their tiles
            List<TsgcDashboardLayoutTile> oTiles = new List<TsgcDashboardLayoutTile>();
            oTiles.Add(new TsgcDashboardLayoutTile("revenue", oLayout.Cell(0), true));
            oTiles.Add(new TsgcDashboardLayoutTile("orders", oLayout.Cell(1), true));
            string vJSON = oLayout.SaveToJSON(oTiles);
            Console.WriteLine(vJSON);
            oTiles[1].Cell = sgcDashboard_Layout.sgcDashboardCell(0, 4, 12, 1);
            oLayout.LoadFromJSON(vJSON, oTiles);
            Console.WriteLine("orders back at column " + oTiles[1].Cell.Col + ", row " + oTiles[1].Cell.Row);
            return oLayout.SaveToJSON(oTiles) == vJSON ? 0 : 1;
        }
    }
}
```

프로그램은 560 x 400 픽셀의 revenue.svg와 revenue.png를 쓰고 다음을 출력합니다.

```
revenue: column 0, row 0
orders: column 2, row 2
{"version":1,"columns":12,"rowHeight":100,"gap":12,"padding":12,"tiles":[{"name":"revenue","col":0,"row":0,"colSpan":4,"row
orders back at column 2, row 0
```

orders 타일은 2열, 0행을 요청하는데, 이는 revenue 타일과 겹치므로 엔진이 2행으로 아래로 밉니다. **Cell**은 타일이 요청한 셀을, **PlacedCell**은 실제로 받은 셀을 반환하며, JSON에는 Delphi와 마찬가지로 요청한 셀이 저장됩니다.
헤드리스 패키지는 Windows, Linux, macOS에서 실행됩니다. Linux에서는 esegece.sgcDashboard.Rendering이 Linux 용 SkiaSharp 네이티브 자산을 함께 설치합니다.

참고 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- 헤드리스 KPI 이미지
- TsgcDashboardKPI
- 레이아웃

.NET 라이브 피드

esegece.sgcDashboard.sgcWebSockets 패키지는 Delphi sgcDashboardWS 브리지처럼 sgcWebSockets .NET의 라이브 연결에서 KPI 카드에 데이터를 공급합니다. 실시간 데이터를 참조하십시오. 소스는 모든 JSON 메시지에서 숫자를 읽어 카드에 전달합니다. 스파크라인에 점이 추가되고 **Value**, **Title**, **TrendValue**, **Trend**가 갱신됩니다. **Registered 에디션 전용**: 브리지 패키지는 Community 에디션에 포함되지 않습니다.

소스

소스	클라이언트	읽는 대상
TsgcDashboardSource_WebSocket	TsgcWebSocketClient	모든 텍스트 메시지. **OnMessage**를 통해 읽습니다.
TsgcDashboardSource_MQTT	TsgcWSProtocol_MQTT_Client	토픽이 **Topic**과 일치하는 모든 publish. MQTT 와일드카드 `+`와 `#`를 사용할 수 있으며 **OnMQTTPublish**를 통해 읽습니다. 소스는 구독하지 않으며, 애플리케이션이 클라이언트를 통해 구독합니다.
TsgcDashboardSource_SSE	**Client**의 TsgcHTTP1Client 또는 **SSEClient**의 TsgcSSEClient	모든 서버 전송 이벤트의 데이터. **GetSSE**와 **OnSSEMessage**의 청크에서, 또는 **OnEvent**의 이벤트에서 가져옵니다. **GetSSE**는 스트림이 끝날 때까지 차단되므로 작업자 스레드에서 호출하십시오.

모든 소스는 TsgcDashboardSource의 멤버를 공유합니다.

멤버	설명
카드	KPI 카드 컨트롤(모든 **IsgcDashboardKPIHost**) 또는 헤드리스 **TsgcDashboardKPI**. 다른 개체는 **EsgcDashboardSource**를 발생시킵니다.
ValuePath	숫자의 JSON 경로. 예: `price` 또는 `data.items[0].value`. 숫자 또는 숫자 문자열을 읽으며 소수점은 마침표를 사용합니다. 비어 있으면 메시지 전체를 숫자로 읽습니다.
TitlePath, TrendValuePath	설정하면 해당 JSON 값을 카드의 **Title**과 **TrendValue**에 복사합니다.
ValueFormat	float 인수 하나를 받는 Delphi Format 문자열. 예: `%.2f` 또는 `%.1f ms`. 현재 문화권으로 서식이 지정됩니다. 비어 있으면 Delphi FloatToStr처럼 숫자를 씁니다.
Active	기본값은 True입니다. False이면 모든 메시지를 무시합니다.
OnValue	읽은 숫자마다 카드가 갱신된 후에 호출됩니다.

스레딩

점은 메시지가 도착한 스레드에서 즉시 스파크라인에 추가됩니다. 텍스트와 추세(소스의 이전 값과 비교한 상승, 하락, 중립)는 소스가 만들어질 때의 현재 `SynchronizationContext`, 즉 WinForms, WPF, Avalonia 애플리케이션의 UI 스레드를 통해 적용되므로 클라이언트의 모든 **NotifyEvents** 모드에서 안전합니다. 동기화 컨텍스트 없이 만들어진 소스(콘솔 애플리케이션이나 서비스)는 메시지의 스레드에서 갱신을 적용합니다.

예제

WinForms 대시보드의 KPI 카드 세 개에 WebSocket 시세 스트림, MQTT 센서 토픽, 서버 전송 이벤트 피드로 데이터를 공급합니다. 호스트 이름은 사용자의 서버로 바꿔야 하는 자리 표시자입니다.

```
using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
using esegece.sgcDashboard;
using esegece.sgcWebSockets;

namespace DashboardLiveFeeds
{
    public class MainForm : Form
    {
        private readonly TsgcWinFormsDashboard Dashboard;
        private readonly TsgcWebSocketClient WSClient;
        private readonly TsgcWebSocketClient MQTTConnection;
        private readonly TsgcWSProtocol_MQTT_Client MQTT;
        private readonly TsgcHTTP1Client HTTP;
        private readonly TsgcDashboardSource_WebSocket srcPrice;
        private readonly TsgcDashboardSource_MQTT srcTemperature;
        private readonly TsgcDashboardSource_SSE srcLatency;

        public MainForm()
        {
            Text = "sgcDashboard live feeds";
            ClientSize = new Size(900, 300);
            Dashboard = new TsgcWinFormsDashboard();
            Dashboard.Dock = DockStyle.Fill;
            Controls.Add(Dashboard);
            TsgcWinFormsKPICard kpiPrice = AddCard("Price", 0);
            TsgcWinFormsKPICard kpiTemperature = AddCard("Temperature", 4);
            TsgcWinFormsKPICard kpiLatency = AddCard("Latency", 8);

            // WebSocket: every text message is JSON, {"symbol":"EURUSD","price":1.0843}
            WSClient = new TsgcWebSocketClient();
            WSClient.Host = "quotes.example.com";
            WSClient.Port = 80;
            srcPrice = new TsgcDashboardSource_WebSocket();
            srcPrice.Card = kpiPrice;
            srcPrice.ValuePath = "price";
            srcPrice.TitlePath = "symbol";
            srcPrice.ValueFormat = "%.4f";
            srcPrice.Client = WSClient;

            // MQTT: the source reads the publishes, the application subscribes
            MQTTConnection = new TsgcWebSocketClient();
            MQTTConnection.Host = "broker.example.com";
            MQTTConnection.Port = 8080;
            MQTT = new TsgcWSProtocol_MQTT_Client();
            MQTT.Client = MQTTConnection;
            MQTT.OnMQTTConnect += (Connection, Session, ReasonCode, ReasonName, ConnectProperties) =>
                MQTT.Subscribe("sensors/+/temperature");
            srcTemperature = new TsgcDashboardSource_MQTT();
            srcTemperature.Card = kpiTemperature;
            srcTemperature.Topic = "sensors/+/temperature";
            srcTemperature.ValuePath = "celsius";
            srcTemperature.ValueFormat = "%.1f C";
            srcTemperature.Client = MQTT;

            // SSE: the data of every event is JSON, {"ms":42}
            HTTP = new TsgcHTTP1Client();
            srcLatency = new TsgcDashboardSource_SSE();
            srcLatency.Card = kpiLatency;
            srcLatency.ValuePath = "ms";
```

```
            srcLatency.ValueFormat = "%.0f ms";
            srcLatency.Client = HTTP;
        }

        private TsgcWinFormsKPICard AddCard(string aTitle, int aCol)
        {
            TsgcWinFormsDashboardTile oTile = new TsgcWinFormsDashboardTile();
            oTile.Title = aTitle;
            oTile.SetCell(sgcDashboard_Layout.sgcDashboardCell(aCol, 0, 4, 2));
            TsgcWinFormsKPICard oCard = new TsgcWinFormsKPICard();
            oCard.Title = aTitle;
            oCard.Dock = DockStyle.Fill;
            oTile.Controls.Add(oCard);
            Dashboard.Controls.Add(oTile);
            return oCard;
        }

        protected override void OnShown(EventArgs e)
        {
            base.OnShown(e);
            WSClient.Active = true;
            MQTTConnection.Active = true;
            // GetSSE blocks until the stream ends, so it runs in a worker thread
            Thread oSSE = new Thread(() => HTTP.GetSSE("https://status.example.com/latency"));
            oSSE.IsBackground = true;
            oSSE.Start();
        }

        protected override void OnFormClosed(FormClosedEventArgs e)
        {
            srcPrice.Dispose();
            srcTemperature.Dispose();
            srcLatency.Dispose();
            WSClient.Active = false;
            MQTTConnection.Active = false;
            base.OnFormClosed(e);
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new MainForm());
        }
    }
}
```

루프백 인터페이스에서 사용해 보기

이 콘솔 프로그램은 같은 프로세스의 루프백 인터페이스에서 **TsgcWebSocketServer**를 시작하고 **TsgcWebSocketClient**를 연결한 다음, 브로드캐스트 메시지 세 개로 헤드리스 카드에 데이터를 공급하므로 네트워크가 필요 없습니다.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using esegece.sgcDashboard;
using esegece.sgcWebSockets;

namespace DashboardLiveLoopback
{
    internal static class Program
```

```
    {
        private static int FreePort()
        {
            TcpListener oListener = new TcpListener(IPAddress.Loopback, 0);
            oListener.Start();
            try
            {
                return ((IPEndPoint)oListener.LocalEndpoint).Port;
            }
            finally
            {
                oListener.Stop();
            }
        }

        private static int Main()
        {
            // a server in the same process, on the loopback interface only
            int vPort = FreePort();
            TsgcWebSocketServer oServer = new TsgcWebSocketServer();
            oServer.NotifyEvents = TwsNotifyEvent.neNoSync;
            oServer.Port = vPort;
            oServer.Bindings.Clear();
            oServer.Bindings.NewBinding("127.0.0.1", vPort);
            oServer.Active = true;

            TsgcWebSocketClient oClient = new TsgcWebSocketClient();
            oClient.NotifyEvents = TwsNotifyEvent.neNoSync;
            oClient.Host = "127.0.0.1";
            oClient.Port = vPort;

            // a headless card; in an application this is a KPI card control
            TsgcDashboardKPI oCard = new TsgcDashboardKPI();
            oCard.Title = "EUR/USD";
            CountdownEvent oReceived = new CountdownEvent(3);
            TsgcDashboardSource_WebSocket oSource = new TsgcDashboardSource_WebSocket();
            oSource.Card = oCard;
            oSource.ValuePath = "quote.price";
            oSource.ValueFormat = "%.4f";
            oSource.OnValue = (Sender, aValue) =>
            {
                if (!oReceived.IsSet)
                    oReceived.Signal();
            };
            oSource.Client = oClient;
            try
            {
                if (!oClient.Connect(5000))
                    return 1;
                Thread.Sleep(200);
                foreach (string vPrice in new[] { "1.0841", "1.0848", "1.0839" })
                    oServer.Broadcast("{\"quote\":{\"symbol\":\"EURUSD\",\"price\":" + vPrice + "}}");
                bool vDone = oReceived.Wait(5000);
                Console.WriteLine(oCard.Title + " " + oCard.Value + ", trend " + oCard.Trend + ", " +
                    oCard.ValueCount + " points");
                return vDone ? 0 : 1;
            }
            finally
            {
                oSource.Dispose();
                oClient.Active = false;
                oServer.Active = false;
            }
        }
    }
}
```

현재 문화권으로 서식이 지정된 값(여기서는 소수점에 쉼표 사용), 이전 값과 비교한 추세, 스파크라인의 점 수를 출력합니다.

```
EUR/USD 1,0839, trend ktDown, 3 points
```

Delphi와의 차이점

- **OnMessage**, **OnMQTTPublish**, **OnSSEMessage**, **OnEvent**는 .NET 이벤트입니다. 소스는 `+=`로 자신의 처리기를 추가하고 자신의 처리기만 제거하는 반면, Delphi는 애플리케이션의 처리기를 저장하고 대체한 뒤 복원합니다. 애플리케이션의 처리기는 항상 구독 순서대로 실행됩니다.
- .NET에는 FreeNotification이 없습니다. 카드가 WinForms 컨트롤처럼 `IComponent`이면 카드가 해제될 때 소스가 **Card**를 지웁니다. 다른 카드는 카드가 사라지기 전에 **Card**를 null로 설정하십시오.
- SSE 소스의 **SSEClient**는 **TsgcSSEClient**를 받으며 .NET에서 추가된 기능입니다.

참고 항목

- .NET 개요
- NuGet에서 설치
- Delphi와 .NET의 차이점
- 실시간 데이터
- TsgcDashboardSource
- TsgcDashboardSource_WebSocket
- TsgcDashboardSource_MQTT
- TsgcDashboardSource_SSE
- .NET 데모

.NET 데모

이 .NET 데모는 제품의 `demos` 폴더와 데모 zip에 솔루션 `sgcDashboard_Demos.sln`과 함께 들어 있습니다. 각 데모에는 README.md가 있습니다.

데모	호스트	설명
Basic	WinForms	12열 그리드의 타일 여섯 개: 랜덤 워크로 타이머가 데이터를 공급하는 **TsgcWinFormsKPICard** 네 개, 일반 컨트롤로 만든 이벤트 로그와 서버 목록, 대시보드의 라이트와 다크를 전환하는 Theme 콤보 상자.
BasicWpf	WPF	**TsgcWpfDashboard**에 XAML로 선언한 같은 대시보드.
BasicAvalonia	Avalonia	**TsgcAvaloniaDashboard**로 만든 같은 대시보드이며 Windows, Linux, macOS에서 실행됩니다.
Suite	WinForms	하나의 대시보드에 모은 eSeGeCe 제품: KPI 카드 두 개, 라이브 선을 표시하는 sgcCharts 차트, 타일 안의 sgcPDF 뷰어와 sgcReports 미리 보기, 라이트와 다크 전환 스위치. sgcMaps에는 .NET 포트가 없으므로 지도 타일은 없습니다.
LiveKPI	WinForms	같은 프로세스의 빈 루프백 포트에서 실행되는 **TsgcWebSocketServer**로부터 **TsgcDashboardSource_WebSocket**이 데이터를 공급하는 KPI 카드 세 개. Registered 에디션 전용입니다.
SaveLayout	WinForms	타일을 끌고 크기를 조정한 다음, 레이아웃을 layout.json에 저장하고 다시 로드하거나 초기화합니다. Registered 에디션 전용입니다.

Community 패키지에서도 LiveKPI와 SaveLayout은 빌드됩니다. Registered 코드는 컴파일에서 제외되고 제한 사항은 상태 표시줄에 표시됩니다.

빌드 및 실행

데모에는 .NET 8 SDK 이상이 필요합니다. 데모의 폴더에서 실행합니다.

```
dotnet run
```

제품 트리 안에서는 데모가 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip에서는 NuGet 패키지를 복원하며 기본값은 Community 에디션입니다. Registered 패키지를 사용하려면 **SgcDashboardEdition**을 설정합니다.

```
dotnet build -p:SgcDashboardEdition=Registered
```

자체 검사

모든 데모에는 창을 화면 밖에서 열고 구동하는 `--test` 스위치가 있습니다. Basic은 타일과 셀을 확인하고 카드에 데이터를 공급하며 테마를 전환합니다. Suite는 타일 안의 차트, PDF 뷰어, 보고서 미리 보기를 확인합니다. LiveKPI는 모든 카드가 값을 받았는지 확인합니다. SaveLayout은 저장하고, 타일을 이동하고, 로드하고, 초기화합니다. 데모는 검사마다 PASS 또는 FAIL 한 줄을 파일에 쓰고 실패한 검사의 수를 종료 코드로 하여 종료하므로, 0은 모든 검사가 통과했음을 뜻합니다.

```
dotnet run -- --test result.txt
```

참고 항목

- .NET 개요
- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작

- Avalonia 빠른 시작
- .NET 라이브 피드
- Delphi 데모

문제 해결

SaveLayout, AllowDrag 또는 OnLayoutChanged에서 Undeclared identifier 오류가 발생합니다

Community 에디션은 SGC_DASHBOARD_PERSIST 없이 컴파일되므로 AllowDrag, AllowResize, OnLayoutChanged, SaveLayout 및 LoadLayout 메서드, VCL Locked 속성이 존재하지 않으며, 이를 사용하는 코드는 E2003 Undeclared identifier 오류로 실패합니다. .dfm에 AllowDrag 또는 AllowResize가 있는 상태로 Registered 빌드에서 저장한 폼은 "Property AllowDrag does not exist"와 함께 "Error reading" 예외를 발생시킵니다. 해당 멤버를 제거하거나 Registered 또는 Trial 빌드를 사용하십시오. 에디션을 참조하십시오.

Unit sgcDashboardWS_Source_WebSocket 또는 sgcWebSocket을 찾을 수 없습니다

실시간 데이터 브리지는 sgcDashboard와 sgcWebSockets가 필요한 별도의 패키지 sgcDashboardWS입니다. 먼저 같은 IDE용 sgcWebSockets를 설치한 다음 sgcDashboardWS 패키지를 설치하고, sgcWebSockets 소스 또는 라이브러리 폴더를 검색 경로에 추가하십시오. 브리지는 Community 에디션에 포함되어 있지 않습니다.

브리지 소스가 공급하는 KPI 카드가 콘솔 애플리케이션이나 서비스에서 변경되지 않습니다

Delphi 2010 이상에서는 카드 텍스트가 TThread.Queue를 통해 메인 스레드에서 적용됩니다. 콘솔 애플리케이션이나 서비스에는 큐를 실행하는 메시지 루프가 없으므로 CheckSynchronize를 정기적으로 호출해야 합니다. 스파크라인 값은 어떤 경우에도 즉시 추가됩니다.

카드의 Value가 소수점 구분 기호로 쉼표를 표시합니다

ValueFormat은 애플리케이션 로캘의 소수점 구분 기호를 사용하는 Format으로 적용됩니다. JSON 숫자 자체는 항상 점으로 읽습니다. ValueFormat을 변경하거나, OnValue에서 직접 서식을 지정하여 값을 설정하십시오.

FMX 대시보드가 Linux64용으로 컴파일되지 않습니다

Linux의 FMX 컨트롤에는 IDE에 FMX for Linux(FMXLinux)가 설치되어 있어야 합니다. 설치되어 있지 않으면 Linux64에서는 프레임워크 중립 유닛인 레이아웃 엔진과, 헤드리스 PNG 및 SVG 출력을 갖춘 KPI 모델만 빌드됩니다. 플랫폼을 참조하십시오.

테마를 변경해도 호스팅된 컨트롤이 자체 색상을 유지합니다

테마는 게시된 Theme, Chart.Theme, BackgroundColor 또는 Color를 통해서만 호스팅된 컨트롤에 전달됩니다. sgcReports 미리 보기는 이들을 하나도 게시하지 않아 자체 색상을 유지하며, 타일의 패널 안에 중첩된 컨트롤에는 테마가 적용되지 않습니다. utCustom에서도 Theme은 무시됩니다. 테마를 참조하십시오.

Suite 또는 LiveKPI 데모가 컴파일되지 않습니다

Suite 데모에는 sgcCharts, sgcMaps, sgcPDF, sgcReports 소스가 검색 경로에 있어야 하고, LiveKPI에는 sgcWebSockets 소스가 필요합니다. 데모를 참조하십시오.

소개

sgcGrid는 VCL과 FireMonkey용 Delphi 및 C++Builder 실시간 데이터 그리드입니다. 계속 변하는 데이터를 위해 만들어졌습니다. 행은 어떤 스레드에서든 키로 갱신되고, 변경된 셀은 녹색이나 빨간색으로 깜박이다 서서히 사라지며, 연속된 갱신은 타이머 기반 다시 그리기로 합쳐지고, 테일 모드는 로그나 피드의 최신 행을 계속 화면에 유지합니다. **TsgcVCLGrid**와 **TsgcFMXGrid**는 여러 열 정렬, 필터링, 행 또는 셀 선택, 열 크기 조정, 조건부 서식 규칙 적용, 셀 제자리 편집을 지원합니다.
sgcGrid는 밴드, 그룹화, 피벗을 갖춘 범용 데이터베이스 그리드가 아닙니다. 처리량을 앞세운 전문 그리드로, 보이는 행만 그리며 행이 100,000개인 그리드의 1080p 프레임 하나를 몇 밀리초 만에 그립니다. 성능을 참조하십시오.
sgcGrid는 eSeGeCe 비주얼 제품군의 일부입니다. 캔버스, 래스터라이저, 라이트 및 다크 테마를 갖춘 공유 런타임인 **sgcUICore**로 그려집니다. 데이터는 sgcUICore 테이블 인터페이스 IsgcUITable을 구현하므로, **sgcCharts** 차트가 TsgcChartDataSource_Table로 그리드의 행을 그릴 수 있으며, **sgcDashboard** 타일에 배치한 그리드는 대시보드의 테마를 따릅니다.

Symbol	Price ▼	Change %	Volume	Halted
MSFT	412.50	+1.25	1,200,500	
AAPL	228.10	-0.84	3,400,200	
AMZN	186.40	-2.10	2,250,000	✓
GOOG	171.90	+0.37	980,100	
NVDA	121.70	+3.02	**5,120,750**	

에디션

sgcGrid는 Registered, Trial, Community로 제공됩니다. Community 에디션에는 제자리 편집(SGC_GRID_EDIT)과, 실시간 피드 및 AI 쿼리를 포함한 sgcGridWS 브리지가 빠져 있습니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 프레임워크에 독립적인 모델, sgcUICore 캔버스 위의 렌더러, VCL 및 FMX 컨트롤, 스레딩 모델.
- 플랫폼과 성능: Delphi 버전, C++Builder, 플랫폼, 프레임 예산.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder: 몇 분 만에 첫 그리드 만들기.
- 구성 요소: 그리드, 모델 클래스, 데이터 소스, AI 쿼리.
- 실시간 갱신과 스레딩, 정렬, 필터링, 선택과 키보드, 열 크기 조정.
- 변경 시 깜박임, 조건부 서식, 제자리 편집, 가상 모드.
- JSON과 CSV, DataSet 바인딩, 실시간 피드, 그리드에 질문하기, 그리드에서 차트 만들기.
- 테마와 헤드리스 렌더링.
- 데모: Delphi 및 C++Builder 데모 프로젝트.

아키텍처

sgcGrid는 계층으로 구성됩니다. 각 계층은 자신보다 아래 계층에만 의존하므로, 모델, 렌더러, 가져오기 및 내보내기 유닛은 VCL이나 FireMonkey 없이 컴파일되며 Linux64에서도 테스트됩니다.

계층	유닛	역할
모델	sgcGrid_Columns, sgcGrid_Data	TsgcGridData는 열과 행을 보관합니다. 열마다 Variant 하나, 셀마다 변경 스탬프와 변경 방향이 있으며 행 키로 인덱싱됩니다. 정렬, 필터링을 하고 가상 모드를 지원합니다. 스레드 안전하며 IsgcUITable을 구현합니다.
줄 뷰	sgcGrid_Lines, sgcGrid_Group, sgcGrid_Summary	TsgcGridLineView는 그리드의 표시 줄(그룹 행과 뷰 행)을 데이터의 뷰 행에 대응시키고, TsgcGridGrouper는 그룹을 만들며, sgcGrid_Summary는 그룹과 푸터 요약을 계산합니다. Registered 에디션에서는 sgcGrid_Tree와 sgcGrid_Detail이 트리 행과 디테일 밴드를 추가합니다.
서식	sgcGrid_Format	변경 시 깜박임과 조건부 서식 규칙이며, 모든 렌더러가 공유합니다.
렌더러	sgcGrid_Render	TsgcGridRenderer는 TsgcGridData를 임의의 sgcUICore 캔버스에 그립니다. 헤더, 행, 격자선, 선택, 정렬 글리프, 체크 표시, 셀 스타일을 그리며 보이는 행만 그립니다. 창 없이 PNG와 SVG 파일도 씁니다.
가져오기 및 내보내기	sgcGrid_IO, sgcGrid_Source_DataSet	JSON과 CSV 로드 및 저장, 그리고 임의의 TDataSet으로 그리드를 채우는 TsgcGridSource_DataSet.
Excel 내보내기	sgcGrid_Zip, sgcGrid_XLSX	SaveToXLSX 뒤에 있는 XLSX 라이터이며 Registered 에디션 전용입니다. Excel 내보내기를 참조하십시오.
VCL 컨트롤	sgcGrid_VCL	TsgcVCLGrid는 TsgcGridData와 렌더러를 소유하며 스크롤 막대, 마우스, 키보드, 클립보드, 제자리 편집기를 추가합니다.
FMX 컨트롤	sgcGrid_FMX	TsgcFMXGrid는 동일한 속성, 메서드, 이벤트를 가진 FireMonkey 컨트롤이며, 직접 그린 스크롤 막대와 터치 패닝을 제공합니다.
브리지	sgcGridWS_*	WebSocket, MQTT, SSE 피드 소스와 TsgcGridAIQuery입니다. sgcWebSockets가 필요합니다.
sgcUICore	sgcUICore_Canvas, sgcUICore_VCL, sgcUICore_FMX, sgcUICore_Theme, sgcUICore_Table	공유 런타임입니다. 벡터 캔버스와 래스터라이저, 프레임 조절(MaxFPS)을 갖춘 VCL 및 FMX 컨트롤 기반, 테마 토큰, IsgcUITable 인터페이스로 구성됩니다.

스레딩 모델

TsgcGridData의 모든 공개 읽기 및 쓰기 메서드는 내부 재진입 가능 크리티컬 섹션을 획득하므로, 피드 스레드가 Synchronize 없이 UpdateRow, AppendRow, UpdateCell, DeleteRow를 직접 호출할 수 있습니다. 데이터는 잠금이 해제된 후 쓰기 스레드에서 OnChange를 발생시킵니다. 그리드는 그곳에서 그리지 않고 내용을 dirty로 표시하며, sgcUICore 컨트롤 기반이 초당 최대 **MaxFPS**회(기본값 60)로 메인 스레드에서 다시 그리기를 예약합니다. 새 행이나 재정렬 같은 뷰 변경은 메인 스레드에 한 번 게시되며, 그곳에서 스크롤 범위, 테일 위치, 키로 다시 찾은 선택 행이 갱신됩니다.
BeginUpdate와 EndUpdate는 여러 변경을 하나의 OnChange로 합칩니다. 열은 메인 스레드에서 변경합니다. 쓰기 스레드가 실행 중일 때는 열 변경을 Lock과 Unlock으로 감싸십시오. 실시간 갱신과 스레딩을 참조하십시오.

뷰와 저장소

행은 삽입 순서대로 저장됩니다. 정렬 열과 필터가 없으면 그리드는 저장소를 그대로 표시합니다. 그렇지 않으면 변경 후 첫 읽기 때 뷰가 지연 방식으로 다시 만들어집니다. 먼저 필터를 적용한 다음 안정적인 다중 열 정렬을 수행합니다. 필터가 없을 때 정렬 대상이 아닌 열이 변경되면 현재 뷰가 유지되므로, 계속 바뀌는 가격 열이 재정렬 비용을 유발하지 않습니다.

줄 뷰

그리드는 뷰 행이 아니라 표시 줄을 그리고 스크롤합니다. 각 그리드는 데이터에 대한 TsgcGridLineView를 하나 소유합니다. 그룹화한 열이 없으면 줄 뷰는 항등 변환이며 줄 i는 뷰 행 i이고 아무것도 만들어지지 않습니다. 그룹화와, Registered 에디션에서는 트리 행과 디테일 밴드는 줄 뷰의 뷰 상태입니다. 데이터, 그 뷰 행, IsgcUITable은 평평하게 유지되므로 차트, 피벗 그리드, 브리지는 이전과 같은 행을 봅니다. 줄 뷰는 데이터의 LayoutVersion이 움직이면 지연하여 다시 만들고, 값만 바뀌었으면 그룹 요약만 새로 고칩니다. 그룹화를 참조하십시오.

참고 항목

- 플랫폼
- 성능
- TsgcGridData
- TsgcGridRenderer
- TsgcGridLineView

에디션

sgcGrid는 모든 에디션이 같은 소스에서 빌드됩니다. 에디션은 빌드에 컴파일되는 기능과 배포되는 유닛만 바꿉니다.

에디션	포함 내용
Registered	모두 포함: VCL과 FMX 그리드, 모델, 정렬, 필터링, 그룹 패널이 있는 그룹화, 그룹과 푸터 요약, 고정 열, 트리 행, 마스터 디테일 행, Excel 내보내기, 선택, 깜박임, 서식 규칙, 인플레이스 편집, 가상 모드, JSON과 CSV, DataSet 바인딩, 테마, 헤드리스 PNG와 SVG 출력, 그리고 WebSocket, MQTT, SSE 피드 소스와 AI 쿼리를 갖춘 sgcGridWS 브리지.
Trial	Registered와 같은 기능을 제공합니다. SGC_TRIAL로 컴파일한 빌드는 빌드에 명시된 릴리스 기간이 지나면 체험판이라는 알림을 표시합니다. 그 외에는 달라지는 것이 없습니다.
Community	인플레이스 편집, 트리 행, 마스터 디테일 행, Excel 내보내기, sgcGridWS 브리지를 제외한 모든 것.

Community에서 빠지는 기능

- 제자리 편집. Community 에디션은 SGC_GRID_EDIT를 정의하지 않고 컴파일하므로 StartEdit, CommitEdit, CancelEdit, EditorActive와 OnBeforeEdit, OnValidate, OnEdited, OnEditError 이벤트가 존재하지 않으며, ReadOnly를 False로 해도 편집기가 없습니다. 제자리 편집을 참조하십시오.
- 트리 행, 마스터 디테일 행, Excel 내보내기. Community 에디션은 SGC_GRID_ADVANCED를 정의하지 않고 컴파일되므로 Tree와 Detail 속성, ExpandNode, CollapseNode, ExpandDetail, CollapseDetail, DetailExpanded, DetailData, SaveToXLSX, OnTreeExpandChange, OnDetailExpandChange, OnGetDetailData, OnDetailCellClick 이벤트, TsgcGridDataStore 컴포넌트가 존재하지 않습니다. 트리 행, 마스터 디테일, Excel 내보내기를 참조하십시오.
- sgcGridWS 브리지. sgcGridWS_*.pas 유닛과 해당 패키지가 배포되지 않으므로 피드 소스와 TsgcGridAIQuery를 사용할 수 없습니다. 실시간 피드와 그리드에 질문하기를 참조하십시오.

그룹화, 그룹 패널, 그룹과 푸터 요약, 고정 열은 Community를 포함한 모든 에디션에 들어 있습니다. 그룹화, 요약, 고정 열을 참조하십시오.

에디션은 sgcVer.inc에서 컴파일 시점에 선택하는 것(SGC_COMMUNITY이며, SGC_TRIAL도 끕니다)이지 런타임 스위치가 아닙니다. 편집 API, 트리, 디테일, Excel 멤버 또는 브리지를 사용하는 코드는 Community 빌드에서 컴파일되지 않습니다. 문제 해결을 참조하십시오.

설치

sgcGrid는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcGrid**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcGrid에 포함된 항목

- **sgcUICore**는 레이아웃 제품과 sgcCharts의 공통 기반입니다. 자체 확인란은 없습니다. sgcGrid를 선택하면 설치 프로그램이 자동으로 설치합니다.
- **sgcGrid for sgcWebSockets**는 **sgcWebSockets bridges** 아래에 표시됩니다. 피드 소스와 TsgcGridAIQuery(실시간 피드 참조)가 포함된 sgcGridWS 패키지를 설치합니다. 설치 프로그램은 sgcWebSockets가 이미 컴퓨터에 설치되어 있고 라이선스에 sgcGrid가 포함된 경우에만 이 항목을 제공합니다. 그렇지 않으면 회색으로 표시됩니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드입니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Grid`	sgcGrid 데모입니다. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcGrid도 포함됩니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 sgcGrid가 포함되며, sgcCharts, sgcDashboard, sgcNavigation, sgcToolbar, sgcWizard도 함께 포함됩니다. sgcWebSockets 브리지는 Community 설치 프로그램에 포함되지 않으며, sgcPDF와 sgcMaps도 Community에 포함되지 않습니다. Community 빌드에서 빠지는 기능은 에디션을 참조하십시오.

이전 독립 실행형 설치

설치 프로그램이 이전 독립 실행형 sgcUICore 설치(예: 독립 실행형 sgcCharts 설치 프로그램이 만든 것)를 발견하면, 그 설치의 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcGridD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcGrid<V>` / `dclsgcGrid<V>`	sgcGrid 컴포넌트(런타임/디자인 타임).
`sgcGridWS<V>` / `dclsgcGridWS<V>`	sgcWebSockets 브리지(런타임/디자인 타임). sgcWebSockets가 필요합니다.

의존 순서대로 빌드합니다. `sgcUICore`, 그다음 `sgcGrid`, 그다음 `dclsgcGrid`, 마지막으로 브리지 패키지입니다. 그런 다음 `dcl*` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder의 경우 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcGridC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

.NET

.NET에서는 달라지는 것이 없습니다. sgcGrid는 여전히 자체 NuGet 패키지(`esegece.sgcGrid`와 그 보조 패키지)로 배포됩니다. .NET 설치를 참조하십시오.

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12, 13. 프레임워크에 독립적인 런타임 패키지, VCL 및 FireMonkey 런타임 패키지, VCL 및 FireMonkey 디자인 타임 패키지가 있습니다. 아래 "패키지"를 참조하십시오.
C++Builder	C++Builder 2007부터 C++Builder 13까지, 생성된 .hpp 헤더와 정적 라이브러리를 제공합니다. 빠른 시작 C++Builder를 참조하십시오.
VCL	Windows에서 TsgcVCLGrid, Win32와 (지원하는 버전부터) Win64.
FireMonkey	Delphi XE6부터(SGC_GRID_FMX) Windows, macOS, iOS, Android에서 TsgcFMXGrid.
Linux64	Delphi 10.2부터(SGC_GRID_LINUX) 프레임워크에 독립적인 유닛(모델, 서식, 렌더러, JSON 및 CSV 가져오기와 내보내기). 예를 들어 서버에서 그리드 이미지를 렌더링할 때 사용합니다.
브리지	sgcGridWS 패키지는 같은 Delphi 또는 C++Builder 버전의 sgcGrid 패키지와 sgcWebSockets 패키지를 필요로 합니다.
.NET	WinForms, WPF, Avalonia에서 같은 API를 쓰는 C# 포트입니다. 헤드리스 PNG 및 SVG 출력, DataTable 소스, sgcWebSockets 및 sgcHTML 브리지를 갖추었고, .NET Framework 4.7.2와 .NET 8, 9에서 동작합니다. .NET 개요를 참고하십시오.

패키지

sgcGrid에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcGridD13.dpk, C++Builder 13에서는 sgcGridC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcGrid	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛. 모델, 열, 서식, 렌더러, 그룹화, 요약, 트리 행과 상세 행, JSON, CSV, XLSX 가져오기와 내보내기, TsgcGridDataStore, TsgcGridSource_DataSet.	rtl, dbrtl, sgcUICore
sgcGridVCL	런타임, VCL	sgcGrid_VCL (TsgcVCLGrid)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcGrid
sgcGridFMX	런타임, FireMonkey, XE6부터	sgcGrid_FMX (TsgcFMXGrid)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcGrid
dclsgcGrid	디자인 타임, VCL	**SGC Grid** 팔레트 페이지에 등록하는 컴포넌트는 TsgcGridSource_DataSet, TsgcGridDataStore(Community 에디션에는 없음), TsgcVCLGrid이며, 상세 옵션의 Source용 속성 편집기도 등록합니다. 이 편집기에는 그리드 데이터를 가진 컴포넌트만 나열됩니다.	designide, rtl, vcl, sgcGrid, sgcGridVCL

dclsgcGridFMX	디자인 타임, FireMonkey, XE6부터	**SGC Grid** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXGrid입니다.	designide, rtl, fmx, sgcGrid, sgcGridFMX, dclsgcGrid
sgcGridWS, dclsgcGridWS	런타임 및 디자인 타임, sgcWebSockets 브리지	라이브 피드 소스 (TsgcGridFeedSource_WebSocket, TsgcGridFeedSource_MQTT, TsgcGridFeedSource_SSE)와 TsgcGridAIQuery, 프레임워크 독립이며 **SGC Grid** 팔레트 페이지에 있습니다.	rtl, sgcGrid, sgcWebSockets

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcGridFMX에는 dclsgcGrid가 필요하므로 dclsgcGrid를 먼저 설치하십시오. 설치 프로그램은 sgcGrid와 함께 sgcUICore를 설치합니다. 패키지를 직접 빌드할 때는 sgcUICore를 먼저 빌드하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcGrid	sgcGridVCL	sgcGridFMX	dclsgcGrid, dclsgcGridFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcGrid, sgcGridVCL	sgcGridFMX	dclsgcGrid, dclsgcGridFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcGrid만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcGrid만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32

13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcGridC13.cbproj, sgcGridVCLC13.cbproj, sgcGridFMXC13.cbproj, dclsgcGridC13.cbproj, dclsgcGridFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXGrid)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcGridVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcGrid_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcGrid는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 모델, 서식, 렌더러, JSON 및 CSV 가져오기와 내보내기(예: 그리드 이미지를 렌더링할 때)입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

참고 항목

- 에디션
- 아키텍처

성능

그리드는 보이는 행과 열만 그리고, 프레임당 데이터 잠금을 한 번만 획득하며, 열 지오메트리를 캐시하고, 래스터 캔버스에서 글리프 이미지 캐시로 셀 텍스트를 그립니다. 프레임 비용은 행 수가 아니라 창 크기에 좌우됩니다.

프레임 예산

QA 스위트에는 성능 게이트가 있습니다. 열 12개, 행 100,000개, 보이는 행 45개로 이루어진 1920 x 1080 프레임을 래스터 캔버스에서 60회 렌더링합니다. 프레임당 Win64에서 16ms, Win32에서 25ms를 넘으면 게이트가 실패합니다. 마지막으로 측정한 평균은 다음과 같습니다.

플랫폼	일반 프레임	깜박임과 서식 규칙 3개
Win32	3.0 ms	3.2 ms
Win64	4.6 ms	6.8 ms
Linux64	3.7 ms	

따라서 1080p 그리드는 몇 밀리초 안에 렌더링되며, 60fps 프레임의 16.7ms에 충분히 못 미칩니다.

갱신 속도

갱신과 프레임은 분리되어 있습니다. 쓰기 스레드는 피드가 전달하는 만큼 자주 UpdateRow를 호출할 수 있으며, 그 사이에 몇 개의 행이 바뀌었든 그리드는 초당 최대 MaxFPS회만 다시 그립니다. LiveTicker 데모는 작업자 스레드에서 정렬되고 깜박이는 그리드로 초당 약 1,000건의 가격 갱신을 60fps로 기록합니다.

팁

- 여러 변경을 BeginUpdate와 EndUpdate로 감싸면 그리드가 알림을 한 번만 받습니다.
- 계속 바뀌는 열로 정렬하면 그 열이 바뀔 때마다 뷰가 재정렬됩니다. 필터가 없을 때 정렬 대상이 아닌 열의 변경은 현재 뷰를 유지합니다.
- 끝없이 늘어나는 로그와 피드에는 MaxRows를, 이미 다른 곳에 있는 매우 큰 데이터에는 가상 모드를 사용하십시오.
- 그리드가 부드러울 필요 없이 실시간처럼 보이기만 하면 될 때는 MaxFPS를 낮추십시오.

참고 항목

- 실시간 갱신과 스레딩
- 가상 모드
- 아키텍처

빠른 시작 VCL

디자인 타임

- sgcGrid 팔레트 페이지에서 **TsgcVCLGrid**를 폼에 놓고 Align을 alClient로 설정합니다.
- **Columns** 편집기를 열어 필드마다 열을 추가합니다. FieldName, Caption, DataType을 설정하고, 숫자와 날짜에는 DisplayFormat도 설정합니다.
- 코드에서 Data.AppendRow로 행을 채우고, 어떤 스레드에서든 Data.UpdateRow로 최신 상태를 유지합니다.

코드에서

AppendRow는 열 순서대로 값을 받고, UpdateRow는 필드 이름을 받으므로 가격만 담긴 메시지는 가격만 갱신합니다. 아직 없는 키는 Data.AutoInsert가 False가 아닌 한 UpdateRow가 삽입하기도 합니다.

참고 항목

- 빠른 시작 FMX
- 실시간 갱신과 스레딩
- TsgcVCLGrid

빠른 시작 FMX

TsgcFMXGrid에는 Delphi XE6 이상이 필요합니다. FireMonkey 폼에 놓거나 코드로 생성하십시오. VCL 그리드와 같은 열, 데이터, 이벤트를 가집니다.
FireMonkey에는 래스터 컨트롤용 네이티브 스크롤 막대가 없으므로 그리드가 자체 스크롤 막대를 그립니다. 마우스 휠은 세 행씩 스크롤하고, Shift를 누르면 가로로 스크롤하며, 터치 화면에서는 팬 제스처로 손가락을 따라 스크롤합니다.

참고 항목

- 빠른 시작 VCL
- TsgcFMXGrid

빠른 시작 C++Builder

C++Builder는 C++Builder 패키지가 생성하는 헤더와 정적 라이브러리를 통해 같은 구성 요소를 사용합니다. `sgcUI\Source`와 `sgcUI\LibD13\win32`(64비트는 `sgcUI\LibD13\win64`)를 include 경로와 라이브러리 경로에 추가합니다. 설치 프로그램이 IDE에 이미 이를 추가해 둡니다. sgcGrid와 sgcUICore의 헤더와 라이브러리는 그 하나의 폴더에 있습니다. sgcGrid_*.hpp 헤더를 include한 다음 sgcGridC13.lib와 sgcUICoreC13.lib를 링크하거나, 디자인 타임 패키지를 설치하여 폼에 TsgcVCLGrid를 놓으십시오.
Delphi 오픈 배열은 OPENARRAY 매크로로 전달합니다. AppendRow와 UpdateRow 호출은 C++에서도 스레드 안전합니다.

참고 항목

- 빠른 시작 VCL
- 데모
- 문제 해결

데모

데모는 설치 폴더의 Demos 폴더에 있으며, Delphi 프로젝트는 Demos\Delphi에, C++Builder 프로젝트는 Demos\CBuilder에 있습니다. 모든 데모 폴더에는 readme.html이 있습니다. 또한 모든 데모는 빌드 검증에 쓰이는 두 가지 스위치를 받습니다. /test는 점검을 실행하고 실패 횟수를 종료 코드로 반환하며, /snapshot은 창을 PNG 파일로 저장합니다.

데모	Delphi	C++Builder	보여 주는 내용
LiveTicker	예	예	작업자 스레드에서 UpdateRow로 초당 약 1,000건의 가격 갱신, 가격순 정렬 뷰, 변경 시 깜박임, 음수 변화에 대한 서식 규칙, TailMode와 다크 테마 토글, 초당 갱신 수와 프레임 수.
LiveTickerFMX	예	아니요	TsgcFMXGrid를 사용하는 LiveTicker 데모.
DataSetGrid	예	예	KeyField, AutoColumns, LiveUpdates를 사용해 레코드 200개의 TClientDataSet에 바인딩한 TsgcGridSource_DataSet, 현재 레코드를 양방향으로 따라가는 선택, JSON 및 CSV 내보내기와 가져오기.
Editing	예	예	선택 목록 열, 불리언 열, OnValidate, OnEditError, OnEdited를 사용한 제자리 편집. Community 에디션에는 없습니다.
GridAndChart	예	예	그리드 타일과 차트 타일이 있는 대시보드. 그리드에 바인딩한 TsgcChartDataSource_Table이 그리드의 행을 그립니다. sgcCharts와 sgcDashboard가 필요합니다.
AskYourGrid	예	예	규칙 기반 파서로, 일상 언어 질문에 따라 주식 200개를 필터링, 정렬, 강조하는 TsgcGridAIQuery. sgcWebSockets가 필요합니다.
LiveFeed	예	아니요	프로세스 내 WebSocket 서버가 공급하는 TsgcGridFeedSource_WebSocket. sgcWebSockets가 필요합니다.

C++Builder 참고 사항

- C++Builder 데모는 Win32와 Win64용 C++Builder 13의 클래식 컴파일러로 빌드하며 sgcGridC13.lib와 sgcUICoreC13.lib를 링크합니다.
- DataSetGrid는 Win32에서 MIDAS를 정적으로 링크합니다. C++Builder에는 Win64용 정적 MIDAS 라이브러리가 없으므로, 64비트 빌드에는 실행 파일 옆에 midas.dll이 필요합니다.
- GridAndChart는 sgcChartsC13.lib와 sgcDashboardC13.lib도 링크하며, AskYourGrid는 sgcGridWSC13.lib와 sgcWebSocketsC13.lib를 링크합니다.

참고 항목

- 빠른 시작 VCL
- 빠른 시작 C++Builder
- 문제 해결

TsgcVCLGrid

sgcGrid_VCL 유닛의 VCL 실시간 데이터 그리드입니다. TsgcGridData와 TsgcGridRenderer를 소유하고, sgcUICore 래스터 캔버스로 그리며, IsgcUITable(Data에 위임)과 IsgcGridDataHost를 구현하므로 차트 데이터 소스, 피드 소스, DataSet 소스를 그리드 구성 요소 자체에 바인딩할 수 있습니다.
기본 Windows 스크롤 막대를 사용합니다. 세로 막대는 행(행을 그룹화하는 동안에는 표시 줄)을, 가로 막대는 논리 픽셀을 세며 헤더 행은 고정되어 있습니다. 편집 멤버는 SGC_GRID_EDIT가 정의된 경우에만 존재합니다.

기능 예제의 이벤트 핸들러

기능 페이지의 예제는 이 작은 클래스의 메서드를 이벤트에 할당합니다.

속성

속성	형식	설명
Columns	TsgcGridColumns	그리드의 열입니다. TsgcGridColumns를 참조하십시오.
Data	TsgcGridData	행, 정렬, 필터, 변경 추적입니다. Public입니다. 그리드를 위해 IsgcUITable도 구현합니다. TsgcGridData를 참조하십시오.
Renderer	TsgcGridRenderer	그리드가 그릴 때 사용하는 렌더러입니다. Public, 읽기 전용입니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	선택과 변경 깜박임의 강조색입니다. 0이면 테마의 강조색을 사용합니다.
FontSize	Double	셀 텍스트의 크기입니다.
HeaderHeight, RowHeight	Integer	헤더(32)와 모든 행(28)의 높이이며, 논리 픽셀 단위입니다.
GridLinesH, GridLinesV	Boolean	가로 및 세로 격자선을 그립니다. 기본값은 True입니다.
AlternateRows	Boolean	한 행씩 건너 색조를 넣습니다. 기본값은 True입니다.
SelectionMode	TsgcGridSelectionMode	gsmRow(기본값)는 행 전체를, gsmCell은 단일 셀을 선택합니다.
BorderStyle	TBorderStyle (VCL), TsgcGridBorderStyle (FMX)	테마의 Border 색상으로 그린 단일 테두리이거나 테두리 없음입니다.
MaxRows	Integer	테일 모드입니다. 0이면 모든 행을 유지하고, 0보다 크면 삽입 시 가장 오래된 행을 삭제합니다.
TailMode	Boolean	그리드가 맨 아래로 스크롤되어 있는 동안 최신 행을 화면에 유지합니다.
FilterText	string	표시되는 텍스트에 이 문자열이 포함된 행만 보여 줍니다. 대소문자를 구분하지 않습니다.

ReadOnly	Boolean	기본값은 True입니다. False로 설정하면 Editable 열의 제자리 편집이 활성화됩니다 (SGC_GRID_EDIT).
FlashChanges, FlashDuration, FlashMode	Boolean, Integer, TsgcGridFlashMode	변경된 셀을 깜박이게 할지, 몇 밀리초 동안(800) 깜박일지, 배경에 적용할지 텍스트에 적용할지를 정합니다.
FlashUpColor, FlashDownColor, FlashChangedColor	TsgcUIColor	깜박임 색상입니다. 0이면 테마 기본값을 사용합니다.
FormatRules	TsgcGridFormatRules	조건부 서식 규칙입니다. 조건부 서식을 참조하십시오.
MaxFPS	Integer	초당 최대 다시 그리기 횟수입니다(60).
FrameCount	Cardinal	지금까지 그린 프레임 수입니다. Public, 읽기 전용입니다.
TopRow, ScrollX	Integer	처음 보이는 행(행을 그룹화하는 동안에는 처음 표시 줄)과 가로 스크롤 위치. Public입니다.
SelectedRow, SelectedCol	Integer	선택된 뷰 행과 열 인덱스이며, 없으면 -1입니다. Public입니다.
SelectedKey	string	선택된 행의 키입니다. 뷰가 바뀌어도 선택은 키를 따라갑니다. Public, 읽기 전용입니다.
FrozenColumns	Integer	가로로 스크롤되지 않는 표시 열의 수. 기본값은 0입니다. 고정 열을 참조하십시오.
ShowFooter	Boolean	열 요약이 있는 푸터 행을 표시합니다. 기본값은 False입니다. 요약을 참조하십시오.
ShowGroupPanel, GroupPanelHeight	Boolean, Integer	헤더 위에 그룹 패널을 표시하며(기본값 False) 그 높이를 DIP로 지정합니다(36). 그룹화를 참조하십시오.
GroupIndent	Integer	각 그룹 단계의 들여쓰기(DIP). 기본값은 20입니다.
GroupsExpanded	Boolean	새 그룹의 상태. 기본값은 True입니다. 설정해도 현재 그룹의 상태는 유지됩니다.
GroupSummaries	Boolean	그룹 행에 열 요약을 표시합니다. 기본값은 True입니다.
Lines	TsgcGridLineView	표시 줄: 그룹 행과 뷰 행. Public이며 읽기 전용입니다. TsgcGridLineView를 참조하십시오.
FocusedLine	Integer	포커스된 표시 줄. 없으면 -1. 그룹 줄에서는 SelectedRow가 -1입니다. Public입니다.
Tree	TsgcGridTreeOptions	트리 행: KeyField, ParentField, TreeField, Indent, Expanded. KeyField와 ParentField가 그리드 열을 가리키면 트리 모드가 켜집니다. Registered 에디션 전용입니다. 트리 행을 참조하십시오.
Detail	TsgcGridDetailOptions	마스터 디테일 행: Active, Source, MasterKeyField, DetailKeyField, Lines, Columns. Registered 에디션 전용입니다. 마스터 디테일을 참조하십시오.

메서드

메서드	반환값	설명
ScrollToRow(aRow)		뷰 행이 보이도록 스크롤합니다.
RowAt(X, Y), ColAt(X, Y)	Integer	한 지점 아래의 뷰 행과 열 인덱스이며, 없으면 -1입니다.
SaveToPNG(aFileName, aWidth, aHeight)		현재 뷰를 PNG 파일로 씁니다.
SaveToSVG(aFileName, aWidth, aHeight)		현재 뷰를 SVG 파일로 씁니다.
StartEdit(aRow, aCol)	Boolean	셀에서 편집기를 열거나 불리언 셀을 토글합니다(SGC_GRID_EDIT).
CommitEdit	Boolean	편집기 텍스트를 변환, 검증하여 씁니다. False이면 편집기가 열린 채로 유지됩니다(SGC_GRID_EDIT).
CancelEdit		쓰지 않고 편집기를 닫습니다(SGC_GRID_EDIT).
EditorActive, Editor, EditRow, EditCol		열려 있는 편집기의 상태입니다(SGC_GRID_EDIT). 속성입니다.
GroupBy(aFieldNames), ClearGrouping		지정한 필드(FieldName, 첫 번째가 단계 0, 알 수 없는 필드는 무시)로 그룹화하거나 그룹화를 제거합니다.
ExpandAll, CollapseAll		모든 그룹과 이후에 만들어지는 그룹을 펼치거나 접습니다. 트리 모드에서는 트리 노드에 작용합니다(Registered 에디션).
ExpandGroup(aGroup), CollapseGroup(aGroup)		그룹 하나를 펼치거나 접습니다.
GroupCount	Integer	그룹의 수.
GroupInfo(aGroup)	TsgcGridGroupInfo	그룹의 단계, 부모, 열, 멤버, 행 수, 상태, 줄.
FooterSummary(aCol), FooterSummaryText(aCol)	Variant, string	열의 푸터 값(Columns 인덱스)과 그 텍스트. 열에 SummaryType이 없으면 Null과 빈 문자열입니다.
ExpandNode(aRow), CollapseNode(aRow)		뷰 행 aRow의 트리 노드를 펼치거나 접고 OnTreeExpandChange를 발생시킵니다. Registered 에디션 전용입니다.
ExpandDetail(aRow), CollapseDetail(aRow)		마스터 뷰 행 aRow의 디테일 밴드를 열거나 닫고 OnDetailExpandChange를 발생시킵니다. Registered 에디션 전용입니다.
DetailExpanded(aRow)	Boolean	뷰 행 aRow의 밴드가 열려 있으면 True입니다. Registered 에디션 전용입니다.
DetailData(aRow)	TsgcGridData	뷰 행 aRow의 밴드의 행. 접혀 있으면 nil입니다. 읽기 전용입니다. Registered 에디션 전용입니다.
SaveToXLSX(aStream; aSheetName = 'Sheet1'), SaveToXLSX(aFileName; aSheetName = 'Sheet1')		현재 열 너비와 FrozenColumns를 유지하여 그리드를 Excel 통합 문서로 씁니다. Registered 에디션 전용입니다. Excel 내보내기를 참조하십시오.

이벤트

이벤트	형식	설명
OnCellClick, OnCellDblClick	TsgcGridCellEvent	셀을 클릭하거나 더블 클릭했습니다. 뷰 행과 열 인덱스가 전달됩니다.
OnSelectionChange	TNotifyEvent	선택된 행이나 셀이 바뀌었습니다.
OnSortChange	TNotifyEvent	헤더 클릭으로 정렬이 바뀌었습니다.
OnColumnResized	TsgcGridColumnEvent	마우스로 열 크기를 조정했습니다.
OnGetCellStyle	TsgcGridCellStyleEvent	그려지는 모든 셀의 색상과 굵은 스타일에 대한 최종 결정권입니다. 조건부 서식을 참조하십시오.
OnBeforeEdit	TsgcGridBeforeEditEvent	편집기가 열리려 합니다. aAllow를 False로 설정하면 거부합니다(SGC_GRID_EDIT).
OnValidate	TsgcGridValidateEvent	쓰기 전에 변환된 값입니다. 값을 바꾸거나 거부할 수 있습니다(SGC_GRID_EDIT).
OnEdited	TsgcGridEditedEvent	값이 기록되었으며, 이전 값과 새 값이 전달됩니다 (SGC_GRID_EDIT).
OnEditError	TsgcGridEditErrorEvent	텍스트가 열의 DataType으로 변환되지 않았습니다. aMessage를 바꿀 수 있습니다 (SGC_GRID_EDIT).
OnGroupingChange	TNotifyEvent	그룹 패널이나 코드로 그룹화한 열 또는 그 단계가 바뀌었습니다.
OnGroupExpandChange	TsgcGridGroupExpandEvent	그룹이 펼쳐지거나 접혔습니다. ExpandAll과 CollapseAll 뒤에는 aGroup이 -1입니다.
OnGetGroupText	TsgcGridGetGroupTextEvent	그룹 행 aGroup의 캡션 aText를 바꿉니다. 데이터 잠금을 잡은 상태로 발생합니다.
OnTreeExpandChange	TsgcGridRowExpandEvent	트리 노드가 펼쳐지거나 접혔습니다. 트리 모드에서 ExpandAll과 CollapseAll 뒤에는 aRow가 -1입니다. Registered 에디션 전용입니다.
OnDetailExpandChange	TsgcGridRowExpandEvent	디테일 밴드가 열리거나 닫혔습니다. aRow는 마스터 뷰 행입니다. Registered 에디션 전용입니다.
OnGetDetailData	TsgcGridGetDetailDataEvent	마스터 키 aMasterKey에 대한 밴드의 행(aData)을 채웁니다. 내용을 유지하려면 aHandled를 True로 설정합니다. Sender는 그리드입니다. 잠금 없이 발생합니다. Registered 에디션 전용입니다.
OnDetailCellClick	TsgcGridDetailCellEvent	밴드의 셀을 클릭했습니다. aRow는 마스터 뷰 행이고, aDetailRow와 aDetailCol은 밴드 데이터의 뷰 행과 Columns 인덱스입니다. Registered 에디션 전용입니다.

예제

참고 항목

- TsgcGridColumns
- TsgcGridData
- TsgcGridRenderer
- 선택과 키보드
- 제자리 편집
- 그룹화
- 요약
- 고정 열
- TsgcGridLineView
- 트리 행
- 마스터 디테일
- Excel 내보내기

TsgcFMXGrid

sgcGrid_FMX 유닛의 FireMonkey 실시간 데이터 그리드입니다(Delphi XE6 이상, Windows, macOS, iOS, Android). TsgcVCLGrid의 속성, 메서드, 이벤트에 더해 Position, Size, Margins, Opacity, 마우스 및 키 이벤트 같은 일반적인 FireMonkey 멤버를 게시합니다.
그리드는 내용이 뷰보다 클 때 오른쪽과 아래쪽 가장자리에 자체 스크롤 막대를 그립니다. 썸을 끌 수 있고 트랙을 누르면 페이지 단위로 이동합니다. 마우스 휠은 세 행씩 스크롤하고(Shift를 누르면 가로), 팬 제스처는 손가락을 따라 스크롤합니다. Ctrl+C는 선택 영역을 복사하며, macOS에서는 Cmd+C입니다. BorderStyle은 TsgcGridBorderStyle이며 gbsNone 또는 gbsSingle입니다. 쓰기 스레드에서는 그리드가 TThread.Queue로 메인 스레드에 뷰 갱신을 한 번 대기열에 넣습니다.

속성

속성	형식	설명
Columns	TsgcGridColumns	그리드의 열입니다. TsgcGridColumns를 참조하십시오.
Data	TsgcGridData	행, 정렬, 필터, 변경 추적입니다. Public입니다. 그리드를 위해 IsgcUITable도 구현합니다. TsgcGridData를 참조하십시오.
Renderer	TsgcGridRenderer	그리드가 그릴 때 사용하는 렌더러입니다. Public, 읽기 전용입니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	선택과 변경 깜박임의 강조색입니다. 0이면 테마의 강조색을 사용합니다.
FontSize	Double	셀 텍스트의 크기입니다.
HeaderHeight, RowHeight	Integer	헤더(32)와 모든 행(28)의 높이이며, 논리 픽셀 단위입니다.
GridLinesH, GridLinesV	Boolean	가로 및 세로 격자선을 그립니다. 기본값은 True입니다.
AlternateRows	Boolean	한 행씩 건너 색조를 넣습니다. 기본값은 True입니다.
SelectionMode	TsgcGridSelectionMode	gsmRow(기본값)는 행 전체를, gsmCell은 단일 셀을 선택합니다.
BorderStyle	TBorderStyle (VCL), TsgcGridBorderStyle (FMX)	테마의 Border 색상으로 그린 단일 테두리이거나 테두리 없음입니다.
MaxRows	Integer	테일 모드입니다. 0이면 모든 행을 유지하고, 0보다 크면 삽입 시 가장 오래된 행을 삭제합니다.
TailMode	Boolean	그리드가 맨 아래로 스크롤되어 있는 동안 최신 행을 화면에 유지합니다.
FilterText	string	표시되는 텍스트에 이 문자열이 포함된 행만 보여 줍니다. 대소문자를 구분하지 않습니다.
ReadOnly	Boolean	기본값은 True입니다. False로 설정하면 Editable 열의 제자리 편집이 활성화됩니다 (SGC_GRID_EDIT).

FlashChanges, FlashDuration, FlashMode	Boolean, Integer, TsgcGridFlashMode	변경된 셀을 깜박이게 할지, 몇 밀리초 동안(800) 깜박일지, 배경에 적용할지 텍스트에 적용할지를 정합니다.
FlashUpColor, FlashDownColor, FlashChangedColor	TsgcUIColor	깜박임 색상입니다. 0이면 테마 기본값을 사용합니다.
FormatRules	TsgcGridFormatRules	조건부 서식 규칙입니다. 조건부 서식을 참조하십시오.
MaxFPS	Integer	초당 최대 다시 그리기 횟수입니다(60).
FrameCount	Cardinal	지금까지 그린 프레임 수입니다. Public, 읽기 전용입니다.
TopRow, ScrollX	Integer	처음 보이는 행(행을 그룹화하는 동안에는 처음 표시 줄)과 가로 스크롤 위치. Public입니다.
SelectedRow, SelectedCol	Integer	선택된 뷰 행과 열 인덱스이며, 없으면 -1입니다. Public입니다.
SelectedKey	string	선택된 행의 키입니다. 뷰가 바뀌어도 선택은 키를 따라갑니다. Public, 읽기 전용입니다.
FrozenColumns	Integer	가로로 스크롤되지 않는 표시 열의 수. 기본값은 0입니다. 고정 열을 참조하십시오.
ShowFooter	Boolean	열 요약이 있는 푸터 행을 표시합니다. 기본값은 False입니다. 요약을 참조하십시오.
ShowGroupPanel, GroupPanelHeight	Boolean, Integer	헤더 위에 그룹 패널을 표시하며(기본값 False) 그 높이를 DIP로 지정합니다(36). 그룹화를 참조하십시오.
GroupIndent	Integer	각 그룹 단계의 들여쓰기(DIP). 기본값은 20입니다.
GroupsExpanded	Boolean	새 그룹의 상태. 기본값은 True입니다. 설정해도 현재 그룹의 상태는 유지됩니다.
GroupSummaries	Boolean	그룹 행에 열 요약을 표시합니다. 기본값은 True입니다.
Lines	TsgcGridLineView	표시 줄: 그룹 행과 뷰 행. Public이며 읽기 전용입니다. TsgcGridLineView를 참조하십시오.
FocusedLine	Integer	포커스된 표시 줄. 없으면 -1. 그룹 줄에서는 SelectedRow가 -1입니다. Public입니다.
Tree	TsgcGridTreeOptions	트리 행: KeyField, ParentField, TreeField, Indent, Expanded. KeyField와 ParentField가 그리드 열을 가리키면 트리 모드가 켜집니다. Registered 에디션 전용입니다. 트리 행을 참조하십시오.
Detail	TsgcGridDetailOptions	마스터 디테일 행: Active, Source, MasterKeyField, DetailKeyField, Lines, Columns. Registered 에디션 전용입니다. 마스터 디테일을 참조하십시오.

메서드

메서드	반환값	설명

ScrollToRow(aRow)		뷰 행이 보이도록 스크롤합니다.
RowAt(X, Y), ColAt(X, Y)	Integer	한 지점 아래의 뷰 행과 열 인덱스이며, 없으면 -1입니다.
SaveToPNG(aFileName, aWidth, aHeight)		현재 뷰를 PNG 파일로 씁니다.
SaveToSVG(aFileName, aWidth, aHeight)		현재 뷰를 SVG 파일로 씁니다.
StartEdit(aRow, aCol)	Boolean	셀에서 편집기를 열거나 불리언 셀을 토글합니다 (SGC_GRID_EDIT).
CommitEdit	Boolean	편집기 텍스트를 변환, 검증하여 씁니다. False이면 편집기가 열린 채로 유지됩니다(SGC_GRID_EDIT).
CancelEdit		쓰지 않고 편집기를 닫습니다(SGC_GRID_EDIT).
EditorActive, Editor, EditRow, EditCol		열려 있는 편집기의 상태입니다(SGC_GRID_EDIT). 속성입니다.
GroupBy(aFieldNames), ClearGrouping		지정한 필드(FieldName, 첫 번째가 단계 0, 알 수 없는 필드는 무시)로 그룹화하거나 그룹화를 제거합니다.
ExpandAll, CollapseAll		모든 그룹과 이후에 만들어지는 그룹을 펼치거나 접습니다. 트리 모드에서는 트리 노드에 작용합니다 (Registered 에디션).
ExpandGroup(aGroup), CollapseGroup(aGroup)		그룹 하나를 펼치거나 접습니다.
GroupCount	Integer	그룹의 수.
GroupInfo(aGroup)	TsgcGridGroupInfo	그룹의 단계, 부모, 열, 멤버, 행 수, 상태, 줄.
FooterSummary(aCol), FooterSummaryText(aCol)	Variant, string	열의 푸터 값(Columns 인덱스)과 그 텍스트. 열에 SummaryType이 없으면 Null과 빈 문자열입니다.
ExpandNode(aRow), CollapseNode(aRow)		뷰 행 aRow의 트리 노드를 펼치거나 접고 OnTreeExpandChange를 발생시킵니다. Registered 에디션 전용입니다.
ExpandDetail(aRow), CollapseDetail(aRow)		마스터 뷰 행 aRow의 디테일 밴드를 열거나 닫고 OnDetailExpandChange를 발생시킵니다. Registered 에디션 전용입니다.
DetailExpanded(aRow)	Boolean	뷰 행 aRow의 밴드가 열려 있으면 True입니다. Registered 에디션 전용입니다.
DetailData(aRow)	TsgcGridData	뷰 행 aRow의 밴드의 행. 접혀 있으면 nil입니다. 읽기 전용입니다. Registered 에디션 전용입니다.
SaveToXLSX(aStream; aSheetName = 'Sheet1'), SaveToXLSX(aFileName; aSheetName = 'Sheet1')		현재 열 너비와 FrozenColumns를 유지하여 그리드를 Excel 통합 문서로 씁니다. Registered 에디션 전용입니다. Excel 내보내기를 참조하십시오.

이벤트

이벤트	형식	설명
OnCellClick, OnCellDblClick	TsgcGridCellEvent	셀을 클릭하거나 더블 클릭했습니다. 뷰 행과 열 인덱스가 전달됩니다.
OnSelectionChange	TNotifyEvent	선택된 행이나 셀이 바뀌었습니다.
OnSortChange	TNotifyEvent	헤더 클릭으로 정렬이 바뀌었습니다.
OnColumnResized	TsgcGridColumnEvent	마우스로 열 크기를 조정했습니다.
OnGetCellStyle	TsgcGridCellStyleEvent	그려지는 모든 셀의 색상과 굵은 스타일에 대한 최종 결정권입니다. 조건부 서식을 참조하십시오.
OnBeforeEdit	TsgcGridBeforeEditEvent	편집기가 열리려 합니다. aAllow를 False로 설정하면 거부합니다(SGC_GRID_EDIT).
OnValidate	TsgcGridValidateEvent	쓰기 전에 변환된 값입니다. 값을 바꾸거나 거부할 수 있습니다(SGC_GRID_EDIT).
OnEdited	TsgcGridEditedEvent	값이 기록되었으며, 이전 값과 새 값이 전달됩니다(SGC_GRID_EDIT).
OnEditError	TsgcGridEditErrorEvent	텍스트가 열의 DataType으로 변환되지 않았습니다. aMessage를 바꿀 수 있습니다 (SGC_GRID_EDIT).
OnGroupingChange	TNotifyEvent	그룹 패널이나 코드로 그룹화한 열 또는 그 단계가 바뀌었습니다.
OnGroupExpandChange	TsgcFMXGridGroupExpandEvent	그룹이 펼쳐지거나 접혔습니다. ExpandAll과 CollapseAll 뒤에는 aGroup이 -1입니다.
OnGetGroupText	TsgcGridGetGroupTextEvent	그룹 행 aGroup의 캡션 aText를 바꿉니다. 데이터 잠금을 잡은 상태로 발생합니다.
OnTreeExpandChange	TsgcFMXGridRowExpandEvent	트리 노드가 펼쳐지거나 접혔습니다. 트리 모드에서 ExpandAll과 CollapseAll 뒤에는 aRow가 -1입니다. Registered 에디션 전용입니다.
OnDetailExpandChange	TsgcFMXGridRowExpandEvent	디테일 밴드가 열리거나 닫혔습니다. aRow는 마스터 뷰 행입니다. Registered 에디션 전용입니다.
OnGetDetailData	TsgcGridGetDetailDataEvent	마스터 키 aMasterKey에 대한 밴드의 행(aData)을 채웁니다. 내용을 유지하려면 aHandled를 True로 설정합니다. Sender는 그리드입니다. 잠금 없이 발생합니다. Registered 에디션 전용입니다.
OnDetailCellClick	TsgcFMXGridDetailCellEvent	밴드의 셀을 클릭했습니다. aRow는 마스터 뷰 행이고, aDetailRow와 aDetailCol은 밴드 데이터의 뷰 행과 Columns 인덱스입니다. Registered 에디션 전용입니다.

예제

참고 항목

- TsgcGridColumns
- TsgcGridData
- TsgcGridRenderer
- 선택과 키보드
- 제자리 편집
- 그룹화
- 요약
- 고정 열
- TsgcGridLineView
- 트리 행
- 마스터 디테일
- Excel 내보내기

TsgcGridColumn

sgcGrid_Columns 유닛에 있는 그리드의 열이며, TsgcGridColumns의 항목입니다. 데이터는 열 ID로 값을 지정하므로, 열을 이동하거나 삭제해도 다른 열의 값이 밀리지 않습니다.

속성

속성	형식	설명
FieldName	string	데이터, 피드, 가져오기, 서식 규칙이 열을 가리킬 때 사용하는 이름입니다.
Caption	string	헤더 텍스트입니다.
DataType	TsgcGridDataType	gdtText(기본값), gdtInteger, gdtFloat, gdtCurrency, gdtDateTime 또는 gdtBoolean. 정렬, 서식, 맞춤, 편집, 규칙 비교를 결정합니다.
Width, MinWidth	Integer	너비(100)와 마우스로 크기를 조정할 때 허용되는 최소 너비(20)이며, 논리 픽셀 단위입니다.
Alignment	TsgcGridAlignment	gaLeft, gaCenter, gaRight 또는 gaAuto(기본값)이며, gaAuto는 숫자는 오른쪽, 텍스트는 왼쪽, 불리언은 가운데입니다.
DisplayFormat	string	숫자용 FormatFloat 형식 또는 날짜용 FormatDateTime 형식입니다. DisplayFormat이 없는 불리언 열은 체크 표시로 그려집니다.
Visible	Boolean	열을 표시합니다. 숨긴 열도 값과 인덱스를 유지합니다.
Editable	Boolean	그리드가 ReadOnly가 아닐 때 열을 제자리에서 편집할 수 있습니다. 기본값은 True입니다.
SortOrder, SortIndex	TsgcGridSortOrder, Integer	gsoNone, gsoAscending 또는 gsoDescending, 그리고 다중 열 정렬에서 열의 순위입니다.
PickList	TStrings	열의 콤보 상자 편집기에 표시되는 값입니다.
Tag	Integer	애플리케이션이 자유롭게 사용할 수 있습니다.
GroupIndex	Integer	0 이상의 값은 그 단계에서 이 열로 그리드를 그룹화합니다(단계는 GroupIndex 순, 그다음 Index 순). -1(기본값)은 그룹화하지 않습니다. 그룹화를 참조하십시오.
SummaryType	TsgcGridSummaryType	gsuNone(기본값), gsuSum, gsuCount, gsuAverage, gsuMin, gsuMax 중 하나. 푸터와 그룹 행에 표시되는 열의 요약입니다. 요약을 참조하십시오.
SummaryFormat	string	요약의 FormatFloat 또는 FormatDateTime 서식 문자열. 리터럴 텍스트는 큰따옴표 안에 넣습니다. 비어 있으면 DisplayFormat을 사용하고, gsuCount는 0을 사용합니다.

메서드

메서드	반환값	설명

IsNumeric	Boolean	gdtInteger, gdtFloat, gdtCurrency이면 True입니다.
EffectiveAlignment	TsgcGridAlignment	gaAuto가 실제로 결정되는 맞춤입니다.

예제

참고 항목

- TsgcGridColumns
- 정렬
- 제자리 편집
- 그룹화
- 요약

TsgcGridColumns

sgcGrid_Columns 유닛에 있는 TsgcGridColumn 항목의 컬렉션이며, 그리드 또는 독립 TsgcGridData가 가집니다.

속성

속성	형식	설명
Items[Index]	TsgcGridColumn	열이며, 기본 속성입니다.
VisibleColumn[Index]	TsgcGridColumn	표시되는 열을 순서대로 나타냅니다.
Version	Cardinal	열이 바뀔 때마다 증가합니다.

메서드

메서드	반환값	설명
Add	TsgcGridColumn	빈 열을 추가합니다.
Add(aFieldName, aCaption, aDataType)	TsgcGridColumn	필드 이름, 캡션, 데이터 형식(기본값 gdtText)을 지정하여 열을 추가합니다.
FindByField(aFieldName)	TsgcGridColumn	필드 이름에 해당하는 열이며, 없으면 nil입니다.
IndexOfField(aFieldName)	Integer	필드 이름에 해당하는 열의 인덱스이며, 없으면 -1입니다.
VisibleCount	Integer	표시되는 열의 수입니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	열의 속성 중 하나가 바뀌었습니다.
OnModelChange	TNotifyEvent	정렬이나 새 열처럼 데이터나 뷰에 영향을 주는 변경입니다.

예제

참고 항목

- TsgcGridColumn
- TsgcGridData

TsgcGridData

sgcGrid_Data 유닛에 있는, 프레임워크에 독립적이고 스레드 안전한 sgcGrid의 행 저장소입니다. 모든 그리드가 Data 속성으로 하나씩 소유하며, 예를 들어 TsgcGridRenderer와 함께 단독으로 사용할 수도 있습니다. Create(nil)은 자체 열을 생성하여 소유합니다.
행은 삽입 순서대로 유지되며 키로 O(log n)에 찾습니다. Cells, CellText, KeyOf, RowOfKey, CellChange의 행 인덱스는 뷰 행(정렬 및 필터링된 행)이고, 열 인덱스는 숨긴 열을 포함한 Columns 인덱스입니다. 범위를 벗어난 읽기는 Null, 빈 문자열 또는 cdNone을 반환하며 예외를 발생시키지 않습니다. 개수를 얻은 뒤 읽기 전에 쓰기 스레드가 행을 삭제할 수 있기 때문입니다.

속성

속성	형식	설명
Columns	TsgcGridColumns	열입니다.
Cells[aRow, aCol]	Variant	뷰 행의 값입니다.
CellText[aRow, aCol]	string	열의 DisplayFormat으로 서식을 적용한 값입니다.
StoreCells[aStoreRow, aCol]	Variant	저장소 행의 값이며, OnFilterRow에서 사용합니다.
FormatSettings	TFormatSettings	값을 서식화하고 구문 분석할 때 사용하는 설정입니다.
AutoInsert	Boolean	UpdateRow와 UpdateCell이 없는 키를 삽입합니다. 기본값은 True입니다.
MaxRows	Integer	0은 제한 없음이며, 그 외에는 삽입 시 가장 오래된 행을 삭제합니다.
FilterText	string	표시되는 텍스트에 이 문자열이 포함된 행만 유지합니다. 대소문자를 구분하지 않습니다.
Virtual, VirtualRowCount	Boolean, Integer	가상 모드와 그 행 수입니다. 가상 모드를 참조하십시오.
LayoutVersion	Cardinal	뷰 행의 그룹화나 트리 레이아웃이 바뀌었을 수 있을 때 증가합니다. 삽입, 삭제, Clear, 뷰나 열의 변경, 그룹화한 열의 변경, GroupIndex가 0 이상인 열이나 SetLayoutFields에 전달된 필드의 값 변경이 해당합니다. 다른 열의 값 변경은 그대로 둡니다. 그리드의 줄 뷰가 이를 읽습니다. 읽기 전용입니다.

메서드

메서드	반환값	설명
AppendRow(aKey, aValues)	Integer	열 순서대로 값을 지정하여 행을 추가하고 그 저장소 행을 반환합니다. 키가 비어 있으면 고유한 키 #<n>을 생성하고, 이미 있는 키이면 해당 행을 갱신합니다.
UpdateRow(aKey, aFields, aValues)		행의 지정한 필드를 설정합니다. 알 수 없는 필드는 무시합니다.

UpdateCell(aKey, aField, aValue)		행의 필드 하나를 설정합니다.
DeleteRow(aKey)	Boolean	키에 해당하는 행을 삭제합니다.
Clear		모든 행을 삭제합니다.
BeginUpdate, EndUpdate, IsUpdating		여러 변경을 하나의 OnChange로 합칩니다.
Lock, Unlock		여러 번의 읽기 동안 내부 잠금을 유지합니다.
RowCount, ViewCount	Integer	저장된 행 수(가상 모드에서는 VirtualRowCount)와 필터를 통과한 행 수입니다.
ViewRow(aRow)	Integer	뷰 행에 해당하는 저장소 행이며, 범위를 벗어나면 -1입니다.
KeyOf(aRow), RowOfKey(aKey), StoreKeyOf(aStoreRow)		뷰 행, 저장소 행, 키 사이를 변환합니다. 키가 없거나 필터로 제외되면 RowOfKey는 -1을 반환합니다.
CellChange(aRow, aCol)	TsgcGridCellChange	셀의 마지막 변경에 대한 Direction(cdNone, cdUp, cdDown, cdChanged)과 밀리초 단위 Age입니다.
GetTick	Int64	변경 스탬프에 쓰이는 단조 증가 밀리초 시계입니다.
InvalidateView		뷰를 다시 만듭니다. 예를 들어 OnFilterRow의 로직이 바뀐 뒤에 사용합니다.
UseInvariantFormat		FormatSettings를 소수점 구분 기호 점, 천 단위 구분 기호 쉼표, yyyy-mm-dd, hh:nn:ss로 설정합니다.
SetLayoutFields(aFields)		값이 바뀌면 그룹화한 열처럼 LayoutVersion도 움직이는 필드(FieldName, 대소문자 구분 없음). 빈 배열은 목록을 지웁니다.

이벤트

이벤트	형식	설명
OnChange	TsgcGridChangeEvent	gckData, gckView 또는 gckStructure이며, 잠금이 해제된 후 쓰기 스레드에서 발생합니다.
OnFilterRow	TsgcGridFilterRowEvent	저장소 행을 수락하거나 거부합니다. 잠금을 잡은 상태에서 호출됩니다.
OnGetValue	TsgcGridGetValueEvent	가상 모드의 값입니다.
OnGetTick	TsgcGridGetTickEvent	변경 스탬프의 시계를 대체합니다. 예를 들어 테스트에서 사용합니다.

TsgcGridData는 IsgcUITable도 구현합니다. IsgcGridDataHost와 IsgcUITable을 참조하십시오.

예제

참고 항목

- 실시간 갱신과 스레딩

- 필터링
- 정렬
- TsgcGridLineView

TsgcGridLineView

sgcGrid의 프레임워크 중립 줄 뷰로, 유닛 sgcGrid_Lines에 있습니다. 그리드의 표시 줄을 TsgcGridData의 뷰 행에 대응시킵니다. 모든 그리드가 Lines 속성에 하나씩 가지고 있으며, TsgcGridData와 TsgcGridRenderer만으로 단독으로도 사용할 수 있습니다. 데이터를 소유하지 않으므로 데이터보다 먼저 해제하십시오.
줄 뷰는 뷰 상태입니다. 데이터, 그 뷰 행, IsgcUITable은 바뀌지 않습니다. 모든 줄은 한 행 높이이므로 줄 인덱스는 스크롤 위치입니다. 그룹화한 열이 없거나(Registered 에디션에서는 트리나 펼쳐진 디테일 밴드도 없을 때), 데이터가 가상이면 아무것도 만들어지지 않으며 줄 i는 뷰 행 i입니다. 이것이 항등 변환이며 평평한 그리드입니다.

줄 종류

TsgcGridLineKind	Line
glkRow	뷰 행. Row가 이를 반환합니다.
glkGroup	그룹 행. Group이 그룹을 반환하고 Row는 -1을 반환합니다.
glkDetail	디테일 밴드의 줄(Registered 에디션, 마스터 디테일 참조). Row는 마스터 행을 반환합니다.

그룹화된 줄은 깊이 우선 순서의 그룹 행이며, 각각 뒤에 자식 그룹이 오고 마지막 단계의 그룹 뒤에는 뷰 행이 옵니다. 그룹 줄은 가장 바깥 단계가 Level 0이고, 행 줄은 Level이 GroupLevels입니다.

메서드

메서드	반환값	설명
Create(aData)		aData에 대한 줄 뷰를 만듭니다.
Update		줄을 데이터에 맞게 최신 상태로 만듭니다. 아래의 모든 리더가 이를 호출하므로 아무것도 바뀌지 않았을 때는 비용이 들지 않습니다.
Invalidate		다음 읽기에서 다시 만들도록 강제합니다.
Identity	Boolean	줄 i가 뷰 행 i이면 True입니다.
Count	Integer	표시 줄의 수.
Kind(aLine)	TsgcGridLineKind	줄의 종류. 범위를 벗어나면 glkRow입니다.
Row(aLine)	Integer	행 줄이면 뷰 행, 디테일 줄이면 마스터 행, 그룹 줄이거나 범위를 벗어나면 -1.
Level(aLine)	Integer	줄의 들여쓰기 수준.
Group(aLine)	Integer	그룹 줄이면 그 그룹, 행 줄이면 가장 안쪽 그룹. 항등 변환이거나 범위를 벗어나면 -1.
Expandable(aLine), Expanded(aLine)	Boolean	줄을 펼치거나 접을 수 있는지와 그 상태.
ToggleLine(aLine)	Boolean	줄을 전환합니다. 전환할 것이 없으면 False입니다.
LineOfRow(aRow)	Integer	뷰 행의 줄. 접힌 그룹 안이거나 범위를 벗어나면 -1.

GroupLevels, GroupCount	Integer	그룹화한 단계의 수와 그룹의 수.
GroupInfo(aGroup)	TsgcGridGroupInfo	그룹. 깊이 우선 순서로 `0`부터 `GroupCount - 1`까지.
GroupSummary(aGroup, aCol)	Variant	그룹의 행에 대한 열(Columns 인덱스)의 요약. 열에 SummaryType이 없거나 aGroup이 범위를 벗어나면 Null입니다.
GroupBy(aFieldNames), ClearGrouping		지정한 열(첫 번째가 단계 0)에 GroupIndex를 설정하고 나머지에는 -1을 설정하거나, 모든 열을 -1로 설정합니다.
SetGroupExpanded(aGroup, aExpanded)		그룹 하나를 펼치거나 접습니다.
ExpandAll, CollapseAll		모든 그룹과 이후에 만들어지는 그룹.
FlattenAll	TsgcGridFlatLines	접혀 있는지 여부와 관계없이 모든 그룹 행과 뷰 행을 줄 순서대로 나열한 것입니다. Hidden은 접힌 그룹 안의 줄을 나타냅니다. 디테일 줄은 나열되지 않습니다.

속성

속성	형식	설명
Data	TsgcGridData	줄 뷰의 데이터. 읽기 전용입니다.
GroupsExpanded	Boolean	전환하지 않은 그룹과 새 그룹의 상태. 설정해도 현재 그룹의 상태는 유지됩니다.

트리와 디테일 멤버(TreeMode, NodeExpanded, SetNodeExpanded, DetailExpandable, DetailExpanded, SetDetailExpanded, ToggleDetail, DetailView, DetailIndex, TreeOptions, DetailOptions, Tree, Detail)는 Registered 에디션에만 있습니다. 트리 행과 마스터 디테일을 참조하십시오.

TsgcGridGroupInfo

Field	형식	설명
Level	Integer	가장 바깥쪽 수준은 0입니다.
Parent	Integer	부모 그룹입니다. 수준 0에서는 -1입니다.
Column	Integer	그룹화한 열의 Columns 인덱스.
RowCount	Integer	그룹에 속한 표시 행의 수이며, 자식 그룹의 행도 포함합니다.
FirstRow	Integer	줄 순서에서 그룹의 첫 번째 행의 뷰 행.
Text	string	멤버 텍스트. 빈 멤버이면 (blank).
값	Variant	첫 번째 행의 형식 값. 빈 멤버이면 Null.
Blank, Expanded	Boolean	그룹이 빈 멤버인지와 그 상태.
Line	Integer	그룹 행의 라인입니다. 접힌 그룹 안에서는 -1입니다.
Path	string	단계 0부터 멤버 텍스트를 #1로 이어 붙인 것.

TsgcGridFlatLines는 Count와 배열 Kind, Ref(뷰 행 또는 그룹), Level, Hidden을 가진 레코드입니다.

스레딩

줄 뷰는 그리드를 소유한 스레드(컨트롤이면 메인 스레드)에서 호출하십시오. 라이터 스레드는 계속 데이터에 쓰며, 줄 뷰는 데이터의 LayoutVersion이 움직였으면 다음 읽기에서 다시 만들고, 값만 바뀌었으면 그룹 요약만 새로 고칩니다. 그룹화는 TsgcGridGrouper(유닛 sgcGrid_Group)가 처리합니다.

예제

컨트롤 없이 세 행을 Region으로 그룹화하고, 각 그룹을 행 수와 Amount의 합계와 함께 출력한 다음 그 아래의 제품을 출력합니다.

참고 항목

- 그룹화
- 요약
- TsgcGridData
- TsgcGridRenderer
- .NET 그룹화와 요약

TsgcGridDataStore

행을 표시하지 않고 보관하는 비주얼이 아닌 컴포넌트로, 유닛 sgcGrid_Detail에 있습니다. 역할은 그리드의 디테일 밴드의 Source가 되는 것입니다. 마스터 디테일을 참조하십시오. Grid가 이 저장소인 TsgcGridSource_DataSet 또는 코드로 채우므로, TDataSet 디테일을 코드 없이 폼 디자이너에서 사용할 수 있습니다. 컴포넌트 팔레트의 SGC Grid 페이지에 있습니다. 이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.
TsgcGridDataStore는 TComponent에서 상속하며 IsgcGridDataHost와 IsgcUITable을 구현하고, IsgcUITable은 Data에 위임됩니다.

속성

속성	형식	설명
Columns	TsgcGridColumns	행의 열. published이므로 디자이너에서 설정합니다. TsgcGridSource_DataSet을 사용하고 AutoColumns가 True이면 필드에서 만들어집니다.
Data	TsgcGridData	행. 그리드의 Data처럼 읽고 씁니다. AppendRow, UpdateRow, DeleteRow, BeginUpdate, EndUpdate. Public이며 읽기 전용입니다.

사용 방법

- 그리드의 Detail.Source를 저장소로 설정하고, DetailKeyField를 마스터 키를 담은 저장소의 열로 설정합니다.
- 코드나 TDataSet 소스에 의한 저장소 행의 변경은 다음 프레임에 열려 있는 디테일 밴드를 다시 채웁니다.
- 저장소를 해제하면 그리드의 Detail.Source는 nil이 됩니다.

예제

주문 TDataSet을 저장소에 불러와 고객 그리드의 디테일로 표시합니다.
디자이너에서도 같은 일을 할 수 있습니다. TsgcGridSource_DataSet에서 DataSet, Grid(저장소), KeyField, Active를 설정한 다음, 그리드에서 Detail.Source, Detail.MasterKeyField, Detail.DetailKeyField, Detail.Active를 설정합니다. TreeAndDetail 데모가 이 방식으로 동작합니다.

참고 항목

- 마스터 디테일
- TsgcGridData
- TsgcGridColumns
- TsgcGridSource_DataSet
- 에디션
- .NET 트리 행과 마스터 디테일

TsgcGridRenderer

sgcGrid_Render 유닛에 있는 프레임워크에 독립적인 sgcGrid 렌더러입니다. TsgcGridData를 래스터든 SVG든 임의의 sgcUICore 캔버스에 그리므로 VCL 그리드, FMX 그리드, 헤드리스 출력이 모두 똑같이 보입니다. 지오메트리는 논리 픽셀 단위이며, Render는 장치 픽셀 단위의 캔버스 크기와 배율을 받아 모든 가장자리를 장치 픽셀의 정수 위치에 맞춥니다.

TsgcGridRenderOptions

그리드의 published 모양: HeaderHeight, RowHeight, CellPadding(8), GridLinesH, GridLinesV, AlternateRows, SelectionMode, Theme, AccentColor, FontSize, HeaderFontSize, ShowSortGlyphs, FlashChanges, FlashDuration, FlashUpColor, FlashDownColor, FlashChangedColor, FlashMode, FormatRules. 그리고 줄 뷰레이아웃인 FrozenColumns, ShowFooter, ShowGroupPanel, GroupPanelHeight, GroupIndent, GroupSummaries가 있습니다. 그리드 컨트롤도 같은 속성을 제공합니다. Colors는 확정된 테마 색을 반환합니다.

TsgcGridRenderState

프레임의 뷰 상태를 담은 레코드: ScrollX, ScrollY, SelectedRow, SelectedCol, HotRow, HotCol, FocusRow, FocusCol, Focused, FocusLine, HotLine, ColWidths. sgcGridRenderState는 아무것도 선택되지 않은 빈 상태를 반환합니다. FocusLine과 HotLine은 표시 줄이며 기본값은 -1이고, FocusLine이 -1이면 FocusRow의 줄에 포커스합니다.

속성

속성	형식	설명
Options	TsgcGridRenderOptions	모양입니다.
CurrentData, CurrentNow	TsgcGridData, Int64	그리고 있는 프레임의 데이터와 틱이며, OnGetCellStyle에서 유효합니다.
Fading	Boolean	아직 진행 중인 깜박임을 그린 프레임 뒤에는 True입니다.
FooterSummaries	TsgcGridSummaryCache	푸터 값. ShowFooter가 켜져 있을 때 모든 프레임마다 새로 고칩니다.

메서드

메서드	반환값	설명
Render(aData, aState, aCanvas, aWidth, aHeight, aScale, aNow)		프레임을 그립니다.
SaveToPNG, SaveToSVG		프레임을 스트림이나 파일에 쓰며, 배율은 선택 사항입니다.
HitTest(aData, aState, aX, aY)	TsgcGridHitInfo	한 지점 아래의 부분(ghpNone, ghpHeader, ghpHeaderResize, ghpCell), 행, 열입니다.
ColumnWidth, ColumnLeft, ColumnRight, ContentWidth, ContentHeight, VisibleRowRange		지오메트리 도우미입니다.

Render(aData, aLines, aState, aCanvas, aWidth, aHeight, aScale, aNow)		표시 줄의 프레임을 그립니다. 그룹 패널, 헤더, 줄, 푸터입니다. aLines가 nil이면 평평한 그리드입니다. TsgcGridLineView를 참조하십시오.
SaveToPNG(aData, aLines, ...), SaveToSVG(aData, aLines, ...)		표시 줄의 프레임을 스트림이나 파일에 씁니다.
HitTest(aData, aLines, aState, aX, aY, aHeight)	TsgcGridHitInfo	표시 줄에 대한 히트 테스트. 푸터를 맞히려면 aHeight에 뷰포트 높이를 전달하며, 0이면 푸터는 맞지 않습니다.
VisibleLineRange, GutterWidth, FrozenWidth, ContentWidth(aColumns, aLines, aState)		줄 뷰 지오메트리: 그룹 패널과 푸터를 제외한 표시 줄, 그룹 셰브론의 너비, 스크롤되지 않는 부분의 너비, 전체 너비.
GroupPanelDropIndex, GroupChipBounds, PaintDragChip		그룹 패널의 놓을 위치, 칩의 경계, 끌고 있는 칩의 그리기. 헤더를 그룹 패널로 끄는 호스트용입니다.
GroupText(aLines, aGroup)	string	그룹 행의 캡션. OnGetGroupText가 적용됩니다.

줄에 대해 TsgcGridHitPart는 ghpGroupPanel, ghpGroupChip, ghpGroupRow, ghpExpandButton(셰브론), ghpFooter, ghpDetail을 추가하고, TsgcGridHitInfo는 Line(표시 줄, 줄 밖이면 -1), Group(그 줄의 그룹이며, ghpGroupChip에서는 칩, ghpGroupPanel에서는 놓을 위치), DetailRow, DetailCol을 추가합니다.

이벤트

이벤트	형식	설명
OnGetCellStyle	TsgcGridCellStyleEvent	그려지는 모든 셀에 대해 데이터 잠금을 잡은 상태에서 마지막으로 발생합니다. Sender는 렌더러입니다.
OnGetGroupText	TsgcGridGetGroupTextEvent	그룹 행의 캡션을 바꿉니다. 데이터 잠금을 잡은 상태로 발생합니다.

프로시저 sgcGridRender, sgcGridSaveToPNG, sgcGridSaveToSVG는 임시 렌더러와 TsgcGridRenderOptions로 같은 작업을 수행합니다.
sgcGridColumnCaption은 그룹 패널과 그룹 행에서 열의 캡션을 반환합니다. Caption이 있으면 그것을, 없으면 FieldName을 반환합니다.

예제

참고 항목

- 헤드리스 렌더링
- 테마
- 조건부 서식
- 그룹화
- 고정 열
- TsgcGridLineView

TsgcGridFormatRule

sgcGrid_Format 유닛에 있는 조건부 서식 규칙이며, TsgcGridFormatRules의 항목입니다.

속성

속성	형식	설명
FieldName	string	값을 검사할 열입니다.
Operator	TsgcGridFormatOperator	gfoEqual, gfoNotEqual, gfoGreater, gfoGreaterEqual, gfoLess, gfoLessEqual, gfoBetween, gfoContains, gfoStartsWith 또는 gfoIsEmpty.
Value, Value2	string	비교할 값과 gfoBetween의 상한(포함)입니다.
BackColor, ForeColor	TsgcUIColor	일치하는 셀의 색상입니다. 0이면 변경하지 않습니다.
Bold	Boolean	일치하는 셀의 텍스트를 굵게 표시합니다.
ApplyToRow	Boolean	FieldName의 값이 일치하면 행의 모든 셀에 스타일을 적용합니다.
Enabled	Boolean	기본값은 True입니다.
StopOnMatch	Boolean	일치하면 해당 셀의 평가를 끝냅니다.

메서드

메서드	반환값	설명
Matches(aValue, aDataType)	Boolean	값을 검사합니다.

예제

참고 항목

- 조건부 서식
- TsgcGridFormatRules

TsgcGridFormatRules

그리드와 TsgcGridRenderOptions가 FormatRules로 게시하는 TsgcGridFormatRule 항목의 컬렉션입니다.

메서드

메서드	반환값	설명
Add	TsgcGridFormatRule	빈 규칙을 추가합니다.
Add(aFieldName, aOperator, aValue, aBackColor, aForeColor, aBold)	TsgcGridFormatRule	한 번의 호출로 규칙을 추가합니다. aForeColor와 aBold는 선택 사항입니다.
Prepare, EvaluateRow, ApplyToCell		렌더러가 그리는 동안 사용합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	규칙이 바뀌었습니다.

참고 항목

- 조건부 서식
- TsgcGridFormatRule

IsgcGridDataHost와 IsgcUITable

IsgcGridDataHost

sgcGrid_Data 유닛의 인터페이스로, 메서드는 GetGridData 하나입니다. TsgcGridData를 소유한 모든 구성 요소(TsgcVCLGrid와 TsgcFMXGrid)가 구현합니다. 데이터 소스와 AI 쿼리는 sgcGridResolveData로 자신의 **Grid** 속성에서 데이터를 찾습니다. 먼저 Delphi 7을 포함한 모든 Delphi 버전에서 동작하는 IsgcGridDataHost를 거치고, 그다음 게시된 Data 속성을, Delphi 2010부터는 public Data 속성을 거칩니다. GridData 속성이 할당되어 있으면 항상 Grid보다 우선합니다.

IsgcUITable

sgcUICore의 테이블 인터페이스(sgcUICore_Table 유닛)로, TsgcGridData가 구현하고 그리드가 위임합니다. sgcCharts의 TsgcChartDataSource_Table처럼 다른 제품이 sgcGrid에 의존하지 않고 그리드를 읽을 수 있게 해 줍니다. 행은 뷰 행이고 열은 Columns 인덱스입니다.

메서드	설명
TableRowCount, TableColCount	뷰 행 수와 열 수입니다.
TableColName(aCol)	열의 FieldName입니다.
TableValue(aRow, aCol), TableText(aRow, aCol)	Cells와 CellText입니다.
TableLock, TableUnlock	Lock과 Unlock입니다.
TableVersion	변경될 때마다 증가합니다.
TableAddListener, TableRemoveListener	OnChange 이후 변경이 일어난 스레드에서 발생하는 이벤트를, 데이터를 Sender로 하여 통지합니다.

참조 카운팅은 비활성화되어 있습니다. 데이터는 그리드가 소유하므로, 그리드를 해제하기 전에 인터페이스 참조를 놓으십시오.

예제

참고 항목

- 그리드에서 차트 만들기
- TsgcGridData

TsgcGridSource_DataSet

sgcGrid_Source_DataSet 유닛에서 임의의 TDataSet으로 그리드를 채웁니다(DB 유닛만 사용하며 VCL이나 FMX에 의존하지 않습니다).

속성

속성	형식	설명
Grid	TComponent	채울 그리드입니다. IsgcGridDataHost를 참조하십시오.
GridData	TsgcGridData	그리드 대신 채울 데이터입니다. Public이며 Grid보다 우선합니다.
DataSet	TDataSet	원본 데이터셋입니다.
KeyField	string	텍스트가 행 키가 되는 필드입니다. 키가 있으면 행이 제자리에서 갱신되며 깜박임을 유지하고, 새 키는 추가되고 없어진 키는 삭제됩니다. 키가 없으면 로드할 때마다 행이 교체됩니다.
AutoColumns	Boolean	그리드에 열이 없으면 필드로부터 열을 만듭니다. 기본값은 True입니다.
LiveUpdates	Boolean	데이터셋을 따라갑니다. Post, 삽입, 삭제, 취소, 새로 고침이 그리드를 다시 동기화합니다. Post되지 않은 편집과 스크롤은 무시합니다. 기본값은 True입니다.
Active	Boolean	데이터셋이 열려 있으면 로드하고 계속 따라갑니다.

메서드

메서드	반환값	설명
Load		데이터셋을 지금 로드합니다.
Refresh		Active이면 다시 로드합니다.
FieldTypeToDataType(aFieldType)	TsgcGridDataType	필드 형식에 해당하는 열 형식입니다. 클래스 함수입니다.

예제

참고 항목

- DataSet 바인딩
- JSON과 CSV

TsgcGridFeedSource

sgcGridWS_Source 유닛에 있는 sgcGridWS 피드 소스의 기본 클래스입니다. 모든 메시지를 JSON으로 읽어 행을 키로 그리드에 씁니다. 행에 있는 경로만 기록되므로 메시지가 부분 갱신을 담을 수 있습니다. JSON 숫자는 Double로, 불리언은 Boolean으로, 문자열은 string으로, null은 Null로 기록됩니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

속성

속성	형식	설명
Grid	TComponent	기록할 그리드입니다.
GridData	TsgcGridData	그리드 대신 기록할 데이터입니다. Public이며 Grid보다 우선합니다.
RowsPath	string	메시지 안에서 행의 경로입니다. 행 객체의 배열 또는 행 객체 하나입니다. 비어 있으면 메시지 자체가 행 하나이거나 행의 배열입니다.
KeyPath	string	행 객체 안에서 행 키의 경로이며, 예를 들면 symbol이나 id.value입니다. 키가 없는 행은 건너뛰고 OnError로 보고합니다.
Fields	TStrings	한 줄에 FieldName=json.path 하나이며, 예를 들면 Price=quote.last 또는 Bid=book.bid[0]입니다. =가 없는 줄은 같은 이름의 경로를 읽습니다. 비어 있으면 모든 열을 FieldName과 같은 이름의 경로에서 읽습니다.
DeletePath	string	행 객체 안의 불리언 경로입니다. true, 0이 아닌 숫자 또는 문자열 true이면 해당 키의 행을 삭제합니다.
MaxRowsPerMessage	Integer	한 메시지에서 적용하는 최대 행 수(10000)이며, 0이면 제한이 없습니다. 초과한 행은 버려지고 보고됩니다.
Active	Boolean	False이면 모든 메시지를 무시합니다. 기본값은 True입니다.

이벤트

이벤트	형식	설명
OnRow	TsgcGridFeedRowEvent	키의 행을 수락하거나 건너뜁니다.
OnError	TsgcGridFeedErrorEvent	메시지나 행을 적용할 수 없었습니다.

스레딩

한 메시지의 모든 행은 메시지가 도착한 스레드에서 하나의 Data.BeginUpdate와 EndUpdate 안에서 적용됩니다. OnRow와 OnError도 그 스레드에서 실행되므로, 그곳에서 컨트롤을 건드리기 전에 동기화하십시오.

예제

참고 항목

- TsgcGridFeedSource_WebSocket
- TsgcGridFeedSource_MQTT
- TsgcGridFeedSource_SSE
- 실시간 피드

TsgcGridFeedSource_WebSocket

sgcGridWS_Source_WebSocket 유닛에 있으며, sgcWebSockets의 TsgcWebSocketClient가 받는 메시지를 읽는 TsgcGridFeedSource입니다.
소스가 Active이고 실행 시점에 Client가 할당되어 있는 동안, 클라이언트에 자체 OnMessage 핸들러를 설치하고 기존 핸들러는 그대로 유지합니다. 모든 텍스트 메시지는 먼저 그리드에 적용된 다음 이전 핸들러로 전달되므로 애플리케이션 핸들러도 그대로 호출됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 소멸시키면 이전 핸들러가 복원됩니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

속성

속성	형식	설명
Client	TsgcWebSocketClient	WebSocket 클라이언트입니다.
Grid	TComponent	기록할 그리드입니다.
GridData	TsgcGridData	그리드 대신 기록할 데이터입니다. Public이며 Grid보다 우선합니다.
RowsPath	string	메시지 안에서 행의 경로입니다. 행 객체의 배열 또는 행 객체 하나입니다. 비어 있으면 메시지 자체가 행 하나이거나 행의 배열입니다.
KeyPath	string	행 객체 안에서 행 키의 경로이며, 예를 들면 symbol이나 id.value입니다. 키가 없는 행은 건너뛰고 OnError로 보고합니다.
Fields	TStrings	한 줄에 FieldName=json.path 하나이며, 예를 들면 Price=quote.last 또는 Bid=book.bid[0]입니다. =가 없는 줄은 같은 이름의 경로를 읽습니다. 비어 있으면 모든 열을 FieldName과 같은 이름의 경로에서 읽습니다.
DeletePath	string	행 객체 안의 불리언 경로입니다. true, 0이 아닌 숫자 또는 문자열 true이면 해당 키의 행을 삭제합니다.
MaxRowsPerMessage	Integer	한 메시지에서 적용하는 최대 행 수(10000)이며, 0이면 제한이 없습니다. 초과한 행은 버려지고 보고됩니다.
Active	Boolean	False이면 모든 메시지를 무시합니다. 기본값은 True입니다.

메서드

메서드	반환값	설명
ProcessMessage(aText)	Integer	메시지를 직접 적용하고 적용한 행 수를 반환합니다.

이벤트

이벤트	형식	설명
OnRow	TsgcGridFeedRowEvent	키의 행을 수락하거나 건너뜁니다.
OnError	TsgcGridFeedErrorEvent	메시지나 행을 적용할 수 없었습니다.

예제

참고 항목

- TsgcGridFeedSource
- 실시간 피드

TsgcGridFeedSource_MQTT

sgcGridWS_Source_MQTT 유닛에 있으며, sgcWebSockets의 TsgcWSPClient_MQTT가 받는 메시지를 읽는 TsgcGridFeedSource입니다.
소스가 Active이고 실행 시점에 Client가 할당되어 있는 동안, 클라이언트에 자체 OnMQTTPublish 핸들러를 설치하고 기존 핸들러는 그대로 유지합니다. 토픽이 Topic과 일치하는 모든 게시는 먼저 그리드에 적용된 다음 이전 핸들러로 전달되므로 애플리케이션 핸들러도 그대로 호출됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 소멸시키면 이전 핸들러가 복원됩니다. 소스는 구독하지 않으며, 구독은 애플리케이션이 클라이언트에서 수행합니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

속성

속성	형식	설명
Client	TsgcWSPClient_MQTT	MQTT 클라이언트입니다.
Topic	string	MQTT 토픽 필터입니다. +는 한 레벨과, #는 해당 레벨과 그 아래 모든 레벨과 일치합니다. 비어 있으면 모든 게시를 수락합니다.
Grid	TComponent	기록할 그리드입니다.
GridData	TsgcGridData	그리드 대신 기록할 데이터입니다. Public이며 Grid보다 우선합니다.
RowsPath	string	메시지 안에서 행의 경로입니다. 행 객체의 배열 또는 행 객체 하나입니다. 비어 있으면 메시지 자체가 행 하나이거나 행의 배열입니다.
KeyPath	string	행 객체 안에서 행 키의 경로이며, 예를 들면 symbol이나 id.value입니다. 키가 없는 행은 건너뛰고 OnError로 보고합니다.
Fields	TStrings	한 줄에 FieldName=json.path 하나이며, 예를 들면 Price=quote.last 또는 Bid=book.bid[0]입니다. =가 없는 줄은 같은 이름의 경로를 읽습니다. 비어 있으면 모든 열을 FieldName과 같은 이름의 경로에서 읽습니다.
DeletePath	string	행 객체 안의 불리언 경로입니다. true, 0이 아닌 숫자 또는 문자열 true이면 해당 키의 행을 삭제합니다.
MaxRowsPerMessage	Integer	한 메시지에서 적용하는 최대 행 수(10000)이며, 0이면 제한이 없습니다. 초과한 행은 버려지고 보고됩니다.
Active	Boolean	False이면 모든 메시지를 무시합니다. 기본값은 True입니다.

메서드

메서드	반환값	설명
ProcessPublish(aTopic, aText)	Integer	게시를 직접 적용합니다.

이벤트

이벤트	형식	설명
OnRow	TsgcGridFeedRowEvent	키의 행을 수락하거나 건너뜁니다.
OnError	TsgcGridFeedErrorEvent	메시지나 행을 적용할 수 없었습니다.

예제

참고 항목

- TsgcGridFeedSource
- 실시간 피드

TsgcGridFeedSource_SSE

sgcGridWS_Source_SSE 유닛에 있으며, sgcWebSockets의 TsgcHTTP1Client가 받는 서버 전송 이벤트(server-sent events)의 메시지를 읽는 TsgcGridFeedSource입니다.
소스가 Active이고 실행 시점에 Client가 할당되어 있는 동안, 클라이언트에 자체 OnSSEMessage 핸들러를 설치하고 기존 핸들러는 그대로 유지합니다. 모든 스트림 청크는 먼저 그리드에 적용된 다음 이전 핸들러로 전달되므로 애플리케이션 핸들러도 그대로 호출됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 소멸시키면 이전 핸들러가 복원됩니다. 청크는 이벤트로 분리됩니다. data 줄은 이어 붙여지고, 빈 줄이 이벤트의 끝이며, 하나의 이벤트가 여러 청크에 걸칠 수 있습니다. 애플리케이션이 Client.GetSSE로 스트림을 시작합니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

속성

속성	형식	설명
Client	TsgcHTTP1Client	HTTP 클라이언트입니다.
Grid	TComponent	기록할 그리드입니다.
GridData	TsgcGridData	그리드 대신 기록할 데이터입니다. Public이며 Grid보다 우선합니다.
RowsPath	string	메시지 안에서 행의 경로입니다. 행 객체의 배열 또는 행 객체 하나입니다. 비어 있으면 메시지 자체가 행 하나이거나 행의 배열입니다.
KeyPath	string	행 객체 안에서 행 키의 경로이며, 예를 들면 symbol이나 id.value입니다. 키가 없는 행은 건너뛰고 OnError로 보고합니다.
Fields	TStrings	한 줄에 FieldName=json.path 하나이며, 예를 들면 Price=quote.last 또는 Bid=book.bid[0]입니다. =가 없는 줄은 같은 이름의 경로를 읽습니다. 비어 있으면 모든 열을 FieldName과 같은 이름의 경로에서 읽습니다.
DeletePath	string	행 객체 안의 불리언 경로입니다. true, 0이 아닌 숫자 또는 문자열 true이면 해당 키의 행을 삭제합니다.
MaxRowsPerMessage	Integer	한 메시지에서 적용하는 최대 행 수(10000)이며, 0이면 제한이 없습니다. 초과한 행은 버려지고 보고됩니다.
Active	Boolean	False이면 모든 메시지를 무시합니다. 기본값은 True입니다.

메서드

메서드	반환값	설명
ProcessChunk(aText)	Integer	스트림의 청크를 직접 적용합니다.
ResetStream		일부만 수신한 이벤트를 버립니다.

이벤트

이벤트	형식	설명
OnRow	TsgcGridFeedRowEvent	키의 행을 수락하거나 건너뜁니다.
OnError	TsgcGridFeedErrorEvent	메시지나 행을 적용할 수 없었습니다.

예제

참고 항목

- TsgcGridFeedSource
- 실시간 피드

TsgcGridAIQuery

sgcGridWS_AI 유닛에 있으며, 일상 언어 질문을 그리드의 필터, 정렬, 강조로 바꿉니다. 질문은 sgcWebSockets의 TsgcHTMLAIAssistant가 처리합니다. Assistant에 공급자가 있으면 LLM이, 없으면 규칙 기반 파서가 처리하며, 파서는 price greater than 100, sort by volume descending, highlight change less than 0 같은 질문을 이해합니다. 답변은 무언가를 적용하기 전에 그리드의 열과 대조하여 검사됩니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

속성

속성	형식	설명
Grid	TComponent	질의할 그리드입니다. IsgcGridDataHost를 참조하십시오.
GridData	TsgcGridData	그리드 대신 질의할 데이터입니다. Public이며 Grid보다 우선합니다.
Assistant	TsgcHTMLAIAssistant	답변하는 어시스턴트입니다. Nil이면 네트워크와 API 키 없이 규칙 기반 파서를 사용합니다.
Active	Boolean	기본값은 True입니다.

메서드

메서드	반환값	설명
Ask(aQuestion)	Boolean	이전 Ask가 적용한 내용을 제거한 다음 답변을 적용합니다. 동기식이므로 메인 스레드에서 호출하십시오.
Reset		이전 OnFilterRow와 이전 정렬을 복원하고, 이 구성 요소가 추가한 서식 규칙만 삭제합니다.
LastExplanation	string	적용된 내용과 무시된 내용입니다.

이벤트

이벤트	형식	설명
OnApplied	TNotifyEvent	답변이 적용되었습니다.
OnError	TsgcGridAIErrorEvent	질문에 답하거나 적용할 수 없었습니다.

예제

참고 항목

- 그리드에 질문하기
- 필터링
- 조건부 서식

실시간 갱신과 스레딩

sgcGrid는 계속 바뀌는 행을 위해 만들어졌습니다. 각 행에는 키가 있으며, 쓰기 스레드는 어떤 스레드에서든 Synchronize 없이 필드 이름으로 키에 해당하는 행을 갱신합니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

키로 갱신

UpdateRow는 키에 해당하는 행의 지정한 필드를 설정하며, 키가 새 것이면 행을 삽입합니다(AutoInsert가 False인 경우 제외). UpdateCell은 필드 하나를 설정하고, AppendRow는 열 순서대로 값을 지정하여 행을 추가하며, DeleteRow는 키에 해당하는 행을 제거합니다. 변경된 모든 셀은 변경 시각과 방향(상승, 하락, 변경)을 기록하며, 이것이 변경 시 깜박임을 구동합니다.

다시 그리기

데이터는 쓰기 스레드에서 OnChange를 발생시킵니다. 그리드는 그곳에서 자신을 dirty로 표시하기만 하고 메인 스레드에서 초당 최대 MaxFPS회 다시 그리므로, 초당 1,000건의 갱신도 최대 60프레임의 비용만 듭니다. 선택된 행은 뷰가 바뀔 때마다 키로 다시 찾습니다.

BeginUpdate와 EndUpdate

여러 변경을 BeginUpdate와 EndUpdate로 감싸면 전체에 대해 알림을 한 번만 받습니다.

테일 모드

계속 늘어나는 로그와 피드에서는 **MaxRows**가 0보다 크면 새 행을 삽입할 때 가장 오래된 행을 삭제하고, **TailMode**는 그리드가 맨 아래로 스크롤되어 있는 동안 최신 행을 화면에 유지합니다. 위로 스크롤하면 따라가기를 멈추고, 다시 맨 아래로 스크롤하면 재개합니다.

참고 항목

- TsgcGridData
- 성능
- 실시간 피드

정렬

모든 열에는 SortOrder(gsoNone, gsoAscending 또는 gsoDescending)와, 다중 열 정렬에서 열의 순위인 SortIndex가 있습니다. 정렬은 안정적입니다. 텍스트는 대소문자를 구분하지 않고 비교하고 다른 형식은 수치로 비교하며, null, 빈 값, 변환할 수 없는 값은 항상 맨 앞에 정렬됩니다.

마우스로

헤더를 클릭하면 열이 오름차순, 내림차순, 정렬 안 함 순으로 바뀝니다. Ctrl을 누른 채 클릭하면 현재 정렬을 대체하지 않고 열이 정렬에 추가되며, 헤더에는 정렬된 모든 열의 순위가 표시됩니다. OnSortChange는 변경 후에 발생합니다.

코드에서

실시간 규칙

값 변경은 그 열이 정렬 열이거나 필터가 활성일 때만 뷰를 재정렬합니다. 정렬 대상이 아닌 채 계속 바뀌는 열은 현재 순서를 유지하며 재정렬 비용이 들지 않습니다. 삽입, 삭제, 정렬 또는 필터 변경은 항상 뷰를 다시 만듭니다.

참고 항목

- 필터링
- TsgcGridColumn

필터링

두 가지 필터가 차례로 적용됩니다. **FilterText**는 표시되는 임의의 열에서 서식이 적용된 텍스트에 해당 텍스트가 포함된 행을 대소문자 구분 없이 유지합니다. 그런 다음 통과한 모든 행에 대해 데이터의 **OnFilterRow**가 호출됩니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

OnFilterRow

핸들러는 뷰 행이 아니라 저장소 행을 받습니다. 값은 StoreCells로, 키는 StoreKeyOf로 읽으십시오. 데이터 잠금을 잡은 상태에서 호출되므로, 그리드를 사용하는 다른 스레드를 기다려서는 안 됩니다. 핸들러의 로직이 바뀌면 InvalidateView를 호출하십시오.
필터가 활성화되어 있으면 어떤 값이 바뀌든 행이 뷰에 들어오거나 나갈 수 있습니다. 필터로 제외된 행에 대해 RowOfKey는 -1을 반환합니다.

참고 항목

- 정렬
- 그리드에 질문하기
- TsgcGridData

그룹화

그룹화는 그리드의 행을 그룹 행 아래에 배치하며, 그룹화한 열마다 한 단계가 만들어집니다. 그룹 행에는 해당 단계의 멤버, 포함된 행 수, 그리고 요약이 있으면 그 합계가 표시됩니다. 그룹 행은 펼치고 접을 수 있으며, 중첩된 단계는 GroupIndent DIP(기본값 20)만큼 들여쓰기됩니다. 그룹화는 Community를 포함한 모든 에디션에 들어 있으며, TsgcVCLGrid와 TsgcFMXGrid에서 똑같이 동작합니다.
그룹화는 뷰 상태입니다. 그리드는 그룹 행을 TsgcGridLineView(Lines)에 보관하고, TsgcGridData, 그 뷰 행, IsgcUITable은 그대로 유지됩니다. 따라서 그리드에 바인딩된 차트나 다른 리더는 이전처럼 평평하고 정렬과 필터가 적용된 행을 봅니다. 그룹화한 열이 없으면 그리드는 그룹화하지 않은 그리드와 완전히 똑같이 동작합니다.

열 그룹화

모든 열에는 GroupIndex가 있으며 기본값은 -1입니다. 0 이상의 값은 그 단계에서 해당 열로 그리드를 그룹화합니다. 단계는 GroupIndex 순으로, 그다음 열의 Index 순으로 정렬됩니다. 숨겨진 열도 그룹화할 수 있습니다. GroupBy는 지정한 열(FieldName, 대소문자 구분 없음, 첫 번째가 단계 0)의 GroupIndex를 설정하고 나머지 모든 열에는 -1을 설정하며, 알 수 없는 필드는 무시됩니다. ClearGrouping은 모든 열의 GroupIndex를 -1로 설정합니다. GroupIndex를 바꿔도 데이터 뷰는 다시 만들어지지 않습니다.

- 한 단계의 멤버는 그리드 정렬과 같은 규칙에 따라 첫 번째 행의 형식 값으로 정렬됩니다. 텍스트는 대소문자를 구분하지 않고, 그 밖의 형식은 숫자로 비교하며, 오름차순입니다. 열의 SortOrder가 gsoDescending이면 내림차순입니다.
- Null, 빈 텍스트, 변환되지 않는 값은 빈 멤버가 되어 (blank)로 표시되며 항상 맨 앞에 옵니다.
- 열에 DisplayFormat이 있으면 멤버는 서식이 적용된 셀 텍스트가 됩니다. 예를 들어 DisplayFormat이 yyyy-mm이면 날짜가 월별로 그룹화됩니다.
- 그룹 안에서 행은 그리드 정렬 순서를 유지하며 필터도 계속 적용됩니다.

그룹 패널

ShowGroupPanel을 설정하면 헤더 위에 높이 GroupPanelHeight DIP(기본값 36)의 패널이 표시되며, 그룹화한 단계마다 칩이 하나씩 나옵니다. 아무것도 그룹화하지 않았을 때는 힌트가 표시됩니다. 마우스로 다음을 할 수 있습니다.

- 열 헤더를 패널 위로 4픽셀 넘게 끌면 놓은 위치에서 해당 열로 그룹화됩니다. 헤더는 패널이 표시되는 동안에만 끌 수 있으며, Escape로 끌기를 취소합니다.
- 패널 안에서 칩을 끌면 그 단계가 이동하고, 패널 밖으로 끌면 해당 열의 그룹화가 해제됩니다.
- 칩을 클릭하면 해당 그룹의 정렬 순서가 바뀝니다. 열의 SortOrder가 뒤집히고 OnSortChange가 발생합니다.

TsgcFMXGrid에서 터치 팬은 계속 그리드를 스크롤하고 헤더 끌기를 취소하므로, 터치로 그룹화하려면 코드나 마우스를 사용합니다.

그룹 행

그룹 행에는 셰브론, 캡션, 그리고 SummaryType이 있는 열의 그룹 요약(GroupSummaries, 기본값 True)이 표시됩니다. 캡션은 Region: North (12)처럼 열 캡션, 멤버, 행 수로 구성됩니다. OnGetGroupText로 바꿀 수 있습니다. 이 이벤트는 데이터 잠금을 잡은 상태로 발생하므로 그 안에서는 그리드를 읽기만 하고 데이터에는 쓰지 마십시오. GroupInfo(aGroup)는 그룹의 단계, 부모 그룹, 그룹화한 열, 멤버 텍스트와 값, 행 수, 펼침 상태, 표시 줄을 반환합니다. TsgcGridLineView를 참조하십시오.

펼치기와 접기

GroupsExpanded(기본값 True)는 새 그룹의 상태입니다. 설정해도 현재 그룹의 상태는 유지됩니다. ExpandAll과 CollapseAll은 모든 그룹과 이후에 만들어지는 그룹을 바꿉니다. ExpandGroup과 CollapseGroup은 그룹 하나를 바꿉니다. OnGroupExpandChange는 그룹과 새 상태를 알려 주며, ExpandAll과 CollapseAll 뒤에는 aGroup이 -1입니다. 그룹의 펼침 상태는 라이브 업데이트와 다시 정렬한 뒤에도 유지됩니다.
셰브론을 클릭하거나 그룹 행을 더블 클릭하면 전환됩니다. 그룹 줄에서 사용하는 키는 다음과 같습니다.

키	동작
Right	그룹을 펼치며, 이미 펼쳐져 있으면 첫 번째 자식 줄로 이동합니다.
Left	그룹을 접으며, 이미 접혀 있으면 부모 그룹으로 이동합니다.

+ 와 -	그룹을 펼치고 접습니다.
*	그룹과 그 아래의 모든 그룹을 펼칩니다.
Enter	그룹을 전환합니다.
Ctrl+C	그룹 행의 캡션을 복사합니다.

접기로 선택한 행이 가려지면 보이는 그룹 줄이 포커스를 받습니다.

줄과 선택

행을 그룹화하는 동안 그리드는 표시 줄 단위로 스크롤합니다. TopRow와 세로 스크롤 막대는 줄 수를 세고, FocusedLine은 포커스된 줄이며 없으면 -1입니다. 그 밖의 곳에서 행 인덱스는 여전히 뷰 행입니다(SelectedRow, Cells, OnCellClick). 그룹 줄에서는 SelectedRow가 -1입니다. 선택된 행은 키로 유지되며 다시 그룹화한 뒤에도 다시 찾습니다. GroupCount는 그룹 수를 반환하고 Lines는 전체 줄 뷰를 제공합니다.
그룹화한 열의 값이 라이브로 바뀌면 그 행이 새 그룹으로 이동하며, 패널이나 코드로 그룹화한 열이나 단계가 바뀔 때마다 OnGroupingChange가 발생합니다.

예제

핸들러는 다른 기능 예제와 마찬가지로 작은 클래스에 들어 있습니다.

참고 항목

- 요약
- 고정 열
- 정렬
- TsgcGridLineView
- TsgcGridColumn
- TsgcVCLGrid
- .NET 그룹화와 요약

Summaries

SummaryType이 있는 열은 푸터 행과 모든 그룹 행에 값의 요약을 표시합니다. 요약은 Community를 포함한 모든 에디션에 들어 있습니다.

SummaryType	값
gsuNone	요약 없음(기본값).
gsuSum	숫자 값의 합계.
gsuCount	Null이 아니고 빈 텍스트도 아닌 값의 개수.
gsuAverage	합계를 숫자 값의 개수로 나눈 값.
gsuMin, gsuMax	가장 작은 값과 가장 큰 값.

- Null, Unassigned, 빈 텍스트는 건너뜁니다. 텍스트는 데이터 모델과 마찬가지로 데이터의 FormatSettings로 변환됩니다.
- 텍스트와 불리언 열은 gsuCount만 지원합니다.
- 날짜 열은 Min과 Max를 날짜로 유지하며 Sum과 Average는 없습니다.
- 숫자 값이 없는 요약은 Null이며, 이때 gsuCount는 0입니다.

서식

SummaryFormat은 FormatFloat 또는 FormatDateTime 서식 문자열입니다. 리터럴 텍스트는 큰따옴표 안에 넣으며, 예를 들어 `"Total: "#,##0.00`와 같이 씁니다. 비어 있으면 열의 DisplayFormat을 사용하고, 그것도 없으면 데이터 형식의 기본값을 사용합니다. gsuCount는 `0`을 사용합니다. SummaryType이나 SummaryFormat을 바꾸면 데이터 뷰를 다시 만들지 않고 요약이 새로 고쳐집니다.

푸터와 그룹 행

ShowFooter(기본값 False)를 설정하면 RowHeight 한 줄 높이의 푸터 행이 그리드 아래쪽에 고정되어 표시되며, SummaryType이 있는 모든 열의 요약이 뷰 행을 대상으로 계산되므로 필터가 적용됩니다. 푸터는 가로 스크롤과 고정 열을 따릅니다.
GroupSummaries(기본값 True)를 설정하면 각 그룹의 요약이 그 그룹의 행을 대상으로 그룹 행 안에 오른쪽 정렬로 표시됩니다. 가장 왼쪽의 표시 열에는 항상 캡션이 들어갑니다. 그곳의 요약은 들어갈 공간이 있으면 캡션 뒤에 그려지고, 그렇지 않으면 푸터에만 표시됩니다. 그룹화를 참조하십시오.
FooterSummary(aCol)는 열(Columns 인덱스)의 푸터 값을 반환하며 없으면 Null이고, FooterSummaryText(aCol)는 서식이 적용된 텍스트를 반환합니다. 그룹 값은 Lines.GroupSummary(aGroup, aCol)에서 가져옵니다. 요약은 한 번 계산되고 데이터가 바뀔 때만 다시 계산되므로, 요약을 그리는 프레임에 추가 비용이 들지 않습니다. 가상 모드에서는 모든 행에 대해 OnGetValue를 통해 값을 읽습니다.

예제

그리드 없이도 TsgcGridSummaryAccumulator(유닛 sgcGrid_Summary)는 값을 더하고 Value로 모든 요약 형식을 반환하며, TsgcGridSummaryCache는 TsgcGridData의 푸터 값을 보관하고, sgcGridSummaryText는 위의 규칙으로 값의 서식을 지정합니다. TsgcGridLineView 예제는 sgcGridSummaryText를 사용합니다.

참고 항목

- 그룹화
- TsgcGridColumn
- TsgcGridLineView

- .NET 그룹화와 요약

트리 행

트리 행은 그리드의 행을 계층 구조로 보여 줍니다. 각 행은 키 열을 통해 부모를 지정하며, 그리드는 행을 깊이 우선으로 펼칠 수 있는 트리 노드 아래에 배치하고 한 열에서 들여쓰기합니다. 트리는 메모리에 있는 행, TDataSet에서 읽은 행, 라이브 업데이트로 바뀌는 행 모두에서 TsgcVCLGrid와 TsgcFMXGrid 양쪽에서 동작합니다.
이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.

트리 켜기

그리드의 Tree 속성(TsgcGridTreeOptions, 유닛 sgcGrid_Tree)에 설정이 들어 있습니다. KeyField와 ParentField가 모두 그리드 열을 가리키면 트리 모드가 켜지며, 둘 중 하나를 지우면 꺼집니다. 키 열과 부모 열은 숨겨져 있어도 됩니다.

속성	Default	설명
KeyField		각 행의 키를 담은 열.
ParentField		부모 행의 키를 담은 열. 빈 값이나 Null이면 그 행은 루트가 됩니다.
TreeField		들여쓰기와 트리 버튼을 표시하는 열. 비어 있으면 처음 표시되는 열을 사용합니다.
Indent	16	한 단계의 들여쓰기(DIP).
Expanded	False	한 번도 펼치거나 접지 않은 노드의 상태. 바꾸면 모든 노드가 펼쳐지거나 접힙니다.

행을 연결하는 방법

- 키는 텍스트로, 대소문자를 구분하여 비교합니다. Null과 빈 텍스트는 빈 키입니다. 정수와 소수부가 없는 부동 소수점 수는 정수 텍스트를 사용하므로 5, 5.0, '5'는 같은 키입니다. 그 밖의 부동 소수점 수와 날짜는 데이터의 FormatSettings 텍스트를 사용합니다.
- 부모는 저장된 모든 행에서 찾습니다. 여러 행이 같은 키를 가지면 먼저 저장된 행이 자식을 소유합니다.
- 부모가 비어 있거나, 알 수 없거나(고아), 행 자신인 행은 루트가 됩니다. 순환은 처음 반복되는 행에서 끊어지며 그 행이 루트가 됩니다.
- 형제는 그리드 정렬 순서를 유지하므로 열을 정렬하면 모든 노드의 자식과 루트가 정렬됩니다.
- 필터를 사용하면 부모가 필터로 제외된 보이는 행은 가장 가까운 보이는 조상 아래에 놓이며, 조상이 없으면 루트가 됩니다.
- 가상 데이터(Data.Virtual)는 트리를 표시하지 않습니다.

펼치기와 접기

ExpandNode(aRow)와 CollapseNode(aRow)는 뷰 행 aRow의 노드를 바꾸며, 자식이 없는 행은 그대로 둡니다. 트리 모드에서 ExpandAll과 CollapseAll은 트리 노드와 이후에 만들어지는 노드에 작용합니다. OnTreeExpandChange는 뷰 행과 새 상태를 알려 주며, ExpandAll과 CollapseAll 뒤에는 aRow가 -1입니다. 상태는 키별로 유지되므로 정렬, 필터, 행 다시 읽기, 라이브 업데이트 뒤에도 남습니다. 접기로 선택한 행이 가려지면 가장 가까운 보이는 부모가 선택됩니다.
트리 버튼을 클릭하면 노드가 전환됩니다. 트리 모드에서 행을 선택했을 때 사용하는 키는 다음과 같습니다.

키	동작
Right	노드를 펼치며, 이미 펼쳐져 있으면 첫 번째 자식으로 이동합니다.
Left	노드를 접으며, 이미 접혀 있으면 부모로 이동합니다.
+ 와 -	노드를 펼치고 접습니다.
*	노드와 그 아래의 모든 노드를 펼칩니다.

Ctrl+Right와 Ctrl+Left	행의 디테일 밴드를 펼치고 접습니다. 마스터 디테일을 참조하십시오.

TsgcFMXGrid에서는 트리 버튼을 탭하면 전환되며, Cmd는 Ctrl처럼 동작합니다.

트리와 다른 기능

- 트리가 켜져 있는 동안 그룹화는 무시됩니다. 열의 GroupIndex는 유지되며 트리 모드를 끄면 다시 적용됩니다.
- 디테일 밴드는 트리 모드에서도 동작합니다. 펼쳐진 모든 마스터 행 아래에 밴드가 표시됩니다.
- SaveToXLSX는 트리를 Excel 개요로 씁니다. Excel 내보내기를 참조하십시오.
- 행 인덱스는 뷰 행으로 유지됩니다(SelectedRow, Cells, OnCellClick). Lines.TreeMode는 트리가 켜져 있는지 알려주고, Lines.Tree(TsgcGridTreeBuilder)는 뷰 행에 대한 Depth, ParentRow, HasChildren, ChildCount, DescendantCount를 제공합니다. TsgcGridLineView를 참조하십시오.

예제

메모리에 보관한 조직도입니다. 핸들러는 다른 기능 예제와 마찬가지로 작은 클래스에 들어 있습니다.

참고 항목

- 마스터 디테일
- 그룹화
- Excel 내보내기
- TsgcVCLGrid
- TsgcGridLineView
- 에디션
- .NET 트리 행과 마스터 디테일

마스터 디테일

마스터 디테일 행은 펼쳐진 모든 마스터 행 아래에 디테일 밴드를 표시합니다. 밴드는 고객의 주문이나 송장의 줄처럼 관련된 행을 담은 작은 그리드입니다. 헤더가 있고, 스크롤되며, 독자적으로 정렬됩니다. 밴드는 평면, 그룹화, 트리 모드 어디서나 TsgcVCLGrid와 TsgcFMXGrid 양쪽에서 동작합니다.
이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.

디테일 옵션

그리드의 Detail 속성(TsgcGridDetailOptions, 유닛 sgcGrid_Detail)에 설정이 들어 있습니다.

속성	Default	설명
Active	False	디테일 밴드를 켭니다.
Source	nil	디테일 행을 담는 컴포넌트로, TsgcGridDataStore 또는 그리드입니다. 그리드는 해제 알림으로 이를 감시하므로, 해제된 소스가 있으면 Source가 nil이 됩니다.
MasterKeyField		값을 세부 행에서 찾는 마스터 열입니다.
DetailKeyField		마스터 키를 가진 소스의 열입니다.
Lines	8	그리드 행 단위의 밴드 높이이며 중첩 헤더를 포함합니다. 최솟값은 2입니다.
Columns		디테일 그리드의 열. 비어 있으면 소스 열의 복사본을 사용합니다. 열이 있을 때만 저장됩니다.

행에 디테일 확장기가 표시되려면 Active가 True이고, MasterKeyField가 마스터 열을 가리키며, Source 또는 OnGetDetailData 핸들러가 설정되어 있어야 합니다. 마스터 키가 비어 있는 행은 펼쳐지지 않으며, 가상 데이터에서는 밴드가 표시되지 않습니다.

행을 가져오는 곳

- OnGetDetailData가 먼저, 잠금 없이 발생합니다. Sender는 그리드, aRow는 마스터 뷰 행, aMasterKey는 MasterKeyField 텍스트, aData는 밴드의 데이터입니다. aData를 채우고 aHandled를 True로 설정하면 작성한 내용이 유지되며 Source는 읽지 않습니다. Source가 없으면 밴드 열은 Detail.Columns에서 가져옵니다.
- 그렇지 않으면 밴드는 Source에서 채워집니다. 소스의 필터와 정렬에 관계없이, DetailKeyField가 마스터 키와 일치하는 소스의 저장된 행이 대상입니다. 행은 FieldName을 기준으로 소스 키와 함께 복사됩니다. 소스 필드가 없는 디테일 열은 Null이며, 가상 소스는 행을 반환하지 않습니다.
- 키는 텍스트로, 대소문자를 구분하여 비교합니다. 정수와 소수부가 없는 부동 소수점 수는 정수 텍스트를 사용하므로 5, 5.0, '5'는 같은 키입니다.
- 밴드 데이터는 소스의 FormatSettings를 사용하며, 행이 OnGetDetailData에서만 올 때는 마스터 그리드의 FormatSettings를 사용합니다.

소스가 바뀌면 열려 있는 모든 밴드가 다음 프레임에 다시 채워집니다. 행은 제자리에서 업데이트되므로 바뀐 셀은 마스터 그리드처럼 깜박입니다. OnGetDetailData는 다시 채울 때마다 다시 발생합니다.

펼치기와 접기

ExpandDetail(aRow)와 CollapseDetail(aRow)는 마스터 뷰 행 aRow의 밴드를 열고 닫으며, DetailExpanded(aRow)는 그 상태를 알려 주고, OnDetailExpandChange는 행과 새 상태를 알려 줍니다. DetailData(aRow)는 열린 밴드의 데이터를 반환하며, 접혀 있으면 nil입니다. 읽기 전용이므로 읽는 동안 잠그십시오. 상태는 마스터 키별로 유지되므로 정렬, 필터, 마스터 행 다시 읽기, 라이브 업데이트 뒤에도 남습니다.
확장기를 클릭하면 밴드가 전환됩니다. Down은 밴드의 줄을 건너뜁니다. 행을 선택했을 때 사용하는 키는 다음과 같습니다.

모드	키
평면 또는 그룹화, 행 선택	Right와 Left로 밴드를 펼치고 접습니다.
평면 또는 그룹화, SelectionMode가 gsmCell	Ctrl+Right와 Ctrl+Left로 밴드를 펼치고 접으며, Right와 Left는 계속 셀 사이를 이동합니다.
트리 모드	Ctrl+Right와 Ctrl+Left로 밴드를 펼치고 접으며, Right와 Left는 트리 노드에 작용합니다.

밴드 안

- 밴드 위에서 마우스 휠을 돌리면 밴드가 스크롤됩니다. 밴드의 끝에 이르면 대신 그리드가 스크롤됩니다.
- 중첩 헤더를 클릭하면 밴드의 행이 정렬됩니다. OnSortChange는 발생하지 않습니다.
- 중첩 셀을 클릭하면 그 셀이 선택되고, 마스터 행과 디테일 데이터의 뷰 행 및 Columns 인덱스와 함께 OnDetailCellClick이 발생합니다. 마스터 선택은 이동하지 않습니다.
- 디테일 그리드는 읽기 전용이며 한 단계입니다. 밴드는 SaveToXLSX로 쓰이지 않습니다.

TsgcFMXGrid에서 휠은 마우스 포인터가 밴드 위에 있을 때만 밴드를 스크롤하고, 터치 팬은 계속 그리드를 스크롤하며, 확장기를 탭하면 전환되고, Cmd는 Ctrl처럼 동작합니다.

예제

고객별로 주문 밴드를 표시하며, 데이터는 코드로 채운 TsgcGridDataStore에 있습니다.
같은 밴드를 OnGetDetailData로 작성한 행으로 표시합니다. 예를 들어 고객마다 쿼리를 실행해 가져옵니다.
TDataSet의 주문을 표시하려면 TsgcGridSource_DataSet으로 TsgcGridDataStore에 불러옵니다. TsgcGridDataStore를 참조하십시오.

참고 항목

- TsgcGridDataStore
- 트리 행
- DataSet 바인딩
- TsgcVCLGrid
- 에디션
- .NET 트리 행과 마스터 디테일

선택과 키보드

SelectionMode가 gsmRow이면 행 전체를, gsmCell이면 단일 셀을 선택합니다. SelectedRow와 SelectedCol은 뷰 행과 열 인덱스이고 SelectedKey는 선택된 행의 키입니다. 실시간 갱신으로 행이 이동하면 선택도 그 행을 따라갑니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

키보드

키	동작
Up, Down	이전 또는 다음 행.
Page Up, Page Down	한 페이지 위 또는 아래.
Home, End	첫 번째 또는 마지막 행. 셀 모드에서는 첫 번째 또는 마지막 열이며, Ctrl+Home 또는 Ctrl+End는 첫 번째 또는 마지막 행.
Left, Right	셀 모드에서 이전 또는 다음 열.
Ctrl+C, Ctrl+Insert	선택된 행을 탭으로 구분된 텍스트로 복사하며, 셀 모드에서는 선택된 셀을 복사합니다. macOS에서는 Cmd+C.
F2, Enter, 문자, Space	그리드가 ReadOnly가 아니면 편집을 시작합니다. 제자리 편집을 참조하십시오.

마우스

클릭하면 선택되고, OnCellClick과 OnCellDblClick이 셀을 알려 주며, 휠은 스크롤하고, RowAt과 ColAt은 한 지점 아래의 셀을 찾습니다.

참고 항목

- TsgcVCLGrid
- 열 크기 조정

열 크기 조정

헤더 셀의 오른쪽 가장자리를 끌어 열 크기를 조정합니다. 가장자리 위에서는 커서가 바뀝니다. 너비는 열의 MinWidth(기본값 20) 아래로 내려가지 않습니다. 새 너비는 열의 Width에 저장되며 OnColumnResized가 발생합니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

참고 항목

- TsgcGridColumn
- 선택과 키보드

고정 열

FrozenColumns(기본값 0)는 나머지 열이 가로로 스크롤되는 동안 처음 표시되는 열을 제자리에 고정하므로, 심볼이나 이름 같은 키 열이 계속 보입니다. 헤더, 푸터, 그룹 요약도 같은 경계로 나뉘며, 고정된 가장자리는 1픽셀 테두리 선으로 표시됩니다. 고정 열은 Community를 포함한 모든 에디션에 들어 있습니다.

- FrozenColumns는 표시되는 열을 셉니다. 숨겨진 열은 건너뜁니다.
- 왼쪽의 그룹 셰브론(거터, 그룹화한 단계마다 GroupIndent 하나)도 스크롤되지 않습니다.
- TsgcVCLGrid에서 기본 가로 스크롤 막대는 스크롤되는 부분만 범위로 삼으므로 ScrollX의 최댓값은 바뀌지 않습니다. TsgcFMXGrid도 같은 방식으로 스크롤되는 부분에 대해 가로 썸을 그립니다.
- 키보드 이동은 고정 경계의 오른쪽에서 스크롤된 열을 보이는 범위로 가져옵니다.

렌더러는 Options에서 같은 설정을 제공하므로 헤드리스 PNG와 SVG 출력이 컨트롤과 일치하며, FrozenWidth는 스크롤되지 않는 부분의 너비를 반환합니다.

예제

참고 항목

- 열 크기 조정
- 그룹화
- 헤드리스 렌더링
- TsgcGridRenderer

변경 시 깜박임

FlashChanges가 True(기본값)이면 값이 바뀐 셀에 변화 방향의 색이 입혀집니다. 값이 올라가면 FlashUpColor, 내려가면 FlashDownColor, 그 외의 변경은 FlashChangedColor입니다. 색조는 **FlashDuration** 밀리초(기본값 800) 동안 선형으로 서서히 사라집니다.
FlashMode가 fmBackground이면 셀 배경에, fmText이면 텍스트에 색조를 입힙니다. 색상이 0이면 테마 기본값을 사용합니다. 녹색, 빨간색, 강조색이며 배경에는 반투명, fmText에는 읽기 쉬운 텍스트 색상이 쓰이고 다크 테마에서는 더 밝습니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.
깜박임이 사라지는 동안 그리드는 MaxFPS까지 계속 프레임을 요청합니다. 선택은 깜박임보다, 깜박임은 서식 규칙보다 우선합니다.

참고 항목

- 조건부 서식
- 테마

조건부 서식

그리드의 **FormatRules**는 값이 조건을 만족하는 셀에 색을 입힙니다. 규칙은 열 FieldName의 값을 Value(gfoBetween이면 Value2도)와 그 열의 DataType에 따라 비교합니다. 숫자는 수치로, 날짜는 날짜로(Value는 ISO 8601 또는 날짜 일련번호), 불리언은 true 또는 false로, 텍스트는 대소문자를 구분하지 않고 비교합니다. gfoContains와 gfoStartsWith는 항상 텍스트를 비교하고, gfoIsEmpty는 null 또는 공백 값과 일치하며, 빈 셀은 순서 연산자와 일치하지 않습니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

우선순위

- 규칙은 컬렉션 순서대로 평가됩니다. 일치하는 활성 규칙은 모두 0이 아닌 BackColor와 ForeColor를 적용하며 나중에 일치한 규칙이 앞선 규칙을 덮어쓰고, Bold는 누적됩니다.
- StopOnMatch가 설정된 규칙이 일치하면 해당 셀의 평가가 끝납니다.
- ApplyToRow 규칙은 자신의 열을 검사하고, 일치하면 행의 모든 셀에 스타일을 적용합니다.
- 그리드가 그릴 때는 선택이 깜박임보다, 깜박임이 규칙보다, 규칙이 교대 행 색조보다 우선합니다.
- OnGetCellStyle은 그려지는 모든 셀에 대해 마지막으로 발생합니다. aFore와 aBold에는 지금까지의 결과가 담겨 있고, aBack은 그 위에 추가로 칠하는 색입니다.

참고 항목

- TsgcGridFormatRule
- 변경 시 깜박임
- 그리드에 질문하기

제자리 편집

제자리 편집에는 SGC_GRID_EDIT가 필요하며, Community 에디션은 이를 정의하지 않습니다. 에디션을 참조하십시오.

편집 시작

ReadOnly를 False로 설정합니다. 열은 Editable이 True인 동안 편집할 수 있습니다. F2, Enter, 문자 입력(텍스트를 대체함), 더블 클릭 중 하나로 셀 위에 TEdit가 열리며, 열에 PickList가 있으면 TComboBox가 열립니다. StartEdit는 코드에서 편집기를 열고, OnBeforeEdit는 이를 거부할 수 있습니다.

확정과 취소

Enter는 확정하고, Tab과 Shift+Tab은 확정한 뒤 다음 또는 이전 편집 가능한 셀로 이동하며, Escape는 취소합니다. 포커스를 잃으면 확정하고, 텍스트가 올바르지 않으면 취소합니다. CommitEdit와 CancelEdit는 코드에서 같은 동작을 합니다.

변환과 검증

텍스트는 Data.FormatSettings를 사용하여 열의 DataType으로 변환됩니다. 변환되지 않는 텍스트는 편집기를 열어 둔 채 OnEditError를 발생시키며, aMessage는 바꿀 수 있습니다. OnValidate는 변환된 값을 받아 바꾸거나 거부할 수 있습니다. 그런 다음 값은 Data.UpdateCell로 기록되므로 변경 추적, 깜박임, 정렬이 적용되며, OnEdited가 이전 값과 새 값을 알려 줍니다.

불리언

gdtBoolean 열에는 편집기가 없습니다. Space, 클릭, F2, Enter 또는 StartEdit가 값을 토글합니다.

편집 중 실시간 갱신

실시간 갱신으로 뷰가 재정렬되거나 필터링되면, 편집기는 키로 다시 찾은 자신의 행과 함께 이동합니다. 행이 삭제되거나 필터로 제외되면 편집은 취소됩니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

참고 항목

- TsgcGridColumn
- TsgcVCLGrid

가상 모드

가상 모드에서 그리드는 행을 저장하지 않습니다. **VirtualRowCount**가 행 수를 정하고, 렌더러가 그릴 때 데이터의 **OnGetValue**가 행과 열의 값을 반환합니다. 보이는 행만 그리므로 행이 백만 개인 그리드도 오십 개인 그리드와 같은 비용이 듭니다.
가상 행에는 키가 없으므로 뷰는 순서대로의 행입니다. 정렬, 필터링, 변경 추적이 없고 KeyOf와 RowOfKey는 아무것도 반환하지 않습니다. 행 수가 바뀌면 VirtualRowCount를 변경하십시오.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

참고 항목

- TsgcGridData
- 성능

JSON과 CSV

sgcGrid_IO 유닛은 TsgcGridData를 문자열, 스트림, 파일에서 JSON 또는 CSV로 로드하고 저장합니다. 프레임워크에 독립적이며 Linux64에서도 빌드됩니다.

로드

- 로드는 행을 교체하며 하나의 BeginUpdate와 EndUpdate 안에서 실행됩니다. 키 필드를 지정하면 행의 키는 그 필드의 텍스트이며, 반복된 키는 앞선 행을 갱신합니다.
- aAutoColumns가 True이고 아직 열이 없으면 입력으로부터 열이 만들어지며 DataType은 값으로 추정합니다. 기존 열은 유지되며 FieldName으로 대응시킵니다.
- JSON 입력은 객체의 배열입니다. price.last처럼 점이 있는 FieldName은 중첩된 멤버를 읽습니다.
- CSV는 RFC 4180을 따르며 첫 번째 레코드가 FieldName을 담습니다. aInvariant가 True(기본값)이면 숫자를 점으로, 날짜를 ISO 8601로 읽고 쓰며, False이면 데이터의 FormatSettings를 사용합니다.

저장

GridSaveToJSON과 GridSaveToCSV는 정렬 및 필터링된 뷰 순서대로 행을 쓰며, 열도 순서대로 쓰고 aVisibleOnly가 True이면 표시되는 열만 씁니다. 텍스트는 BOM 없는 UTF-8입니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

참고 항목

- DataSet 바인딩
- TsgcGridData

Excel 내보내기

SaveToXLSX는 외부 라이브러리 없이 그리드를 Excel 통합 문서(Office Open XML, 워크시트 하나)로 스트림이나 파일에 씁니다. 통합 문서에는 그리드에 표시되는 내용이 그대로 담깁니다. 표시되는 열은 표시 순서대로 현재 너비와 숫자 서식을 갖고, 고정된 헤더와 FrozenColumns, 접을 수 있는 개요로 된 그룹이나 트리, 수식으로 된 요약이 포함됩니다.
이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.

그리드 내보내기

메서드	설명
SaveToXLSX(aStream; aSheetName = 'Sheet1')	통합 문서를 스트림에 씁니다.
SaveToXLSX(aFileName; aSheetName = 'Sheet1')	통합 문서를 파일에 씁니다.

UI 스레드에서 호출하십시오. 시트 이름에서는 Excel이 거부하는 문자가 제거되고(_로 바뀝니다) 31자로 잘립니다. 1,048,576행 또는 16,384열을 넘는 시트는 아무것도 쓰기 전에 EsgcGridXLSXError를 발생시킵니다.

시트의 내용

- 1행은 굵은 헤더이며, 처음 FrozenColumns개 열과 함께 고정됩니다. 열 너비는 화면에서의 너비입니다.
- 접혀 있는지 여부와 관계없이 모든 그룹 행과 뷰 행이 쓰입니다. 접힌 그룹 안의 행은 숨겨지고 그룹 행은 접힘으로 표시되므로, Excel은 그리드에 보이는 대로 개요를 엽니다. 그룹 행은 자신의 행 위에 놓입니다.
- 그룹 행은 캡션(Region: North (12))을 첫 번째 열에, 또는 요약이 없는 첫 번째 열에 담습니다. OnGetGroupText는 사용되지 않습니다.
- 트리 행에는 깊이에 맞는 개요 수준과 트리 열의 들여쓰기가 지정됩니다. 개요 수준은 Excel의 최댓값인 7에서 멈춥니다.
- 디테일 밴드는 내보내지 않습니다.
- 숫자는 숫자로, 날짜는 Excel 일련 날짜로, 불리언은 불리언으로, 텍스트는 제어 문자를 제외한 공유 문자열로 쓰입니다.
- 열의 DisplayFormat은 Excel 숫자 서식이 됩니다. FormatFloat 서식 문자열은 그대로 전달되며, 날짜 서식 문자열에서는 nn이 mm으로, z가 000으로, am/pm이 AM/PM으로 바뀝니다. DisplayFormat이 없으면 정수 열은 0, 통화 열은 #,##0.00, 날짜 열은 yyyy-mm-dd hh:mm:ss를 사용합니다.

Summaries

그룹 행의 각 요약 열에는 그 그룹의 행에 대한 SUBTOTAL 수식이 들어 있고 그리드의 값이 셀에 캐시되므로, Excel이 다시 계산하기 전에도 파일에 합계가 표시됩니다. 표시되는 열에 요약이 있으면 푸터 행에 모든 데이터 행에 대한 SUBTOTAL이 들어 있습니다. 푸터 행은 ShowFooter의 값과 관계없이 쓰입니다.

SummaryType	SUBTOTAL	열 형식
gsuSum	9	숫자
gsuAverage	1	숫자
gsuMin	5	숫자와 날짜
gsuMax	4	숫자와 날짜
gsuCount	3	모든 형식(비어 있지 않은 셀)

이 코드는 숨겨진 행도 세므로 접힌 그룹도 합계를 유지하며, SUBTOTAL은 안쪽 그룹의 중첩된 SUBTOTAL 셀을 무시합니다.

TsgcGridXLSXWriter

SaveToXLSX 뒤에 있는 라이터는 유닛 sgcGrid_XLSX에 있으며 그리드가 필요하지 않습니다. 모든 TsgcGridData를 쓸 수 있습니다.

멤버	설명
SaveToStream(aData, aLines, aStream), SaveToFile(aData, aLines, aFileName)	aData의 통합 문서를 씁니다. aLines는 그룹화, 트리, 펼침 상태를 제공하며, nil이면 자체 줄 뷰를 사용합니다.
SheetName	워크시트의 이름. 기본값은 'Sheet1'입니다.
FrozenColumns	보이는 상태로 유지할 앞쪽의 표시 열.
SetColumnWidth(aCol, aDIPs)	열 aCol(Columns 인덱스)의 너비(DIP)이며 Width 대신 사용됩니다. 음수 값은 Width로 되돌립니다.
Summaries	그룹과 푸터의 SUBTOTAL 셀을 씁니다. 기본값은 True입니다.
FileTime	zip 항목의 날짜와 시간. 라이터를 만들 때는 Now입니다. 같은 데이터와 같은 FileTime에서는 항상 같은 바이트가 나옵니다.
sgcGridExcelNumberFormat(aDataType, aDisplayFormat)	열 형식과 DisplayFormat에 대한 Excel 숫자 서식 코드.

예제

참고 항목

- JSON과 CSV
- 요약
- 그룹화
- 트리 행
- 고정 열
- 에디션
- .NET Excel 내보내기

DataSet 바인딩

TsgcGridSource_DataSet은 임의의 TDataSet으로 그리드를 채웁니다. KeyField가 있으면 그리드를 다시 로드하지 않고 동기화합니다. 변경된 행은 제자리에서 갱신되며 깜박이고, 새 키는 추가되고 없어진 키는 삭제됩니다. LiveUpdates를 사용하면 그리드가 데이터셋에서 Post된 모든 변경을 따라갑니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.
데이터셋은 DisableControls로 순회하며 현재 레코드는 복원됩니다. 데이터셋을 닫아도 행은 유지됩니다. DataSetGrid 데모는 TClientDataSet을 이용한 바인딩을 보여 줍니다.

참고 항목

- TsgcGridSource_DataSet
- JSON과 CSV
- 데모

실시간 피드

sgcGridWS 피드 소스는 sgcWebSockets 클라이언트의 JSON 메시지를 그리드에 바로 기록합니다. TsgcGridFeedSource_WebSocket, TsgcGridFeedSource_MQTT, TsgcGridFeedSource_SSE가 있습니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

매핑 규칙

속성	규칙
RowsPath	메시지에서 행이 있는 위치입니다. 비어 있으면 메시지가 행 객체 하나이거나 행의 배열입니다.
KeyPath	행 안에서 키의 경로이며, 예를 들면 symbol이나 id.value입니다. 문자열, 숫자, 불리언을 사용할 수 있습니다.
Fields	FieldName=json.path 형식의 줄입니다. =가 없는 줄은 같은 이름의 경로를 읽고, 비어 있으면 모든 열을 FieldName과 같은 이름의 경로에서 읽습니다. 있는 경로만 기록되므로 부분 갱신이 동작합니다.
DeletePath	true이면 해당 키의 행을 삭제하는 불리언 경로입니다.

WebSocket

MQTT

서버 전송 이벤트

다른 곳에서 온 메시지

한 메시지의 행은 클라이언트의 스레드에서 하나의 BeginUpdate와 EndUpdate 안에서 적용되며, 데이터가 스레드 안전하므로 문제가 없습니다. LiveFeed 데모는 프로세스 내 서버로 WebSocket 소스를 보여 줍니다.

참고 항목

- TsgcGridFeedSource
- 실시간 갱신과 스레딩

그리드에 질문하기

TsgcGridAIQuery를 사용하면 사용자가 price greater than 100 sort by volume descending highlight change less than 0 같은 일상 언어 질문으로 그리드를 필터링, 정렬, 강조할 수 있습니다.
이 기능은 sgcGridWS 브리지의 일부입니다. 유닛에는 sgcWebSockets 소스 또는 패키지가 필요하며, Community 에디션에는 포함되지 않습니다.

답변이 적용되는 방식

- 필터는 AND로 결합되어 데이터의 OnFilterRow를 통해 적용됩니다. 이미 할당된 핸들러는 유지되며 먼저 호출됩니다.
- 정렬은 열의 SortOrder와 SortIndex를 대체합니다.
- 강조는 ApplyToRow를 설정한 채 그리드의 FormatRules에 추가됩니다.
- 그룹화와 행 수 제한은 그리드가 지원하지 않으므로 무시되며 LastExplanation에 보고됩니다.
- 모든 Ask는 먼저 Reset을 호출하므로 질문이 쌓이지 않습니다. Reset은 이전 필터 핸들러와 정렬을 복원하고 쿼리가 추가한 규칙만 제거합니다.

LLM 사용

공급자가 있는 TsgcHTMLAIAssistant를 할당하면 LLM으로 답변합니다. Ask는 동기식이며 공급자가 답변하는 동안 블록되므로 메인 스레드에서 호출하십시오. Assistant가 없으면 AskYourGrid 데모처럼 네트워크와 API 키 없이 규칙 기반 파서가 답변합니다.

참고 항목

- TsgcGridAIQuery
- 필터링
- 조건부 서식

그리드에서 차트 만들기

그리드의 데이터는 sgcUICore의 테이블 인터페이스인 IsgcUITable을 구현하며 그리드가 이를 위임합니다. sgcCharts 데이터 소스 **TsgcChartDataSource_Table**은 이런 테이블을 읽습니다. Table을 그리드로, XColumn과 YColumn을 열의 FieldName으로 설정하십시오. 피드의 갱신이든 제자리 편집이든 그리드 행의 모든 변경이 중간 코드 없이 차트에 반영되며, 차트는 그리드의 정렬과 필터를 따릅니다.
이 예제에는 sgcCharts 유닛이 필요합니다. GridAndChart 데모는 그리드와 차트를 두 개의 sgcDashboard 타일에 배치합니다.

참고 항목

- IsgcGridDataHost와 IsgcUITable
- 데모

테마

그리드는 sgcUICore의 라이트 및 다크 테마 토큰을 사용합니다. Theme은 utLight(기본값) 또는 utDark이고, AccentColor는 선택과 변경 깜박임에 쓰이며, 밀도는 FontSize, RowHeight, HeaderHeight, AlternateRows, 격자선으로 조정합니다. 색상은 TsgcUIColor ARGB 값이며 sgcUIARGB로 만듭니다.
sgcDashboard 타일에 놓은 TsgcVCLGrid는 게시된 Theme 속성을 통해 대시보드의 테마를 따릅니다.
예제는 **TsgcVCLGrid**를 받는 일반 프로시저입니다. 이벤트 핸들러는 TsgcVCLGrid에 선언된 작은 클래스 TGridHandlers의 메서드입니다. 같은 코드가 TsgcFMXGrid에서도 동작합니다.

참고 항목

- 변경 시 깜박임
- 헤드리스 렌더링

헤드리스 렌더링

렌더러에는 창이 필요하지 않습니다. 그리드의 SaveToPNG와 SaveToSVG는 현재 뷰를 씁니다. sgcGridSaveToPNG와 sgcGridSaveToSVG 프로시저 또는 TsgcGridRenderer는 독립 TsgcGridData를 렌더링하며, Linux64에서도 가능하여 보고서, 이메일, 서버가 생성하는 이미지에 활용할 수 있습니다. 소개의 이미지는 아래 예제로 만들었습니다.

그리드에서

참고 항목

- TsgcGridRenderer
- 테마

.NET 개요

sgcGrid .NET은 Delphi sgcGrid 라이브러리를 감싼 래퍼가 아니라 관리형 C#으로 옮긴 포트입니다. C# 클래스는 **esegece.sgcGrid** 네임스페이스에 있고 Delphi 이름을 그대로 유지하므로, **TsgcGridData**, **TsgcGridColumns**, **TsgcGridColumn**, **TsgcGridFormatRules**, **TsgcGridRenderer**, **TsgcGridSource_DataSet**은 두 언어에서 똑같이 읽힙니다. 그리드 컨트롤은 Delphi 명명 패턴을 따라 WinForms용은 **TsgcWinFormsGrid**, WPF용은 **TsgcWpfGrid**, Avalonia용은 **TsgcAvaloniaGrid**입니다. 세 가지 모두 **sgcUICore**의 .NET 포트로 그립니다. 캔버스, 글꼴, 라이트 및 다크 테마가 eSeGeCe 비주얼 스위트의 다른 제품과 같으므로, 그리드는 어떤 호스트에서도, 그리고 헤드리스 PNG 및 SVG 출력에서도 같게 보입니다.
그리드는 실시간 데이터를 위해 만들어졌습니다. 행 저장소는 키로 관리되고 스레드에 안전하므로, 워커 스레드가 행을 갱신, 추가, 삭제하는 동안에도 그리드는 보이는 행만 제어된 프레임 속도로 그립니다. 변경된 셀은 깜박이고, 서식 규칙은 셀이나 행 전체에 색을 입히며, 그리드는 정렬, 필터링, 가상 모드 동작, JSON 및 CSV 내보내기를 지원합니다. Registered 에디션에서는 사용자가 셀을 제자리에서 편집할 수 있고, sgcWebSockets 브리지가 WebSocket, MQTT, SSE 연결에서 그리드로 데이터를 공급하며, sgcHTML 브리지가 행에 대한 자연어 질문에 답합니다.

패키지

포트는 NuGet 패키지로 나뉘어 있으므로, 프로젝트는 사용하는 것만 가져옵니다.

PackageId	내용	대상 프레임워크
esegece.sgcGrid	모델(**TsgcGridColumns**, **TsgcGridData**, **TsgcGridFormatSettings**), 정렬과 필터링, 가상 모드, 깜박임 및 서식 규칙(**TsgcGridFormatRules**), 헤드리스 SVG 출력을 갖춘 렌더러 **TsgcGridRenderer**, JSON 및 CSV 가져오기와 내보내기(**sgcGrid_IO**), DataTable 소스 **TsgcGridSource_DataSet**. 네이티브 종속성은 없습니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcGrid.Rendering	헤드리스 PNG 출력(**SaveToPNG**). SkiaSharp를 사용합니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcGrid.WinForms	**TsgcWinFormsGrid**.	net472, net8.0-windows, net9.0-windows
esegece.sgcGrid.Wpf	**TsgcWpfGrid**.	net472, net8.0-windows, net9.0-windows
esegece.sgcGrid.Avalonia	**TsgcAvaloniaGrid**. Avalonia 11.3을 사용합니다.	net8.0, net9.0
esegece.sgcGrid.sgcWebSockets	실시간 피드 브리지: **TsgcGridFeedSource**, **TsgcGridFeedSource_WebSocket**, **TsgcGridFeedSource_MQTT**, **TsgcGridFeedSource_SSE**. esegece.sgcWebSockets 패키지에 종속됩니다. Registered 에디션 전용입니다.	netstandard2.0, net8.0, net9.0
esegece.sgcGrid.sgcHTML	AI 쿼리 브리지: **TsgcGridAIQuery**와 **TsgcGridAIFilterItem**. esegece.sgcHTML 패키지의 **TsgcHTMLAIAssistant**를 사용합니다. Registered 에디션 전용입니다.	netstandard2.0, net8.0, net9.0

각 호스트 패키지는 esegece.sgcGrid.Rendering과 같은 호스트용 sgcUICore 패키지에 종속되므로, 호스트 패키지만 추가하면 충분합니다.

에디션

에디션	패키지 ID	포함 내용
Registered	esegece.sgcGrid, .Rendering, .WinForms, .Wpf, .Avalonia, .sgcWebSockets, .sgcHTML	모든 것을 포함합니다. 세 가지 그리드, 실시간 갱신, 깜박임 및 서식 규칙, 정렬, 필터링, 가상 모드, JSON 및 CSV, DataTable 소스, 헤드리스 출력, 제자리 편집, 두 브리지가 해당됩니다.
Community	같은 ID에 .Community 접미사를 붙입니다. 예: esegece.sgcGrid.WinForms.Community. Community용 브리지 패키지는 없습니다.	제자리 편집 (**StartEdit**, **CommitEdit**, **CancelEdit**, 이벤트 **OnBeforeEdit**, **OnValidate**, **OnEdited**, **OnEditError**) 과 두 브리지를 제외한 모든 것입니다.

Delphi 에디션과 같은 구분입니다. 에디션을 참고하십시오.

호스트

호스트	Delphi 대응 요소	플랫폼
WinForms	TsgcVCLGrid	Windows. .NET Framework 4.7.2와 .NET 8, 9에서 동작합니다.
WPF	TsgcFMXGrid	Windows. .NET Framework 4.7.2와 .NET 8, 9에서 동작합니다.
Avalonia	TsgcFMXGrid	Windows, Linux, macOS. .NET 8, 9에서 동작합니다.
헤드리스	TsgcGridRenderer	Linux 서버를 포함한 .NET Standard 2.0의 모든 플랫폼.

참고 항목

- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET DataTable 바인딩, JSON 및 CSV
- .NET 제자리 편집
- .NET 실시간 피드
- .NET 그리드에 질문하기

- .NET 성능
- .NET 데모

NuGet에서 설치

패키지는 nuget.org에 있습니다. 호스트 패키지를 추가하면 코어 패키지와 렌더링 패키지가 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcGrid.WinForms
dotnet add package esegece.sgcGrid.Wpf
dotnet add package esegece.sgcGrid.Avalonia
```

실시간 피드와 자연어 쿼리용 브리지를 추가합니다. esegece.sgcWebSockets와 esegece.sgcHTML 패키지가 함께 설치됩니다.

```
dotnet add package esegece.sgcGrid.sgcWebSockets
dotnet add package esegece.sgcGrid.sgcHTML
```

Community 에디션

```
dotnet add package esegece.sgcGrid.WinForms.Community
dotnet add package esegece.sgcGrid.Wpf.Community
dotnet add package esegece.sgcGrid.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcGrid.WinForms" Version="2026.10.0" />
</ItemGroup>
```

한 프로젝트에서는 Registered 패키지나 Community 패키지 중 하나만 참조하고, 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있기 때문입니다. 편집 API를 사용하는 코드는 Community 패키지에서는 컴파일되지 않습니다. 한 프로젝트에서 두 에디션을 모두 빌드할 때는 `#if !SGC_COMMUNITY`로 감싸십시오.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서는 `xmlns:grid="clr-namespace:esegece.sgcGrid;assembly=esegece.sgcGrid.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: 일반적인 Avalonia 데스크톱 애플리케이션과 마찬가지로 Avalonia 11.3의 **Avalonia.Desktop** 및 **Avalonia.Themes.Fluent** 패키지를 사용하는 `net8.0` 또는 `net9.0` 프로젝트. XAML에서는 `xmlns:grid="using:esegece.sgcGrid"`로 네임스페이스를 매핑합니다.
- 헤드리스: 모델, DataTable 소스, JSON, CSV, SVG에는 esegece.sgcGrid만으로 충분합니다. PNG에는 esegece.sgcGrid.Rendering을 추가하며, 이 패키지는 SkiaSharp와 Windows 및 Linux용 네이티브 자산을 함께 가져옵니다.
- 브리지: esegece.sgcGrid.sgcWebSockets와 esegece.sgcGrid.sgcHTML은 netstandard2.0, net8.0, net9.0을 대상으로 하므로 .NET Framework 4.7.2를 포함한 모든 호스트에서 동작합니다.

C#에서는 모든 호스트에 `using esegece.sgcGrid;`가 필요하고, **TsgcUITheme** 등 테마 형식에는 `using esegece.sgcUICore;`도 필요합니다. 실시간 피드 소스에는 소켓 클라이언트용으로 `using esegece.sgcWebSockets;`도 필요합니다.

참고 항목

- .NET 개요
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 실시간 피드
- .NET 그리드에 질문하기

- .NET 데모

WinForms 빠른 시작

이 예제는 시세 그리드가 있는 WinForms 폼을 만듭니다. 워커 스레드가 1초에 네 번 행을 갱신하고, 변경된 셀은 깜박이며, 서식 규칙이 변동이 음수인 행을 연한 빨간색으로 칠하고, 체크 박스로 그리드의 라이트 테마와 다크 테마를 전환합니다.

단계

- `net8.0-windows` WinForms 프로젝트를 만들고 **esegece.sgcGrid.WinForms** 패키지를 추가합니다. NuGet에서 설치를 참고하십시오.
- **TsgcWinFormsGrid**를 만들고 도킹합니다. **Columns.Add**로 열을 추가합니다. 필드 이름, 캡션, **TsgcGridDataType**을 지정하고, 숫자와 날짜에는 **DisplayFormat**을 설정합니다.
- **Data.AppendRow**로 행을 추가합니다. 첫 번째 인수는 행 키이고, 값은 열 순서대로 이어집니다.
- **Data.UpdateRow** 또는 **Data.UpdateCell**로 어떤 스레드에서든 키로 행을 갱신합니다. **BeginUpdate**와 **EndUpdate**는 많은 변경을 한 번의 다시 그리기로 묶습니다.
- **FormatRules**에 규칙을 추가해 셀에 색을 입힙니다. **ApplyToRow**를 사용하면 행 전체에 색을 입힐 수 있습니다.

예제

```
using System;
using System.Drawing;
using System.Threading;
using System.Windows.Forms;
using esegece.sgcGrid;
using esegece.sgcUICore;

namespace GridQuickStartWinForms
{
    internal class MainForm : Form
    {
        private static readonly string[] Symbols = { "AAPL", "MSFT", "NVDA", "AMZN", "GOOG", "META" };
        private readonly TsgcWinFormsGrid Grid = new TsgcWinFormsGrid();
        private readonly CheckBox chkDark = new CheckBox();
        private readonly Random FRandom = new Random();
        private System.Threading.Timer tmrFeed;

        public MainForm()
        {
            Text = "sgcGrid Quick Start";
            ClientSize = new Size(720, 360);

            Grid.Dock = DockStyle.Fill;
            Grid.Columns.Add("symbol", "Symbol");
            Grid.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat).DisplayFormat = "#,##0.00";
            Grid.Columns.Add("change", "Change", TsgcGridDataType.gdtFloat).DisplayFormat = "+0.00;-0.00;0.00";
            Grid.Columns.Add("volume", "Volume", TsgcGridDataType.gdtInteger).DisplayFormat = "#,##0";
            Grid.FlashChanges = true;
            Grid.FlashDuration = 800;
            // the whole row in light red while the change is negative
            Grid.FormatRules.Add("change", TsgcGridFormatOperator.gfoLess, "0", 0x30E53935).ApplyToRow = true;
            Grid.OnCellClick += (Sender, aRow, aCol) => Text = "sgcGrid Quick Start: " + Grid.SelectedKey;

            // one row per symbol, the key is the symbol and the values follow the column order
            foreach (string vSymbol in Symbols)
                Grid.Data.AppendRow(vSymbol, vSymbol, 100.0, 0.0, 0);

            chkDark.Text = "Dark theme";
            chkDark.Dock = DockStyle.Top;
            chkDark.CheckedChanged += (s, e) =>
                Grid.Theme = chkDark.Checked ? TsgcUITheme.utDark : TsgcUITheme.utLight;
```

```
            Controls.Add(Grid);
            Controls.Add(chkDark);

            // TsgcGridData is thread safe: a worker thread updates the rows and the grid
            // repaints on the UI thread, paced by MaxFPS
            tmrFeed = new System.Threading.Timer(s => UpdateQuotes(), null, 250, 250);
        }

        private void UpdateQuotes()
        {
            Grid.Data.BeginUpdate();
            try
            {
                foreach (string vSymbol in Symbols)
                {
                    double vChange = Math.Round(FRandom.NextDouble() * 4 - 2, 2);
                    Grid.Data.UpdateRow(vSymbol, new[] { "price", "change", "volume" },
                        new object[] { 100.0 + vChange, vChange, FRandom.Next(1000, 90000) });
                }
            }
            finally
            {
                Grid.Data.EndUpdate();
            }
        }

        protected override void OnFormClosed(FormClosedEventArgs e)
        {
            tmrFeed.Dispose();
            base.OnFormClosed(e);
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

색은 $AARRGGBB 값입니다. FF보다 작은 알파는 셀을 배경 위에 덧입혀 색조를 줍니다. 그리드는 데이터가 아무리 빠르게 바뀌어도 초당 최대 **MaxFPS**회만 다시 그립니다. 실시간 갱신과 스레딩과 .NET 성능을 참고하십시오.

참고 항목

- NuGet에서 설치
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- TsgcVCLGrid
- 변경 시 깜박임
- 조건부 서식
- .NET 데모

WPF 빠른 시작

같은 시세 그리드를 WPF로 만듭니다. 그리드와 표시 옵션은 XAML에서 선언하고, 열과 행은 코드 비하인드에서 설정합니다.

단계

- `net8.0-windows` WPF 프로젝트를 만들고 **esegece.sgcGrid.Wpf** 패키지를 추가합니다. NuGet에서 설치를 참고하십시오.
- 창에서 `xmlns:grid="clr-namespace:esegece.sgcGrid;assembly=esegece.sgcGrid.Wpf"`로 네임스페이스를 매핑합니다.
- **TsgcWpfGrid**를 선언하고 **RowHeight**, **FlashChanges**, **TailMode** 등 다른 표시 옵션을 특성으로 설정합니다.
- 열, 서식 규칙, 행은 창의 생성자에서 `InitializeComponent` 뒤에 추가합니다.

예제

MainWindow.xaml:

```
<Window x:Class="GridQuickStartWpf.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:grid="clr-namespace:esegece.sgcGrid;assembly=esegece.sgcGrid.Wpf"
        Title="sgcGrid Quick Start" Width="720" Height="400">
  <DockPanel>
    <CheckBox x:Name="chkDark" DockPanel.Dock="Top" Margin="8" Content="Dark theme"
              Checked="OnThemeChanged" Unchecked="OnThemeChanged" />
    <grid:TsgcWpfGrid x:Name="Grid" RowHeight="26" FlashChanges="True" TailMode="False" />
  </DockPanel>
</Window>
```

MainWindow.xaml.cs:

```
using System;
using System.Windows;
using System.Windows.Threading;
using esegece.sgcGrid;
using esegece.sgcUICore;

namespace GridQuickStartWpf
{
    public partial class MainWindow : Window
    {
        private static readonly string[] Symbols = { "AAPL", "MSFT", "NVDA", "AMZN", "GOOG", "META" };
        private readonly DispatcherTimer tmrFeed = new DispatcherTimer();
        private readonly Random FRandom = new Random();

        public MainWindow()
        {
            InitializeComponent();
            // the columns are a model collection, filled from code
            Grid.Columns.Add("symbol", "Symbol");
            Grid.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat).DisplayFormat = "#,##0.00";
            Grid.Columns.Add("change", "Change", TsgcGridDataType.gdtFloat).DisplayFormat = "+0.00;-0.00;0.00";
            Grid.FormatRules.Add("change", TsgcGridFormatOperator.gfoLess, "0", 0x30E53935).ApplyToRow = true;
            foreach (string vSymbol in Symbols)
                Grid.Data.AppendRow(vSymbol, vSymbol, 100.0, 0.0);

            tmrFeed.Interval = TimeSpan.FromMilliseconds(250);
            tmrFeed.Tick += (s, e) => UpdateQuotes();
            tmrFeed.Start();
        }
```

```
        private void UpdateQuotes()
        {
            string vSymbol = Symbols[FRandom.Next(Symbols.Length)];
            double vChange = Math.Round(FRandom.NextDouble() * 4 - 2, 2);
            Grid.Data.UpdateRow(vSymbol, new[] { "price", "change" }, new object[] { 100.0 + vChange, vChange });
        }

        private void OnThemeChanged(object sender, RoutedEventArgs e)
        {
            Grid.Theme = chkDark.IsChecked == true ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        }
    }
}
```

App.xaml은 새 WPF 프로젝트의 것을 그대로 사용하며 `StartupUri="MainWindow.xaml"`이 지정되어 있습니다. WPF 그리드는 FireMonkey 그리드와 마찬가지로 자체 스크롤 막대를 그립니다. Delphi와 .NET의 차이점을 참고하십시오.

참고 항목

- NuGet에서 설치
- WinForms 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- TsgcFMXGrid
- .NET 데모

Avalonia 빠른 시작

같은 시세 그리드를 Windows, Linux, macOS의 Avalonia로 만듭니다.

단계

- **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 11.3을 사용하는 `net8.0` Avalonia 데스크톱 프로젝트를 만들고 **esegece.sgcGrid.Avalonia** 패키지를 추가합니다. NuGet에서 설치를 참고하십시오.
- 창에서 `xmlns:grid="using:esegece.sgcGrid"`로 네임스페이스를 매핑합니다.
- **TsgcAvaloniaGrid**를 `Name`과 함께 선언하고, `AvaloniaXamlLoader.Load` 뒤에 찾아서 열과 행을 추가합니다.
- 코드 비하인드에서 **Theme**을 전환합니다.

예제

MainWindow.axaml:

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:grid="using:esegece.sgcGrid"
        x:Class="GridQuickStartAvalonia.MainWindow"
        Title="sgcGrid Quick Start" Width="720" Height="400">
  <DockPanel>
    <CheckBox Name="chkDark" DockPanel.Dock="Top" Margin="8" Content="Dark theme"
              IsCheckedChanged="OnThemeChanged" />
    <grid:TsgcAvaloniaGrid Name="Grid" RowHeight="26" FlashChanges="True" />
  </DockPanel>
</Window>
```

MainWindow.axaml.cs:

```
using System;
using Avalonia.Controls;
using Avalonia.Interactivity;
using Avalonia.Markup.Xaml;
using Avalonia.Threading;
using esegece.sgcGrid;
using esegece.sgcUICore;

namespace GridQuickStartAvalonia
{
    public partial class MainWindow : Window
    {
        private static readonly string[] Symbols = { "AAPL", "MSFT", "NVDA", "AMZN", "GOOG", "META" };
        private readonly DispatcherTimer tmrFeed = new DispatcherTimer();
        private readonly Random FRandom = new Random();
        private readonly TsgcAvaloniaGrid FGrid;

        public MainWindow()
        {
            AvaloniaXamlLoader.Load(this);
            FGrid = this.FindControl<TsgcAvaloniaGrid>("Grid");
            FGrid.Columns.Add("symbol", "Symbol");
            FGrid.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat).DisplayFormat = "#,##0.00";
            FGrid.Columns.Add("change", "Change", TsgcGridDataType.gdtFloat).DisplayFormat = "+0.00;-0.00;0.00";
            FGrid.FormatRules.Add("change", TsgcGridFormatOperator.gfoLess, "0", 0x30E53935).ApplyToRow = true;
            foreach (string vSymbol in Symbols)
                FGrid.Data.AppendRow(vSymbol, vSymbol, 100.0, 0.0);

            tmrFeed.Interval = TimeSpan.FromMilliseconds(250);
            tmrFeed.Tick += (s, e) => UpdateQuotes();
            tmrFeed.Start();
```

```
        }

        private void UpdateQuotes()
        {
            string vSymbol = Symbols[FRandom.Next(Symbols.Length)];
            double vChange = Math.Round(FRandom.NextDouble() * 4 - 2, 2);
            FGrid.Data.UpdateRow(vSymbol, new[] { "price", "change" }, new object[] { 100.0 + vChange, vChange });
        }

        private void OnThemeChanged(object sender, RoutedEventArgs e)
        {
            // Theme is declared new on the grid, so it is the sgc theme,
            // not the Avalonia ControlTheme of StyledElement
            CheckBox oDark = (CheckBox)sender;
            if (FGrid != null)
                FGrid.Theme = oDark.IsChecked == true ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        }
    }
}
```

Program.cs:

```
using System;
using Avalonia;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;

namespace GridQuickStartAvalonia
{
    internal class App : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new MainWindow();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main(string[] args)
        {
            BuildAvaloniaApp().StartWithClassicDesktopLifetime(args);
        }

        public static AppBuilder BuildAvaloniaApp()
        {
            return AppBuilder.Configure<App>().UsePlatformDetect().LogToTrace();
        }
    }
}
```

터치 화면에서는 내용이 손가락을 따라 움직이고 탭하면 셀이 선택됩니다. `Control` 형식의 참조를 통하면 **Theme**은 Avalonia 컨트롤 테마입니다. Delphi와 .NET의 차이점을 참고하십시오.

참고 항목

- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- TsgcFMXGrid
- .NET 데모

.NET 클래스

모든 클래스는 **esegece.sgcGrid** 네임스페이스에 있습니다. 세 가지 그리드 컨트롤은 프레임워크에 독립적인 모델과 렌더러 위에 놓인 얇은 호스트이며, 모델과 렌더러는 호스트, 헤드리스 출력, 브리지가 공유합니다.

호스트 클래스

호스트	클래스	기본 클래스	Delphi
WinForms	TsgcWinFormsGrid	TsgcUIWinFormsControl	TsgcVCLGrid
WPF	TsgcWpfGrid	TsgcUIWpfControl	TsgcFMXGrid
Avalonia	TsgcAvaloniaGrid	TsgcUIAvaloniaControl	TsgcFMXGrid

세 그리드는 **IsgcUITable**을 구현하므로 sgcCharts 차트가 **TsgcChartDataSource_Table**을 통해 그리드를 읽을 수 있습니다.

프레임워크 독립 형식

형식	설명
TsgcGridColumns, TsgcGridColumn	열: **FieldName**, **Caption**, **DataType**, **Width**, **MinWidth**, **Alignment**, **DisplayFormat**, **Visible**, **Editable**, **SortOrder**, **SortIndex**, **PickList**, 그리고 그룹화와 요약을 위한 **GroupIndex**, **SummaryType**, **SummaryFormat**. TsgcGridColumn을 참고하십시오.
TsgcGridData	스레드 안전한 키 기반 행 저장소: **AppendRow**, **UpdateRow**, **UpdateCell**, **DeleteRow**, **Clear**, **BeginUpdate**와 **EndUpdate**, 변경 스탬프, **MaxRows** 꼬리 모드, 정렬, **FilterText**와 **OnFilterRow**, 가상 모드(**Virtual**, **VirtualRowCount**, **OnGetValue**), **FormatSettings**. **LayoutVersion**은 그룹화나 트리를 바꿀 수 있는 모든 변경 때마다 증가하며, **SetLayoutFields**는 값이 바뀌어도 증가하게 하는 필드를 추가합니다. TsgcGridData를 참고하십시오.
TsgcGridDataType	`gdtText` , `gdtInteger` , `gdtFloat` , `gdtCurrency` , `gdtDateTime` , `gdtBoolean` .
TsgcGridFormatRules, TsgcGridFormatRule	조건부 서식: **FieldName**, **Operator**(`gfoEqual` 부터 `gfoIsEmpty` 까지), **Value**, **Value2**, **BackColor**, **ForeColor**, **Bold**, **ApplyToRow**, **Enabled**, **StopOnMatch**. TsgcGridFormatRule을 참고하십시오.
TsgcGridSummaryType	`gsuNone` , `gsuSum` , `gsuCount` , `gsuAverage` , `gsuMin` , `gsuMax` .
TsgcGridLineView, TsgcGridGroupInfo	데이터의 표시 라인(그룹 행, 데이터 행, 트리 행, 세부 정보 밴드)이며, **GroupBy**, **GroupCount**, **GroupInfo**, **GroupSummary**와 펼침 상태를 제공합니다. .NET 그룹화와 요약을 참고하십시오.
TsgcGridTreeOptions	트리 행: **KeyField**, **ParentField**, **TreeField**, **Indent**, **Expanded**. .NET 트리 행과 마스터 디테일을 참고하십시오. Registered 에디션 전용입니다.
TsgcGridDetailOptions	세부 정보 밴드: **Active**, **Source**, **MasterKeyField**, **DetailKeyField**, **Lines**, **Columns**. .NET 트리 행과 마스터 디테일을 참고하십시오. Registered 에디션 전용입니다.

TsgcGridDataStore	세부 행을 보관하는 비시각적 홀더이며, 코드 또는 **TsgcGridSource_DataSet**으로 채웁니다. Registered 에디션 전용입니다.
TsgcGridXLSXWriter	데이터와 그 라인 뷰를 Excel 통합 문서로 기록합니다. .NET Excel 내보내기를 참고하십시오. Registered 에디션 전용입니다.
TsgcGridRenderer, TsgcGridRenderOptions	**TsgcGridData**의 프레임을 **TsgcUICanvas**에 그리고, SVG로, 또는 esegece.sgcGrid.Rendering을 사용하면 PNG로 기록합니다. .NET 헤드리스 렌더링을 참고하십시오.
sgcGrid_IO	JSON 및 CSV 가져오기와 내보내기. .NET DataTable 바인딩, JSON 및 CSV를 참고하십시오.
TsgcGridSource_DataSet	`System.Data.DataTable` 을 그리드에 바인딩합니다. .NET DataTable 바인딩, JSON 및 CSV를 참고하십시오.
TsgcGridFeedSource와 그 하위 클래스	esegece.sgcGrid.sgcWebSockets의 실시간 피드 소스. .NET 실시간 피드를 참고하십시오. Registered 에디션 전용입니다.
TsgcGridAIQuery	esegece.sgcGrid.sgcHTML의 자연어 질문. .NET 그리드에 질문하기를 참고하십시오. Registered 에디션 전용입니다.

Grid

TsgcWinFormsGrid, **TsgcWpfGrid**, **TsgcAvaloniaGrid**에서 공통인 멤버입니다.

멤버	형식	설명
Columns	TsgcGridColumns	열.
Data	TsgcGridData	행 저장소. 읽기 전용입니다. 어떤 스레드에서든 쓸 수 있습니다.
FormatRules	TsgcGridFormatRules	조건부 서식 규칙.
Renderer	TsgcGridRenderer	그리드의 렌더러. 읽기 전용입니다.
HeaderHeight, RowHeight	int	헤더와 행의 높이. 기본값은 각각 32와 28입니다.
GridLinesH, GridLinesV, AlternateRows	bool	가로 및 세로 그리드 선과 번갈아 나오는 행의 색조. 기본값은 true입니다.
SelectionMode	TsgcGridSelectionMode	`gsmRow` (기본값) 또는 `gsmCell` .
Theme, AccentColor	TsgcUITheme, uint	그리드의 테마와 선택 사항인 `$AARRGGBB` 강조 색. 0은 테마의 강조 색을 사용합니다.
FontSize	double	셀과 헤더의 글꼴 크기. 기본값은 13입니다.
MaxRows, TailMode	int, bool	테일 모드: **MaxRows**를 넘으면 가장 오래된 행이 삭제되고, **TailMode**는 마지막 행이 계속 보이게 합니다.
FilterText	string	서식이 적용된 텍스트에 해당 문자열이 포함된 행만 표시합니다.
FlashChanges, FlashDuration, FlashMode	bool, int, TsgcGridFlashMode	변경 깜박임. 기본값은 800 ms이며 배경 또는 텍스트에 적용됩니다.

FlashUpColor, FlashDownColor, FlashChangedColor	uint	깜박임 색. 0은 테마 색을 사용합니다.
ReadOnly	bool	기본값은 True입니다. False로 설정하면 **Editable** 열의 제자리 편집이 활성화됩니다. Registered 에디션 전용입니다.
SelectedRow, SelectedCol, SelectedKey	int, int, string	선택. 행을 다시 정렬해도 선택은 키를 통해 해당 행을 따라갑니다.
TopRow, ScrollX, ScrollToRow	int, int, method	스크롤 위치. **TopRow**는 표시 라인을 세며, 아무것도 그룹화되지 않았을 때는 표시 행과 같습니다.
RowAt, ColAt	method	한 지점 아래의 표시 행과 열.
StartEdit, CommitEdit, CancelEdit, EditorActive	method, bool	코드에서의 제자리 편집. Registered 에디션 전용입니다.
SaveToSVG, SaveToPNG	method	현재 보기를 이미지로 기록합니다.
OnCellClick, OnCellDblClick	TsgcGridCellEvent	셀을 클릭했습니다. 표시 행과 열 인덱스가 함께 전달됩니다.
OnSelectionChange, OnSortChange	EventHandler	선택 또는 정렬이 변경되었습니다.
OnColumnResized	TsgcGridColumnEvent	사용자가 열 크기를 조정했습니다.
OnGetCellStyle	TsgcGridCellStyleEvent	셀이 그려지기 전에 해당 셀의 색을 변경합니다.
OnBeforeEdit, OnValidate, OnEdited, OnEditError	이벤트	제자리 편집. .NET 제자리 편집을 참고하십시오. Registered 에디션 전용입니다.
ShowGroupPanel, GroupPanelHeight, GroupIndent, GroupsExpanded, GroupSummaries	bool, int, int, bool, bool	그룹화 표시 옵션입니다. .NET 그룹화와 요약을 참고하십시오.
ShowFooter, FrozenColumns	bool, int	요약 바닥글과, 다른 열이 스크롤되어도 계속 표시되는 열입니다.
Lines, FocusedLine	TsgcGridLineView, int	표시 라인과 포커스가 있는 라인입니다.
GroupBy, ClearGrouping, ExpandAll, CollapseAll, ExpandGroup, CollapseGroup, GroupCount, GroupInfo, FooterSummary, FooterSummaryText	method	코드에서의 그룹화와 요약입니다.
OnGroupingChange, OnGroupExpandChange, OnGetGroupText	이벤트	그룹화나 그룹의 상태가 변경되었을 때와 그룹 행의 캡션입니다.
Tree, Detail	TsgcGridTreeOptions, TsgcGridDetailOptions	트리 행과 세부 정보 밴드입니다. .NET 트리 행과 마스터 디테일을 참고하십시오. Registered 에디션 전용입니다.

ExpandNode, CollapseNode, ExpandDetail, CollapseDetail, DetailExpanded, DetailData	method	코드에서의 트리 노드와 세부 정보 밴드 조작입니다. Registered 에디션 전용입니다.
OnTreeExpandChange, OnDetailExpandChange, OnDetailCellClick, OnGetDetailData	이벤트	트리와 세부 정보 이벤트입니다. Registered 에디션 전용입니다.
SaveToXLSX	method	뷰를 Excel 통합 문서로 기록합니다. .NET Excel 내보내기를 참고하십시오. Registered 에디션 전용입니다.
VertBarVisible, HorzBarVisible, BorderStyle	bool, bool, TsgcGridBorderStyle	WPF와 Avalonia: 그려지는 스크롤 막대와 테두리. WinForms는 `System.Windows.Forms` 의 **BorderStyle** 과 네이티브 스크롤 막대를 사용합니다.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET DataTable 바인딩, JSON 및 CSV
- .NET 실시간 피드
- .NET 그리드에 질문하기
- .NET 그룹화와 요약
- .NET 트리 행과 마스터 디테일
- .NET Excel 내보내기
- TsgcGridData
- TsgcGridRenderer

Delphi와 .NET의 차이점

.NET 포트는 Delphi의 이름, 기본값, 렌더링 규칙을 유지하며, SVG 출력은 Delphi의 출력과 바이트 단위로 같습니다. 아래 차이점은 호스트 프레임워크와 .NET 클래스 라이브러리에서 비롯됩니다.

IsgcGridDataHost 대신 Data와 FormatRules

Delphi에서는 소스가 IsgcGridDataHost 및 IsgcUITable 인터페이스를 통해 그리드의 데이터를 찾습니다. .NET 포트에는 이 인터페이스가 없습니다. **TsgcWinFormsGrid**, **TsgcWpfGrid**, **TsgcAvaloniaGrid**는 **Data**와 **FormatRules**를 public 속성으로 노출하고, **TsgcGridSource_DataSet**, 피드 소스, **TsgcGridAIQuery**는 리플렉션으로 이를 찾습니다. **Grid**에는 **TsgcGridData** 형식의 public **Data** 속성을 가진 모든 객체를 지정할 수 있고, AI 쿼리는 강조 표시를 위해 **TsgcGridFormatRules** 형식의 public **FormatRules** 속성도 읽습니다. 따라서 직접 만든 클래스를 소스의 **Grid**로 쓸 수 있습니다. .NET 그리드에 질문하기의 예제를 참고하십시오. **GridData**가 할당되어 있으면 항상 **Grid**보다 우선합니다.
.NET에는 FreeNotification이 없습니다. **Grid**가 WinForms 컨트롤 같은 `IComponent`이면 소스가 해당 `Disposed` 이벤트를 수신하고 그때 **Grid**를 지웁니다. 그 밖의 객체에서는 그리드가 사라지기 전에 **Grid**를 null로 설정하십시오. **GridData**는 감시되지 않습니다.

스크롤 막대

TsgcWinFormsGrid는 VCL 그리드와 마찬가지로 Windows 네이티브 스크롤 막대를 사용합니다. **TsgcWpfGrid**와 **TsgcAvaloniaGrid**는 FireMonkey 그리드와 마찬가지로 자체 스크롤 막대를 그리며, 썸 끌기와 트랙 페이지 이동을 지원합니다. **VertBarVisible**과 **HorzBarVisible**로 표시하거나 숨깁니다.

WPF의 FontSize, Avalonia의 Theme

그리드는 텍스트를 직접 그리므로 **TsgcWpfGrid**는 `new`로 자체 **FontSize**를 선언하여 `Control.FontSize`를 가립니다. `Control` 형식의 참조를 통하면 이것은 WPF 속성이며, 그리드는 이를 읽지 않습니다. 마찬가지로 모든 Avalonia 컨트롤에는 이미 **Theme** 속성, 즉 `StyledElement`의 `ControlTheme`가 있습니다. **TsgcAvaloniaGrid**는 `new`로 **TsgcUITheme** 형식의 자체 **Theme**을 선언합니다. 그리드 형식의 참조를 통하면 sgc 테마이고, `Control` 또는 `StyledElement` 형식의 참조를 통하면 Avalonia 컨트롤 테마입니다.
WPF와 Avalonia에서 두 속성은 선택 및 기타 표시 옵션과 마찬가지로 바인딩과 스타일을 받습니다. .NET 데이터 바인딩을 참조하십시오.

터치

Avalonia에서는 터치 입력으로 그리드를 이동하며, FireMonkey 이동 제스처처럼 관성 스크롤 없이 콘텐츠가 손가락을 따라가고 탭하면 셀이 선택됩니다. 거기서는 터치로 헤더를 그룹 패널로 끌 수 없으며, 펼침 화살표, 확장기, 세부 정보 밴드를 탭하면 클릭처럼 동작합니다. WPF에는 터치 이동이 없습니다. 터치와 펜은 마우스 입력으로 승격되므로 터치로 선택하고 그려진 스크롤 막대로 스크롤합니다. WinForms는 마우스와 휠을 사용합니다.

데이터 형식

CLR 형식은 Delphi 필드 형식과 같은 방식으로 그리드 데이터 형식에 대응합니다. `byte`, `short`, `int`, `long` 등의 정수 형식은 `gdtInteger`에, `float`, `double`, `decimal`은 `gdtFloat`에 대응하며(Delphi의 `ftBCD`와 `ftFMTBcd` 필드와 같습니다), `DateTime`, `DateTimeOffset`, `TimeSpan`은 `gdtDateTime`에, `bool`은 `gdtBoolean`에 대응하고, 그 밖의 모든 형식은 `gdtText`입니다. CLR에는 통화 형식이 없으므로 `gdtCurrency`는 직접 추가한 열에서만 나옵니다.

TDataSet 대신 DataTable

TsgcGridSource_DataSet은 Delphi `TDataSet`에 해당하는 .NET 클래스인 `System.Data.DataTable`을 바인딩합니다. DataTable은 항상 열려 있고 커서가 없으므로, 현재 레코드도 없고 로드 후에 복원할 북마크도 없습니다. Delphi 데이터 링크 대신 **LiveUpdates** 이벤트(`RowChanged`, `RowDeleted`, `TableCleared`)가 쓰이며, 완료되지 않은 편집과 `AcceptChanges`는 무시됩니다. .NET DataTable 바인딩, JSON 및 CSV를 참고하십시오.

항목 ID가 없는 서식 규칙

.NET 서식 규칙에는 컬렉션 항목 ID가 없습니다. **TsgcGridAIQuery.Reset**은 추가한 규칙을 ID가 아니라 참조로 제거하며, 열은 여전히 **ID**로 찾습니다.

트리 행, 세부 정보 밴드, Excel 내보내기의 에디션

Delphi에서는 트리 행, 세부 정보 밴드, **SaveToXLSX**가 `SGC_GRID_ADVANCED` 정의로 컴파일됩니다. .NET Community 패키지에는 **TsgcGridTreeOptions**, **TsgcGridDetailOptions**, **TsgcGridDataStore**, **TsgcGridXLSXWriter**, 그리고 이를 사용하는 그리드 멤버가 들어 있지 않습니다. 그룹화, 요약, 바닥글, 고정 열은 두 에디션 모두에 있습니다. .NET 트리 행과 마스터 디테일과 .NET Excel 내보내기를 참고하십시오.

그룹화, 트리, 세부 정보 이벤트

새 그리드 이벤트는 그리드의 다른 이벤트와 마찬가지로 C# 이벤트입니다. 델리게이트인 **TsgcGridGroupExpandEvent**, **TsgcGridRowExpandEvent**, **TsgcGridDetailCellEvent**는 esegece.sgcGrid에 한 번만 선언되며 세 호스트가 공유합니다. **GroupCount**처럼 인수가 없는 메서드는 메서드로 남습니다.

세부 정보 소스

Detail.Source는 `object`이며, **TsgcGridDataStore**, **TsgcGridData**, 또는 **TsgcGridData** 형식의 public **Data** 속성을 가진 모든 개체가 될 수 있고, 소스가 그리드를 찾는 방식과 같은 방식으로 찾습니다. 해제 알림이 없으므로 `IComponent`인 Source는 `Disposed` 이벤트가 발생할 때 지워집니다.

Excel 파일

통합 문서는 `System.IO.Compression.DeflateStream`으로 압축됩니다. XML 부분은 같은 데이터에 대한 Delphi 출력과 바이트가 동일하며, 압축된 바이트, 즉 파일 크기만 다릅니다.

새 옵션의 디자이너 직렬화

DFM은 없습니다. WinForms에서는 7개의 그룹화 옵션이 기본값을 가지며 기본값과 다를 때만 직렬화되고, **Tree**와 **Detail**은 콘텐츠로 직렬화됩니다. WPF와 Avalonia에서는 이 옵션이 종속성 속성과 스타일 속성입니다.

디자이너 동사 없음

Delphi 그리드는 IDE에 디자인 타임 동사를 추가하지만 .NET 그리드에는 없습니다. 빠른 시작 예제처럼 **Columns**와 **FormatRules**는 코드에서 추가하십시오.

참고 항목

- .NET 개요
- .NET 클래스
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET DataTable 바인딩, JSON 및 CSV
- .NET 그리드에 질문하기
- .NET 트리 행과 마스터 디테일
- .NET Excel 내보내기
- IsgcGridDataHost and IsgcUITable

.NET 데이터 바인딩

이 항목은 **TsgcWpfGrid**와 **TsgcAvaloniaGrid**만 다룹니다. 이들의 사용자용 스칼라 속성은 WPF 종속성 속성과 Avalonia 스타일 속성이므로 바인딩, 스타일, XAML setter를 받을 수 있으며, 뷰 모델이 코드 비하인드 없이 그리드를 구동할 수 있습니다. **TsgcWinFormsGrid**는 변경되지 않았으며, 해당 속성은 이전과 같은 일반 CLR 속성입니다.

바인딩 가능한 속성

아래 속성은 두 그리드 모두에서 바인딩을 받습니다. TwoWay로 표시된 속성은 기본적으로 TwoWay로 바인딩되므로 `Mode`가 없는 바인딩도 그리드의 변경을 뷰 모델로 되돌려 보냅니다. 나머지는 모두 기본적으로 OneWay로 바인딩됩니다.

속성	형식	기본 모드
SelectedRow	int	TwoWay
SelectedCol	int	TwoWay
Theme	TsgcUITheme	OneWay
FontSize	double	OneWay
BorderStyle	TsgcGridBorderStyle	OneWay
ReadOnly	bool	OneWay
TailMode	bool	OneWay
MaxRows	int	OneWay
FilterText	string	OneWay
HeaderHeight	int	OneWay
RowHeight	int	OneWay
GridLinesH	bool	OneWay
GridLinesV	bool	OneWay
AlternateRows	bool	OneWay
SelectionMode	TsgcGridSelectionMode	OneWay
AccentColor	uint	OneWay
FlashChanges	bool	OneWay
FlashDuration	int	OneWay
FlashUpColor	uint	OneWay
FlashDownColor	uint	OneWay
FlashChangedColor	uint	OneWay
FlashMode	TsgcGridFlashMode	OneWay

SelectedRow와 **SelectedCol**은 클릭, 키 입력, 코드를 따라갑니다. Avalonia에서는 직접 속성입니다. WPF에서 **FontSize**는 `new`로 선언되어 `Control.FontSize`를 가리며, Avalonia에서는 **Theme**이 컨트롤 테마를 가립니다. Delphi와 .NET의 차이점을 참조하십시오. 예제처럼 그리드 요소 자체에 바인딩하십시오.

동작

- 그리드 모델은 자체 유효성 검사를 유지합니다. 예를 들어 음수 **FlashDuration**이나 마지막 행을 넘는 **SelectedRow**처럼 모델이 값을 거부하거나 수정하면, 그리드는 유지한 값을 속성에 다시 기록합니다. WPF에서 값이 TwoWay 소스에서 온 경우, 유지된 값은 바인딩이 전송을 마친 뒤 디스패처 틱 한 번 뒤에 그 소스로도 돌아갑니다.
- 코드가 **Renderer.Options**를 통해 모델을 직접 변경하면 새 값은 UI 스레드에서만 바인딩 가능한 속성으로 복사되므로, 바인딩된 뷰 모델이 다른 스레드에서 업데이트되는 일은 없습니다.
- **OnSelectionChange**와 기타 이벤트는 바인딩, 스타일, 코드, 마우스, 키보드 중 어디에서 온 변경이든 발생합니다.
- **FilterText**가 행을 옮기면 선택은 행 키를 따라가고, 바인딩된 **SelectedRow**도 따라갑니다.

CLR로 남는 속성

- **Data**, **Columns**, **FormatRules**는 **TsgcGridSource_DataSet**, 피드 소스, **TsgcGridAIQuery**가 리플렉션으로 찾기 때문에 CLR 속성으로 남습니다. 코드에서 또는 소스를 통해 채우십시오. .NET DataTable 바인딩, JSON 및 CSV와 .NET 실시간 피드를 참조하십시오.
- **TopRow**와 **ScrollX**는 스크롤 단계마다 바뀌는 뷰 상태이므로 CLR 속성으로 남습니다.
- **SelectedKey**, **VertBarVisible**, **HorzBarVisible**, **Renderer** 같은 읽기 전용 상태 속성은 바인딩할 대상이 없으므로 CLR 속성으로 남습니다.

뷰 모델

두 예제는 같은 뷰 모델을 사용합니다. **SelectedRow**는 TwoWay로 바인딩되고, **FilterText**는 텍스트 상자와 공유되며, **Theme**은 확인란에서 계산됩니다.
MainViewModel.cs:

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using esegece.sgcUICore;

namespace GridBinding
{
    public class MainViewModel : INotifyPropertyChanged
    {
        private int FSelectedRow = -1;
        private string FFilterText = "";
        private bool FDark;

        public event PropertyChangedEventHandler PropertyChanged;

        // bound to SelectedRow, TwoWay by default
        public int SelectedRow
        {
            get { return FSelectedRow; }
            set { DoSet(ref FSelectedRow, value); }
        }

        public string FilterText
        {
            get { return FFilterText; }
            set { DoSet(ref FFilterText, value); }
        }

        public bool Dark
        {
            get { return FDark; }
            set
            {
                if (DoSet(ref FDark, value))
```

```
                    DoChanged(nameof(Theme));
            }
        }

        // bound to Theme, OneWay
        public TsgcUITheme Theme
        {
            get { return FDark ? TsgcUITheme.utDark : TsgcUITheme.utLight; }
        }

        private bool DoSet<T>(ref T aField, T aValue, [CallerMemberName] string aName = null)
        {
            if (EqualityComparer<T>.Default.Equals(aField, aValue))
                return false;
            aField = aValue;
            DoChanged(aName);
            return true;
        }

        private void DoChanged(string aName)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(aName));
        }
    }
}
```

WPF 예제

MainWindow.xaml:

```
<Window x:Class="GridBinding.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:GridBinding"
        xmlns:grid="clr-namespace:esegece.sgcGrid;assembly=esegece.sgcGrid.Wpf"
        Title="sgcGrid binding" Width="720" Height="400">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <StackPanel DockPanel.Dock="Top" Orientation="Horizontal" Margin="8">
            <TextBox Width="160" Text="{Binding FilterText, UpdateSourceTrigger=PropertyChanged}" />
            <CheckBox Margin="16,0,0,0" Content="Dark theme" IsChecked="{Binding Dark}" />
            <TextBlock Margin="16,0,0,0" Text="{Binding SelectedRow}" />
        </StackPanel>
        <!-- SelectedRow binds TwoWay by default, so the view model follows a click, a key or code -->
        <grid:TsgcWpfGrid x:Name="Quotes" SelectedRow="{Binding SelectedRow}"
                          FilterText="{Binding FilterText}" Theme="{Binding Theme}" RowHeight="26" />
    </DockPanel>
</Window>
```

MainWindow.xaml.cs:

```
using System.Windows;
using esegece.sgcGrid;

namespace GridBinding
{
    public partial class MainWindow : Window
    {
        private static readonly string[] Symbols = { "AAPL", "MSFT", "NVDA", "AMZN", "GOOG", "META" };

        public MainWindow()
        {
            InitializeComponent();
```

```
            // Columns and Data stay CLR properties, so they are filled from code
            Quotes.Columns.Add("symbol", "Symbol");
            Quotes.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat).DisplayFormat = "#,##0.00";
            foreach (string vSymbol in Symbols)
                Quotes.Data.AppendRow(vSymbol, vSymbol, 100.0);
        }
    }
}
```

Avalonia 예제

Avalonia 창은 컴파일된 바인딩을 사용하므로 `x:DataType`을 뷰 모델로 지정합니다.
MainWindow.axaml:

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:GridBinding"
        xmlns:grid="using:esegece.sgcGrid"
        x:Class="GridBinding.MainWindow"
        x:DataType="local:MainViewModel"
        Title="sgcGrid binding" Width="720" Height="400">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <StackPanel DockPanel.Dock="Top" Orientation="Horizontal" Margin="8" Spacing="16">
            <TextBox Width="160" Text="{Binding FilterText}" />
            <CheckBox Content="Dark theme" IsChecked="{Binding Dark}" />
            <TextBlock VerticalAlignment="Center" Text="{Binding SelectedRow}" />
        </StackPanel>
        <!-- SelectedRow is a direct property that binds TwoWay by default -->
        <grid:TsgcAvaloniaGrid x:Name="Quotes" SelectedRow="{Binding SelectedRow}"
                               FilterText="{Binding FilterText}" Theme="{Binding Theme}" RowHeight="26" />
    </DockPanel>
</Window>
```

MainWindow.axaml.cs:

```
using Avalonia.Controls;
using esegece.sgcGrid;

namespace GridBinding
{
    public partial class MainWindow : Window
    {
        private static readonly string[] Symbols = { "AAPL", "MSFT", "NVDA", "AMZN", "GOOG", "META" };

        public MainWindow()
        {
            InitializeComponent();
            // Columns and Data stay CLR properties, so they are filled from code
            Quotes.Columns.Add("symbol", "Symbol");
            Quotes.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat).DisplayFormat = "#,##0.00";
            foreach (string vSymbol in Symbols)
                Quotes.Data.AppendRow(vSymbol, vSymbol, 100.0);
        }
    }
}
```

`vm.SelectedRow = 3`을 설정하면 AMZN 행이 선택되고, 다른 행을 클릭하면 그 인덱스가 `vm.SelectedRow`에 다시 기록됩니다. 텍스트 상자에 입력하면 그리드가 필터링되고, 확인란은 테마를 전환합니다.

참고 항목

- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET DataTable 바인딩, JSON 및 CSV
- .NET 실시간 피드

.NET 헤드리스 렌더링

렌더러는 컨트롤도 창도 필요하지 않으므로 서비스, 웹 애플리케이션, Linux 서버에서도 그리드 이미지를 기록할 수 있습니다. **TsgcGridRenderer.SaveToSVG**는 코어 패키지에서 SVG를 기록합니다. esegece.sgcGrid.Rendering의 확장 메서드인 **SaveToPNG**는 SkiaSharp를 통해 PNG를 기록합니다.

예제

이 콘솔 프로그램은 모델을 만들고 정렬한 다음 서식 규칙을 추가하고, 같은 프레임을 SVG와 두 배 크기의 PNG로 기록합니다.

```
using System;
using System.IO;
using esegece.sgcGrid;
using esegece.sgcUICore;

namespace GridHeadless
{
    internal static class Program
    {
        private static void Main()
        {
            // the model: columns and keyed rows, no control and no window
            TsgcGridData oData = new TsgcGridData();
            oData.UseInvariantFormat();
            oData.Columns.Add("region", "Region");
            oData.Columns.Add("orders", "Orders", TsgcGridDataType.gdtInteger).DisplayFormat = "#,##0";
            oData.Columns.Add("revenue", "Revenue", TsgcGridDataType.gdtFloat).DisplayFormat = "#,##0.00";
            oData.Columns.Add("growth", "Growth %", TsgcGridDataType.gdtFloat).DisplayFormat = "+0.0;-0.0";
            oData.AppendRow("EU", "Europe", 1840, 245300.50, 6.2);
            oData.AppendRow("NA", "North America", 2310, 318950.00, 4.1);
            oData.AppendRow("APAC", "Asia Pacific", 1275, 162420.75, -1.8);
            oData.AppendRow("LATAM", "Latin America", 640, 71880.20, 10.5);

            // sort by revenue, highest first
            oData.Columns.FindByField("revenue").SortOrder = TsgcGridSortOrder.gsoDescending;

            // the renderer paints the same frame as the WinForms, WPF and Avalonia grids
            TsgcGridRenderer oRenderer = new TsgcGridRenderer();
            oRenderer.Options.Theme = TsgcUITheme.utDark;
            oRenderer.Options.FormatRules.Add("growth", TsgcGridFormatOperator.gfoLess, "0", 0x40E53935);
            TsgcGridRenderState vState = sgcGrid_Render.sgcGridRenderState();

            // SVG from the core package, PNG from esegece.sgcGrid.Rendering
            oRenderer.SaveToSVG(oData, vState, "regions.svg", 520, 170);
            oRenderer.SaveToPNG(oData, vState, "regions.png", 520, 170, 2);

            for (int i = 0; i < oData.ViewCount(); i++)
                Console.WriteLine(oData.CellText(i, 0) + ": " + oData.CellText(i, 2));
            Console.WriteLine("regions.svg " + new FileInfo("regions.svg").Length + " bytes, regions.png " +
                new FileInfo("regions.png").Length + " bytes");
        }
    }
}
```

프로그램의 출력:

```
North America: 318,950.00
Europe: 245,300.50
Asia Pacific: 162,420.75
```

```
Latin America: 71,880.20
regions.svg 6846 bytes, regions.png 49632 bytes
```

참고

- 너비와 높이는 장치 독립 픽셀 단위입니다. **SaveToSVG**와 **SaveToPNG**의 마지막 인수는 배율이므로, 배율 2에서 `520 x 170`은 1040 x 340 픽셀 이미지를 기록합니다.
- `sgcGrid_Render.sgcGridRenderState()`는 선택과 스크롤이 없는 상태를 반환합니다. 스크롤되었거나 선택된 프레임을 렌더링하려면 해당 필드를 설정하십시오.
- 렌더링 옵션(**Theme**, **FontSize**, **RowHeight**, 깜박임 설정, **FormatRules**)은 그리드 컨트롤이 노출하는 것과 같으며, 그리드 컨트롤은 **SaveToSVG**와 **SaveToPNG**로 자신의 보기를 기록합니다.
- SVG 텍스트는 같은 데이터에 대해 Delphi 렌더러가 기록하는 것과 바이트 단위로 같습니다. PNG 출력은 Skia 안티앨리어싱을 사용하므로 Delphi 래스터와 개별 픽셀 단위로 다를 수 있습니다.
- **UseInvariantFormat**은 **CellText**가 컴퓨터의 문화권에 의존하지 않게 합니다.

참고 항목

- .NET 개요
- .NET 클래스
- .NET 성능
- 헤드리스 렌더링
- TsgcGridRenderer

.NET DataTable 바인딩, JSON 및 CSV

TsgcGridSource_DataSet은 `System.Data.DataTable`을 그리드에 바인딩하고, **sgcGrid_IO** 클래스는 행을 JSON과 CSV로 로드하고 저장합니다. 둘 다 코어 패키지에 있으며 모든 호스트와 헤드리스에서 동작합니다.

DataTable 바인딩

- **Grid**를 그리드 컨트롤로, 또는 **GridData**를 그 **Data**로 설정합니다.
- **AutoColumns**가 true(기본값)이고 그리드에 열이 없으면 DataTable에서 열이 만들어집니다. 필드 이름은 `ColumnName`, 캡션은 `Caption`이 되고, `ColumnMapping`이 `Hidden`인 열은 숨겨지며, 데이터 형식은 CLR 형식을 따릅니다 (`decimal`은 `gdtFloat`에 대응합니다).
- **KeyField**는 값이 행 키인 열을 지정합니다. 키가 있으면 다시 로드할 때 행이 제자리에서 갱신되고 변경 깜박임이 유지되며, 새 키는 추가되고 사라진 키는 삭제됩니다. 키가 없으면 로드할 때마다 행이 교체됩니다.
- **LiveUpdates**가 true(기본값)이면 DataTable의 커밋된 변경, 삭제, 지우기가 즉시 그리드에 반영됩니다. **Load**는 언제든 로드하고 **Refresh**는 다시 로드합니다.

```
using System;
using System.Data;
using esegece.sgcGrid;

namespace GridDataTableBinding
{
    internal static class Program
    {
        private static void Main()
        {
            DataTable oTable = new DataTable("Products");
            oTable.Columns.Add("sku", typeof(string));
            oTable.Columns.Add("name", typeof(string)).Caption = "Product";
            oTable.Columns.Add("stock", typeof(int)).Caption = "Stock";
            oTable.Columns.Add("price", typeof(decimal)).Caption = "Price";
            oTable.Rows.Add("A-100", "Keyboard", 42, 49.90m);
            oTable.Rows.Add("A-200", "Mouse", 120, 19.50m);
            oTable.Rows.Add("B-300", "Monitor", 8, 229.00m);

            // in an application GridData is Grid.Data, or set Grid to the grid control
            TsgcGridData oData = new TsgcGridData();
            oData.UseInvariantFormat();
            TsgcGridSource_DataSet oSource = new TsgcGridSource_DataSet();
            oSource.GridData = oData;
            oSource.KeyField = "sku";
            oSource.DataSet = oTable;
            oSource.Active = true;

            // AutoColumns created the columns: decimal maps to gdtFloat
            foreach (string vField in new[] { "sku", "name", "stock", "price" })
            {
                TsgcGridColumn oColumn = oData.Columns.FindByField(vField);
                Console.WriteLine(oColumn.FieldName + " \"" + oColumn.Caption + "\" " + oColumn.DataType);
            }

            // LiveUpdates: a committed change of the DataTable reaches the grid at once
            oTable.Rows[1]["stock"] = 95;
            oTable.Rows.Add("B-400", "Webcam", 30, 59.00m);
            oTable.Rows[0].Delete();

            Console.WriteLine(oData.RowCount() + " rows");
            for (int i = 0; i < oData.ViewCount(); i++)
                Console.WriteLine(oData.KeyOf(i) + ": " + oData.CellText(i, 1) + ", stock " + oData.CellText(i, 2));
            oSource.Dispose();
        }
```

```
    }
}
```

프로그램의 출력:

```
sku "sku" gdtText
name "Product" gdtText
stock "Stock" gdtInteger
price "Price" gdtFloat
3 rows
A-200: Mouse, stock 95
B-300: Monitor, stock 8
B-400: Webcam, stock 30
```

DataSet에 null을 할당하면 테이블 바인딩이 해제되고 행은 유지됩니다. DataTable을 해제(Dispose)해도 마찬가지입니다.

JSON 및 CSV

GridLoadFromJSON은 객체 배열을 읽습니다. `weather.temp`처럼 점이 있는 필드 이름은 중첩된 객체를 읽으며, **GridSaveToJSON**은 그것을 다시 같은 형태로 기록합니다. **GridLoadFromCSV**와 **GridSaveToCSV**는 RFC 4180 따옴표 규칙, 임의의 구분자, 헤더 행을 사용합니다. 키 필드 인수는 행 키를 지정하고, `aAutoColumns`를 지정하면 첫 번째 객체 또는 헤더 행에서 열이 만들어지며 형식은 값에서 추측합니다. 문자열, 스트림, 파일 오버로드는 UTF-8을 읽고 BOM을 건너뛰며, BOM 없이 UTF-8로 기록합니다.

```
using System;
using esegece.sgcGrid;

namespace GridJsonCsv
{
    internal static class Program
    {
        private static void Main()
        {
            // dotted paths are read from nested objects, and the key field names the row key
            string vJSON = "[{\"id\":1,\"city\":\"Barcelona\",\"weather\":{\"temp\":21.5,\"sky\":\"Sunny\"}}," +
                "{\"id\":2,\"city\":\"Oslo\",\"weather\":{\"temp\":8.0,\"sky\":\"Rain, light\"}}]";
            TsgcGridData oData = new TsgcGridData();
            oData.Columns.Add("id", "Id", TsgcGridDataType.gdtInteger);
            oData.Columns.Add("city", "City");
            oData.Columns.Add("weather.temp", "Temp", TsgcGridDataType.gdtFloat);
            oData.Columns.Add("weather.sky", "Sky");
            sgcGrid_IO.GridLoadFromJSON(oData, vJSON, "id", false);
            Console.WriteLine(oData.RowCount() + " rows from JSON, key of row 0: " + oData.KeyOf(0));

            // CSV with a header row and RFC 4180 quoting, invariant number format
            string vCSV = sgcGrid_IO.GridSaveToCSV(oData);
            Console.Write(vCSV);

            // and back: the columns are created from the header, the types guessed from the values
            TsgcGridData oCopy = new TsgcGridData();
            sgcGrid_IO.GridLoadFromCSV(oCopy, vCSV, "id");
            Console.WriteLine(oCopy.Columns.Count + " columns, " + oCopy.RowCount() + " rows from CSV, Temp is " +
                oCopy.Columns.FindByField("weather.temp").DataType);

            // JSON again, the dotted field names written as nested objects
            Console.WriteLine(sgcGrid_IO.GridSaveToJSON(oCopy));
        }
    }
}
```

프로그램의 출력:

```
2 rows from JSON, key of row 0: 1
id,city,weather.temp,weather.sky
1,Barcelona,21.5,Sunny
2,Oslo,8,"Rain, light"
```

```
4 columns, 2 rows from CSV, Temp is gdtFloat
[{"id":1,"city":"Barcelona","weather":{"temp":21.5,"sky":"Sunny"}},{"id":2,"city":"Oslo","weather":{"temp":8,"sky":"Rain, l
```

CSV 숫자는 `aInvariant`가 false가 아닌 한 고정 문화권을 사용하며, false이면 데이터의 **FormatSettings**가 사용됩니다. 행은 보기 순서대로 기록되고, `aVisibleOnly`를 지정하면 보이는 열만 기록됩니다.

참고 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- TsgcGridSource_DataSet
- DataSet 바인딩
- JSON과 CSV

.NET 제자리 편집

이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.
ReadOnly가 false이고 열이 **Editable**이면 세 그리드는 셀을 제자리에서 편집합니다. 셀 위에 텍스트 상자가 열리고, 열에 **PickList**가 있으면 콤보 상자가 열립니다. 불리언 셀은 스페이스 키나 클릭으로 전환됩니다. 편집은 F2, Enter, 문자 입력, 더블 클릭 또는 **StartEdit**로 시작되고, Enter, Tab 또는 편집기가 포커스를 잃을 때 확정되며, Escape 또는 **CancelEdit**로 취소됩니다.

이벤트

이벤트	설명
OnBeforeEdit	편집기가 열리기 전에 발생합니다. `aAllow` 를 false로 설정하면 편집을 거부합니다.
OnValidate	값이 기록되기 전에, 이미 열의 **DataType**으로 변환된 상태로 발생합니다. `aValue` 를 변경하거나 `aAccept` 를 false로 설정해 거부합니다. 거부하면 편집기는 열린 채로 남습니다.
OnEdited	**Data.UpdateCell**로 값이 기록된 뒤에 이전 값과 새 값과 함께 발생합니다.
OnEditError	텍스트를 열의 **DataType**으로 변환할 수 없습니다. `aMessage` 를 변경해 메시지를 설정합니다.

편집 이벤트의 열 인수는 보이는 위치가 아니라 **Columns**의 인덱스입니다.

예제

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcGrid;

namespace GridEditing
{
    internal class MainForm : Form
    {
        private readonly TsgcWinFormsGrid Grid = new TsgcWinFormsGrid();
        private readonly Label lblStatus = new Label();

        public MainForm()
        {
            Text = "sgcGrid in-place editing";
            ClientSize = new Size(640, 300);

            Grid.Dock = DockStyle.Fill;
            Grid.SelectionMode = TsgcGridSelectionMode.gsmCell;
            Grid.Columns.Add("sku", "SKU");
            TsgcGridColumn oName = Grid.Columns.Add("name", "Product");
            oName.Editable = true;
            TsgcGridColumn oStock = Grid.Columns.Add("stock", "Stock", TsgcGridDataType.gdtInteger);
            oStock.Editable = true;
            TsgcGridColumn oStatus = Grid.Columns.Add("status", "Status");
            oStatus.Editable = true;
            oStatus.PickList.AddRange(new[] { "Active", "On hold", "Retired" });
            TsgcGridColumn oOnSale = Grid.Columns.Add("onsale", "On sale", TsgcGridDataType.gdtBoolean);
            oOnSale.Editable = true;
            Grid.Data.AppendRow("A-100", "A-100", "Keyboard", 42, "Active", false);
            Grid.Data.AppendRow("A-200", "A-200", "Mouse", 120, "Active", true);
            Grid.Data.AppendRow("B-300", "B-300", "Monitor", 8, "On hold", false);
```

```
            // editing is off until ReadOnly is false, and then only Editable columns open an editor
            Grid.ReadOnly = false;
            Grid.OnValidate += DoValidate;
            Grid.OnEdited += (Sender, aRow, aCol, aOldValue, aNewValue) =>
                lblStatus.Text = Grid.Data.KeyOf(aRow) + ": " + aOldValue + " -> " + aNewValue;
            Grid.OnEditError += (object Sender, int aRow, int aCol, string aText, ref string aMessage) =>
                aMessage = "\"" + aText + "\" is not a number";

            lblStatus.Dock = DockStyle.Bottom;
            lblStatus.Text = "F2, Enter, a key or a double click edits the cell";
            Controls.Add(Grid);
            Controls.Add(lblStatus);
        }

        // the value is already converted to the column DataType
        private void DoValidate(object Sender, int aRow, int aCol, ref object aValue, ref bool aAccept)
        {
            if (Grid.Columns[aCol].FieldName == "stock" && Convert.ToInt64(aValue) < 0)
            {
                lblStatus.Text = "The stock cannot be negative";
                aAccept = false;
            }
            else if (aValue is string)
                aValue = ((string)aValue).Trim();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

실시간 피드는 열려 있는 편집기를 덮어쓰지 않습니다. 행을 다시 정렬하면 편집기는 키를 통해 해당 행을 따라가고, 행이 삭제되면 닫힙니다. WinForms에서 편집기는 자식 컨트롤인 **TsgcWinFormsGridEdit**와 **TsgcWinFormsGridCombo**이고, WPF와 Avalonia에서는 호스팅되는 `TextBox`와 `ComboBox`입니다. 한 프로젝트에서 두 에디션을 모두 빌드할 때는 편집 코드를 `#if !SGC_COMMUNITY`로 감싸십시오.

참고 항목

- .NET 개요
- .NET 클래스
- WinForms 빠른 시작
- 제자리 편집
- 선택과 키보드
- 에디션

.NET 실시간 피드

이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.
esegece.sgcGrid.sgcWebSockets 패키지는 sgcWebSockets .NET 연결에서 그리드로 데이터를 공급합니다. **TsgcGridFeedSource** 가 기본 클래스이고, **TsgcGridFeedSource_WebSocket** 은 **TsgcWebSocketClient** 를, **TsgcGridFeedSource_MQTT**는 MQTT 클라이언트의 게시를, **TsgcGridFeedSource_SSE**는 서버 전송 이벤트를 읽습니다. 모든 메시지는 JSON으로 읽혀 그리드 행에 매핑됩니다.

매핑

속성	설명
Grid, GridData	대상: public **Data** 속성으로 찾는 그리드 컨트롤 또는 **TsgcGridData**. **GridData**가 우선합니다.
RowsPath	메시지 안에서 행의 경로: 행 객체의 배열 또는 하나의 행 객체. 메시지 자체가 행이거나 배열이면 비워 둡니다.
KeyPath	행 객체 안에서 행 키의 경로. 예: `symbol` 또는 `id.value` . 키가 없는 행은 건너뛰고 **OnError**로 보고합니다.
Fields	한 줄에 하나의 `FieldName=json.path` . 예: `price=quote.last` 또는 `bid=book.bid[0]` . `=` 가 없는 줄은 같은 이름의 경로를 읽습니다. **Fields**가 비어 있으면 모든 열이 각자의 **FieldName** 경로에서 읽힙니다. 행에 있는 경로만 기록되므로 메시지에 부분 갱신을 담을 수 있습니다.
DeletePath	행 객체 안의 불리언 경로: true이면 해당 키의 행을 삭제합니다.
MaxRowsPerMessage	한 메시지에서 적용할 행 수의 상한. 0이면 제한이 없습니다. 초과한 행은 **OnError**로 보고합니다.
Active	False이면 모든 메시지를 무시합니다.
OnRow, OnError	행이 적용되려는 시점(`aAccept` 를 false로 설정하면 건너뜁니다) 또는 메시지를 적용할 수 없을 때 발생합니다.

JSON 숫자는 double로, 불리언은 bool로, 문자열은 string으로, null은 그리드 null로 기록됩니다.

소스

클래스	Client 속성	읽는 대상
TsgcGridFeedSource_WebSocket	**Client** (`TsgcWebSocketClient`)	모든 텍스트 메시지.
TsgcGridFeedSource_MQTT	**Client** (`TsgcWSProtocol_MQTT_Client`)	토픽이 **Topic**과 일치하는 게시. `+` 와 `#` 와일드카드를 지원합니다.
TsgcGridFeedSource_SSE	**Client** (`TsgcHTTP1Client`) or **SSEClient** (`TsgcSSEClient`)	모든 서버 전송 이벤트의 데이터. **SSEClient**는 .NET에서 추가된 것입니다.

소스는 애플리케이션의 핸들러와 나란히 클라이언트의 이벤트를 구독하며, 핸들러를 대체하지 않습니다.

예제

이 콘솔 프로그램은 같은 프로세스 안의 루프백 인터페이스에서 **TsgcWebSocketServer**를 시작하고, 세 개의 메시지로 그리드 모델에 데이터를 공급합니다. 행 묶음, 삭제가 포함된 부분 갱신, 단일 행 객체입니다.

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using esegece.sgcGrid;
using esegece.sgcWebSockets;

namespace GridLiveFeed
{
    internal static class Program
    {
        private static int FreePort()
        {
            TcpListener oListener = new TcpListener(IPAddress.Loopback, 0);
            oListener.Start();
            try
            {
                return ((IPEndPoint)oListener.LocalEndpoint).Port;
            }
            finally
            {
                oListener.Stop();
            }
        }

        private static int Main()
        {
            // a server in the same process, on the loopback interface only
            int vPort = FreePort();
            TsgcWebSocketServer oServer = new TsgcWebSocketServer();
            oServer.NotifyEvents = TwsNotifyEvent.neNoSync;
            oServer.Port = vPort;
            oServer.Bindings.Clear();
            oServer.Bindings.NewBinding("127.0.0.1", vPort);
            oServer.Active = true;

            TsgcWebSocketClient oClient = new TsgcWebSocketClient();
            oClient.NotifyEvents = TwsNotifyEvent.neNoSync;
            oClient.Host = "127.0.0.1";
            oClient.Port = vPort;

            // in an application set Grid to the grid control instead of GridData
            TsgcGridData oData = new TsgcGridData();
            oData.UseInvariantFormat();
            oData.Columns.Add("symbol", "Symbol");
            oData.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat).DisplayFormat = "0.00";
            oData.Columns.Add("volume", "Volume", TsgcGridDataType.gdtInteger);

            CountdownEvent oRows = new CountdownEvent(6);
            TsgcGridFeedSource_WebSocket oSource = new TsgcGridFeedSource_WebSocket();
            oSource.GridData = oData;
            oSource.RowsPath = "quotes";
            oSource.KeyPath = "symbol";
            oSource.Fields.Add("symbol");
            oSource.Fields.Add("price=last.price");
            oSource.Fields.Add("volume=last.size");
            oSource.DeletePath = "delisted";
            oSource.OnRow = (object Sender, string aKey, ref bool aAccept) =>
            {
                if (!oRows.IsSet)
                    oRows.Signal();
            };
```

```
            oSource.Client = oClient;
            try
            {
                if (!oClient.Connect(5000))
                    return 1;
                Thread.Sleep(200);
                oServer.Broadcast("{\"quotes\":[" +
                    "{\"symbol\":\"AAPL\",\"last\":{\"price\":227.5,\"size\":300}}," +
                    "{\"symbol\":\"MSFT\",\"last\":{\"price\":431.1,\"size\":120}}," +
                    "{\"symbol\":\"NVDA\",\"last\":{\"price\":121.4,\"size\":900}}]}");
                // a partial update and a delete in one message
                oServer.Broadcast("{\"quotes\":[" +
                    "{\"symbol\":\"AAPL\",\"last\":{\"price\":228.05}}," +
                    "{\"symbol\":\"MSFT\",\"delisted\":true}]}");
                // one row object without an array
                oServer.Broadcast("{\"quotes\":{\"symbol\":\"JPM\",\"last\":{\"price\":212.8,\"size\":50}}}");
                bool vDone = oRows.Wait(5000);
                for (int i = 0; i < oData.ViewCount(); i++)
                    Console.WriteLine(oData.KeyOf(i) + " " + oData.CellText(i, 1) + " x " + oData.CellText(i, 2));
                return vDone ? 0 : 1;
            }
            finally
            {
                oSource.Dispose();
                oClient.Active = false;
                oServer.Active = false;
            }
        }
    }
}
```

프로그램의 출력:

```
AAPL 228.05 x 300
NVDA 121.40 x 900
JPM 212.80 x 50
```

스레딩

한 메시지의 모든 행은 메시지가 도착한 스레드에서 한 쌍의 **BeginUpdate**와 **EndUpdate** 안에 적용됩니다. **TsgcGridData**는 스레드에 안전하고 그리드는 다시 그리기를 UI 스레드로 마샬링하므로 동기화가 필요 없습니다. **OnRow**와 **OnError**도 그 스레드에서 실행됩니다. 컨트롤을 건드리기 전에 UI 스레드로 마샬링하십시오.

참고 항목

- .NET 개요
- NuGet에서 설치
- .NET 클래스
- .NET 데모
- TsgcGridFeedSource
- TsgcGridFeedSource_WebSocket
- TsgcGridFeedSource_MQTT
- TsgcGridFeedSource_SSE
- 실시간 피드

.NET 그리드에 질문하기

이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.
esegece.sgcGrid.sgcHTML 패키지의 **TsgcGridAIQuery**는 자연어 질문을 그리드의 필터, 정렬, 강조 표시로 바꿉니다. 질문에는 esegece.sgcHTML의 **TsgcHTMLAIAssistant**가 답합니다. 어시스턴트에 공급자가 있으면 언어 모델이, **Assistant**가 null이거나 공급자가 없으면 규칙 기반 파서가 답합니다. 규칙 기반 파서는 네트워크가 필요 없으며 `price greater than 100`, `sort by volume descending`, `highlight change less than 0` 같은 질문을 이해합니다.

답변이 적용되는 방식

- 답변은 무언가를 적용하기 전에 그리드의 열(**FieldName**, **Caption**, **DataType**)과 대조하여 검증됩니다.
- 필터는 **Data.OnFilterRow**를 통해 AND로 결합됩니다. 직접 할당한 핸들러는 유지되어 먼저 호출되며, 그 핸들러가 거부한 행은 계속 거부됩니다.
- 정렬은 열의 **SortOrder**와 **SortIndex**를 교체합니다.
- 강조 표시는 어시스턴트 스타일의 색으로 **ApplyToRow**를 지정하여 그리드의 **FormatRules**에 추가됩니다.
- 그룹화와 행 수 제한은 그리드가 지원하지 않으므로 **LastExplanation**으로 보고됩니다.
- **Ask**를 호출할 때마다 먼저 이전 결과를 되돌립니다. **Reset**은 이전 필터 핸들러와 열 정렬을 복원하고 쿼리가 추가한 규칙만 제거합니다.

Ask는 동기 메서드이며 열과 서식 규칙을 변경하므로 UI 스레드에서 호출해야 합니다. 언어 모델을 사용하면 공급자가 답할 때까지 차단됩니다. 결과는 **OnApplied**와 **OnError**로 보고됩니다.

예제

쿼리의 **Grid**는 public **Data** 속성을 가진 모든 객체입니다. 강조 표시에는 public **FormatRules** 속성도 필요합니다. 그리드 컨트롤에는 둘 다 있습니다. 이 콘솔 프로그램은 직접 만든 작은 클래스를 사용하고 어시스턴트는 사용하지 않으므로 규칙 기반 파서가 답합니다.

```
using System;
using esegece.sgcGrid;

namespace GridAskYourGrid
{
    // any object with public Data and FormatRules properties can be the Grid of the
    // query, TsgcWinFormsGrid, TsgcWpfGrid and TsgcAvaloniaGrid expose both
    internal class ReportGrid
    {
        public TsgcGridData Data { get; } = new TsgcGridData();
        public TsgcGridFormatRules FormatRules { get; } = new TsgcGridFormatRules(null);
    }

    internal static class Program
    {
        private static void Main()
        {
            ReportGrid oGrid = new ReportGrid();
            oGrid.Data.UseInvariantFormat();
            oGrid.Data.Columns.Add("symbol", "Symbol");
            oGrid.Data.Columns.Add("sector", "Sector");
            oGrid.Data.Columns.Add("price", "Price", TsgcGridDataType.gdtFloat);
            oGrid.Data.Columns.Add("change", "Change", TsgcGridDataType.gdtFloat);
            oGrid.Data.Columns.Add("volume", "Volume", TsgcGridDataType.gdtInteger);
            oGrid.Data.AppendRow("AAPL", "AAPL", "Tech", 227.50, 1.20, 51000);
            oGrid.Data.AppendRow("MSFT", "MSFT", "Tech", 431.10, -2.35, 22000);
            oGrid.Data.AppendRow("JPM", "JPM", "Finance", 212.80, 0.45, 9000);
            oGrid.Data.AppendRow("XOM", "XOM", "Energy", 118.25, -0.80, 15000);
            oGrid.Data.AppendRow("NVDA", "NVDA", "Tech", 121.40, 3.10, 98000);

            // no Assistant: the rule-based parser answers, without any network call
            TsgcGridAIQuery oQuery = new TsgcGridAIQuery();
            oQuery.Grid = oGrid;
```

```
            if (oQuery.Ask("price greater than 120 sort by volume descending highlight change less than 0"))
            {
                Console.WriteLine(oQuery.LastExplanation().Trim());
                for (int i = 0; i < oGrid.Data.ViewCount(); i++)
                    Console.WriteLine(oGrid.Data.KeyOf(i) + " " + oGrid.Data.CellText(i, 4));
                Console.WriteLine(oGrid.FormatRules.Count + " highlight rule");
            }

            // Reset restores the filter, the sort and the format rules of the grid
            oQuery.Reset();
            Console.WriteLine(oGrid.Data.ViewCount() + " rows after Reset");
            oQuery.Dispose();
        }
    }
}
```

프로그램의 출력:

```
Filter: Price > 120
Sort: Volume descending
Highlight (warning): Change < 0
NVDA 98000
AAPL 51000
MSFT 22000
JPM 9000
1 highlight rule
5 rows after Reset
```

언어 모델을 사용하려면 공급자가 구성된 **TsgcHTMLAIAssistant**를 **Assistant**에 할당하십시오.

참고 항목

- .NET 개요
- NuGet에서 설치
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 데모
- TsgcGridAIQuery
- 그리드에 질문하기

.NET 성능

그리드는 보이는 행과 열만 그리고, 데이터가 아무리 빠르게 바뀌어도 초당 최대 **MaxFPS**회만 다시 그립니다. WinForms 호스트는 테스트 스위트의 성능 게이트로 측정합니다. 이 게이트는 Delphi 성능 테스트를 옮긴 것으로 호스트 그리기 경로와 실시간 피드가 추가되어 있습니다.

장면

- **TsgcWinFormsGrid**의 `1920 x 1080` 클라이언트 영역, 장치 독립 픽셀 160 너비의 12개 열(텍스트, 정수, 실수, 통화, 시간, 불리언), 행 높이 23.
- 키가 있는 행 100,000개.
- 워커 스레드가 초당 1000건의 갱신을 보내며, 그중 절반은 보기 상단 근처의 행에 대한 것입니다.

프레임 시간

프레임 시간은 피드가 실행되는 동안 60프레임에 걸친 컨트롤의 전체 그리기(렌더링과 블릿)입니다. 예산은 64비트 프로세스에서 프레임당 16 ms(60 fps에서 한 프레임), 32비트 프로세스에서 25 ms입니다.

런타임	프레임 시간
.NET 8, sgcUICore 글리프 커버리지 마스크 도입 전	44 ms
.NET 8	11.4 ~ 13.1 ms
.NET Framework 4.7.2	7.3 ~ 7.7 ms

첫 번째 행은 sgcUICore 글리프 커버리지 마스크를 도입하게 된 측정입니다. 모든 셀 텍스트가 Skia를 통해 윤곽 전체를 채우고 있었습니다.

페이싱

게이트의 두 번째 단계는 **MaxFPS**를 60으로 하고 실제 메시지 루프에서 3초 동안 지속적인 다시 그리기 속도를 측정하며, 피드가 자신의 속도의 90퍼센트 이상을 유지하는지와 다시 그리기가 **MaxFPS**를 넘지 않는지 확인합니다. 아래 범위는 게이트를 세 번 실행한 결과입니다.

런타임	다시 그리기 속도
.NET 8	59.5 ~ 59.7 fps
.NET Framework 4.7.2	59.8 ~ 60 fps

.NET 8에서 두 프레임 사이 간격의 95번째 백분위수는 21.4 ~ 22.2 ms입니다.

게이트 실행

게이트는 타이밍에 민감하고 다른 작업이 없는 컴퓨터가 필요하므로 요청할 때만 실행됩니다. 환경 변수 `SGC_GRID_RUN_PERF=1`을 설정하고 WinForms 테스트의 `Perf` 범주를 실행합니다.
수치는 테스트 어셈블리 옆의 `sgcGridQA_perf.txt`에 추가됩니다. 게이트는 참고용으로 같은 프레임의 헤드리스 **SaveToPNG**와 **Render** 시간도 기록합니다. 이 프레임의 헤드리스 **Render**는 5.0 ms가 걸립니다.

참고 항목

- .NET 개요
- WinForms 빠른 시작
- .NET 헤드리스 렌더링
- 성능
- 실시간 갱신과 스레딩

.NET 그룹화와 요약

행은 하나 이상의 열을 기준으로 그룹화할 수 있습니다. 각 그룹에는 펼침 화살표, 멤버 텍스트, 행 수, 열 요약이 표시되는 그룹 행이 생깁니다. 바닥글에는 모든 행에 대한 동일한 요약이 표시되며, 고정 열은 다른 열이 가로로 스크롤되어도 계속 표시됩니다. 그룹화, 요약, 바닥글, 고정 열은 **TsgcWinFormsGrid**, **TsgcWpfGrid**, **TsgcAvaloniaGrid**에서 두 에디션 모두에 포함됩니다.
그룹화는 뷰 상태입니다. **TsgcGridData**는 평면 행과 정렬을 그대로 유지하고 **IsgcUITable**은 여전히 그룹화되지 않은 표시 행을 보고하므로, 그리드에 바인딩된 sgcCharts 차트는 아무 변화도 보지 못합니다. 그리드는 **Lines** 속성으로 사용할 수 있는 **TsgcGridLineView**를 통해 행을 배치합니다.

열 속성

속성	Default	설명
GroupIndex	-1	0 이상의 값은 해당 수준에서 그 열을 기준으로 행을 그룹화합니다. 수준은 **GroupIndex** 순서로, 그다음 열 인덱스 순서로 정렬됩니다. **GroupBy**, **ClearGrouping**, 그룹 패널이 이 값을 대신 설정합니다.
SummaryType	`gsuNone`	`gsuSum`, `gsuCount`, `gsuAverage`, `gsuMin`, `gsuMax` 중 하나입니다. 요약은 모든 그룹 행과 바닥글에 표시됩니다.
SummaryFormat	비어 있음	요약의 서식 패턴으로, **DisplayFormat**과 같은 형식입니다. 예를 들어 `#,##0.00` 이며, 리터럴 텍스트는 큰따옴표로 묶습니다(`"Total: "#,##0.00`). 비어 있으면 **DisplayFormat**을 사용하고, 개수에는 `0` 을 사용합니다.

이 속성을 변경하면 열의 **OnChange**만 발생하며 데이터는 다시 로드되지 않습니다. 요약 값은 값이 없으면 `null`, 개수는 `int`, 합계, 평균, 최솟값, 최댓값은 `double`, 날짜 열의 최솟값이나 최댓값은 `DateTime`입니다.

그리드 멤버

같은 멤버가 **TsgcWinFormsGrid**, **TsgcWpfGrid**, **TsgcAvaloniaGrid**에 있습니다. WPF와 Avalonia에서는 7개의 옵션이 종속성 속성과 스타일 속성이므로 XAML에서 설정하고 바인딩하고 스타일을 지정할 수 있습니다.

멤버	형식	설명
ShowGroupPanel, GroupPanelHeight	bool, int	헤더 위의 패널입니다. 열 헤더를 여기로 끌어 놓으면 해당 열로 그룹화됩니다. 기본적으로 숨겨져 있으며 높이는 36픽셀입니다.
GroupIndent	int	각 그룹 수준의 들여쓰기입니다. 기본값은 20입니다.
GroupsExpanded	bool	새 그룹의 상태입니다. 기본값은 true입니다.
GroupSummaries	bool	그룹 행에 요약을 표시합니다. 기본값은 true입니다.
ShowFooter	bool	모든 행에 대한 요약이 있는 바닥글을 표시합니다. 기본값은 false입니다.
FrozenColumns	int	다른 열이 스크롤되어도 계속 표시되는 앞쪽 표시 열의 수입니다. 기본값은 0입니다.
GroupBy, ClearGrouping	method	`GroupBy("region", "category")` 는 이 필드들을 이 순서로 그룹화하고 다른 모든

		GroupIndex를 지웁니다. **ClearGrouping**은 평면 행을 다시 표시합니다.
ExpandAll, CollapseAll, ExpandGroup, CollapseGroup	method	모든 그룹을 펼치거나 접거나, 인덱스로 지정한 그룹 하나를 펼치거나 접습니다.
GroupCount, GroupInfo	method	그룹의 수와 그룹의 **TsgcGridGroupInfo**입니다.
FooterSummary, FooterSummaryText	method	열의 바닥글 값(없으면 null)과 바닥글에 표시되는 텍스트입니다.
Lines	TsgcGridLineView	그리드의 라인 뷰입니다. 그룹 행, 데이터 행, 그리고 Registered 에디션에서는 트리 행과 세부 정보 밴드를 포함합니다.
FocusedLine	int	포커스가 있는 라인입니다. 그룹 행에서는 **SelectedRow**가 -1이며 **FocusedLine**이 라인을 알려 줍니다.
OnGroupingChange	EventHandler	그룹화된 열이나 그 순서가 코드 또는 그룹 패널에서 변경되었습니다.
OnGroupExpandChange	TsgcGridGroupExpandEvent	그룹을 펼치거나 접었습니다. **ExpandAll**과 **CollapseAll** 이후에는 `aGroup` 이 -1입니다.
OnGetGroupText	TsgcGridGetGroupTextEvent	그룹 행을 그리기 전에 캡션을 변경합니다.

행 인덱스는 계속 표시 행입니다. **SelectedRow**, **OnCellClick**, **Data**는 그룹화하지 않을 때와 같은 행을 사용합니다. **TopRow**는 표시 라인을 세며, 아무것도 그룹화되지 않았을 때는 표시 행과 같습니다.

그룹 정보

Field	설명
Level	가장 바깥쪽 수준은 0입니다.
Parent	부모 그룹입니다. 수준 0에서는 -1입니다.
Column	그룹화된 열의 **Columns** 인덱스입니다.
RowCount	그룹에 속한 표시 행의 수이며, 자식 그룹의 행도 포함합니다.
FirstRow	그룹의 첫 번째 행에 해당하는 표시 행입니다.
Text, Value, Blank	멤버 텍스트, 형식이 지정된 값, 그리고 빈 멤버이면 true이며, 빈 멤버는 `(blank)` 로 표시됩니다.
Expanded	그룹이 펼쳐져 있습니다.
Line	그룹 행의 라인입니다. 접힌 그룹 안에서는 -1입니다.
Path	수준 0부터의 멤버 텍스트입니다.

마우스와 키보드

- 열 헤더를 그룹 패널로 끌어 놓으면 그 열로 그룹화되며, 놓일 위치는 표시자로 나타납니다. 패널 안에서 칩을 끌면 순서가 바뀌고, 패널 밖으로 끌면 제거되며, 클릭하면 그룹의 정렬이 반대로 바뀝니다. Esc 키를 누르면 끌기가 취소됩니다.

- 그룹 행의 펼침 화살표를 클릭하거나 행을 더블 클릭하면 펼쳐지거나 접힙니다.
- 그룹 행에서는 오른쪽 키와 왼쪽 키로 펼치고 접으며, + 키와 빼기 키도 같은 동작을 하고, * 키는 모든 그룹을 펼치고, Enter 키는 그룹을 전환합니다. 그룹 행에서 Ctrl+C를 누르면 캡션이 복사됩니다.
- Avalonia에서는 터치 입력으로도 그리드를 이동할 수 있으며, 펼침 화살표를 탭하면 클릭처럼 동작합니다.

예제

주문을 지역과 카테고리로 그룹화하고, 모든 그룹 행과 바닥글에 합계와 평균을 표시하며, 고정 열 두 개와 지역 그룹용 사용자 지정 캡션을 사용하는 WinForms 폼입니다.

```
using System;
using System.Windows.Forms;
using esegece.sgcGrid;

namespace GridGroupingWinForms
{
    public class SalesForm : Form
    {
        public readonly TsgcWinFormsGrid Grid = new TsgcWinFormsGrid();
        private readonly Label lblStatus = new Label();

        public SalesForm()
        {
            Text = "Sales";
            Width = 900;
            Height = 600;
            Grid.Dock = DockStyle.Fill;
            lblStatus.Dock = DockStyle.Bottom;
            Controls.Add(Grid);
            Controls.Add(lblStatus);

            Grid.Columns.Add("order", "Order");
            Grid.Columns.Add("customer", "Customer");
            Grid.Columns.Add("region", "Region");
            Grid.Columns.Add("category", "Category");
            Grid.Columns.Add("qty", "Qty", TsgcGridDataType.gdtInteger).SummaryType = TsgcGridSummaryType.gsuSum;
            TsgcGridColumn oColumn = Grid.Columns.Add("price", "Price", TsgcGridDataType.gdtCurrency);
            oColumn.DisplayFormat = "#,##0.00";
            oColumn.SummaryType = TsgcGridSummaryType.gsuAverage;
            oColumn.SummaryFormat = "\"avg \"#,##0.00";

            Grid.ShowGroupPanel = true;  // drag a header here to group by it
            Grid.ShowFooter = true;      // the column summaries over all rows
            Grid.FrozenColumns = 2;      // Order and Customer stay in view
            Grid.GroupsExpanded = false; // new groups start collapsed

            Grid.OnGetGroupText += GridGetGroupText;
            Grid.OnGroupExpandChange += GridGroupExpandChange;
            Grid.OnGroupingChange += (s, e) => lblStatus.Text = "Grouping changed";

            Grid.Data.BeginUpdate();
            try
            {
                Grid.Data.AppendRow("1", "1", "Acme", "North", "Books", 2, 15.0);
                Grid.Data.AppendRow("2", "2", "Brick", "North", "Games", 1, 60.0);
                Grid.Data.AppendRow("3", "3", "Acme", "South", "Books", 4, 13.1);
            }
            finally
            {
                Grid.Data.EndUpdate();
            }
            Grid.GroupBy("region", "category");
        }

        private void GridGetGroupText(object Sender, int aGroup, ref string aText)
        {
```

```
            TsgcGridGroupInfo vInfo = Grid.GroupInfo(aGroup);
            if (vInfo.Level == 0)
                aText = "Region " + vInfo.Text + ", " + vInfo.RowCount + " orders";
        }

        private void GridGroupExpandChange(object Sender, int aGroup, bool aExpanded)
        {
            // aGroup is -1 after ExpandAll and CollapseAll
            lblStatus.Text = "Total qty: " + Grid.FooterSummaryText(Grid.Columns.IndexOfField("qty"));
        }
    }
}
```

그룹화된 상태에서도 데이터는 어느 스레드에서든 쓸 수 있습니다. 그리드는 UI 스레드에서 그룹과 요약을 다시 동기화하고 작업 속도를 조절하므로 인터페이스가 계속 응답합니다.

컨트롤 없이 그룹화

TsgcGridLineView는 서비스나 콘솔 프로그램처럼 **TsgcGridData**만으로도 동작합니다. **GroupSummary**는 그룹과 열의 요약 값을 반환하고, **SetGroupExpanded**는 그룹을 펼치거나 접습니다.

```
using System;
using esegece.sgcGrid;

namespace GridGrouping
{
    public static class SalesGroups
    {
        public static TsgcGridData CreateSales()
        {
            TsgcGridData oData = new TsgcGridData();
            oData.Columns.Add("order", "Order");
            // two group levels: Region, then Category
            oData.Columns.Add("region", "Region").GroupIndex = 0;
            oData.Columns.Add("category", "Category").GroupIndex = 1;
            // summaries shown on every group row and in the footer
            oData.Columns.Add("status", "Status").SummaryType = TsgcGridSummaryType.gsuCount;
            oData.Columns.Add("qty", "Qty", TsgcGridDataType.gdtInteger).SummaryType = TsgcGridSummaryType.gsuSum;
            TsgcGridColumn oColumn = oData.Columns.Add("total", "Total", TsgcGridDataType.gdtCurrency);
            oColumn.DisplayFormat = "#,##0.00";
            oColumn.SummaryType = TsgcGridSummaryType.gsuSum;
            oColumn.SummaryFormat = "\"Total: \"#,##0.00";

            oData.BeginUpdate();
            try
            {
                oData.AppendRow("1", "1", "North", "Books", "Open", 2, 30.0);
                oData.AppendRow("2", "2", "North", "Games", "Paid", 1, 60.0);
                oData.AppendRow("3", "3", "North", "Books", "Paid", 4, 52.5);
                oData.AppendRow("4", "4", "South", "Books", "Open", 3, 41.0);
                oData.AppendRow("5", "5", "South", "Games", "Paid", 5, 99.9);
            }
            finally
            {
                oData.EndUpdate();
            }
            return oData;
        }

        public static void PrintGroups(TsgcGridData aData)
        {
            // the line view lays the rows out under their group rows
            TsgcGridLineView oLines = new TsgcGridLineView(aData);
            int vTotal = aData.Columns.IndexOfField("total");
            Console.WriteLine("{0} levels, {1} groups, {2} lines",
                oLines.GroupLevels(), oLines.GroupCount(), oLines.Count());
```

```
            for (int i = 0; i < oLines.GroupCount(); i++)
            {
                TsgcGridGroupInfo vInfo = oLines.GroupInfo(i);
                Console.WriteLine("{0}{1}: {2} rows, total {3}", new string(' ', 2 * vInfo.Level),
                    vInfo.Text, vInfo.RowCount, oLines.GroupSummary(i, vTotal));
            }
            // collapse the first group: its rows leave the line view
            oLines.SetGroupExpanded(0, false);
            Console.WriteLine("{0} lines with North collapsed", oLines.Count());
        }
    }
}
```

출력:

```
2 levels, 6 groups, 11 lines
North: 3 rows, total 142.5
  Books: 2 rows, total 82.5
  Games: 1 rows, total 60
South: 2 rows, total 140.9
  Books: 1 rows, total 41
  Games: 1 rows, total 99.9
6 lines with North collapsed
```

데이터와 라인 뷰는 **TsgcGridXLSXWriter**가 기록하는 대상이기도 합니다. .NET Excel 내보내기를 참고하십시오.

참고 항목

- .NET 개요
- .NET 클래스
- .NET 트리 행과 마스터 디테일
- .NET Excel 내보내기
- .NET 데모
- TsgcGridColumn
- 그룹화
- 요약
- 고정 열
- TsgcGridLineView

.NET 트리 행과 마스터 디테일

이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.
TsgcWinFormsGrid, **TsgcWpfGrid**, **TsgcAvaloniaGrid**는 키 필드와 부모 필드로 구성한 트리로 행을 표시하며, 마스터 행 아래에 두 번째 소스의 자식 행이 있는 세부 정보 밴드를 엽니다. 트리 모드와 그룹화는 동시에 활성화되지 않습니다. 트리 모드에서는 그룹화가 무시되고, 열의 **GroupIndex**는 유지되며, 그룹 패널은 숨겨집니다.

트리 행

Tree 속성은 **TsgcGridTreeOptions**입니다. **KeyField**와 **ParentField**가 모두 설정되면 트리가 켜집니다.

속성	Default	설명
KeyField	비어 있음	행의 키가 있는 열입니다.
ParentField	비어 있음	부모 행의 키가 있는 열입니다. 부모 키가 비어 있거나 알 수 없거나 자기 자신인 행은 루트입니다.
TreeField	비어 있음	들여쓰기와 펼침 버튼을 표시하는 열입니다. 비어 있으면 첫 번째 표시 열을 사용합니다.
Indent	16	각 수준의 들여쓰기입니다.
Expanded	false	새 노드의 상태입니다.

필드는 그리드의 열이어야 하며 숨겨져 있어도 됩니다. 키는 텍스트로 비교됩니다. 정수와 소수부가 없는 실수는 정수 텍스트를 사용하므로 `int` 키와 `double` 부모가 일치합니다. 여러 행이 같은 키를 가지면 첫 번째 행이 자식을 소유하고, 순환은 처음 반복되는 노드에서 끊어집니다. 형제 노드는 표시 순서를 유지하므로 정렬하면 형제 노드가 정렬되며, 필터를 사용하면 표시되는 행은 가장 가까운 표시 조상 아래에 놓입니다. 펼침 상태는 키별로 유지되므로 다시 로드해도 보존됩니다.

멤버	설명
ExpandNode, CollapseNode	표시 행의 노드를 펼치거나 접습니다.
ExpandAll, CollapseAll	트리 모드에서는 모든 노드를 펼치거나 접습니다.
OnTreeExpandChange	노드를 펼치거나 접었습니다. **ExpandAll**과 **CollapseAll** 이후에는 `aRow` 가 -1입니다.

노드의 버튼을 클릭하면 펼치거나 접습니다. 노드에서 오른쪽 키는 노드를 펼치거나 첫 번째 자식으로 이동하고, 왼쪽 키는 노드를 접거나 부모로 이동합니다.

마스터 디테일

Detail 속성은 **TsgcGridDetailOptions**입니다. 펼침 열을 누르면 마스터 행 아래에 밴드가 열립니다. 밴드는 자체 헤더, 스크롤 위치, 선택을 가진 중첩 그리드입니다. 세부 정보는 한 수준까지만 중첩되며 읽기 전용입니다.

속성	Default	설명
Active	false	펼침 열을 표시합니다.
Source	null	세부 행입니다. **TsgcGridDataStore**, **TsgcGridData**, 또는 **TsgcGridData** 형식의 public **Data** 속성을 가진 모든 개체(예: 다른 그리드)입니다.

MasterKeyField	비어 있음	값을 세부 행에서 찾는 마스터 열입니다.
DetailKeyField	비어 있음	마스터 키를 가진 소스의 열입니다.
Lines	8	그리드 행 단위의 밴드 높이이며, 중첩된 헤더를 포함합니다.
Columns	비어 있음	밴드의 열입니다. 비어 있으면 소스 열의 복사본을 사용합니다.

멤버	설명
ExpandDetail, CollapseDetail, DetailExpanded	표시 행의 밴드를 열고, 닫고, 상태를 확인합니다.
DetailData	열려 있는 밴드의 **TsgcGridData**이며, 밴드가 닫혀 있으면 null입니다.
OnDetailExpandChange	밴드를 열거나 닫았습니다.
OnDetailCellClick	밴드의 셀을 클릭했습니다. 마스터 행, 세부 행, 세부 열이 전달됩니다.
OnGetDetailData	밴드를 채우기 전에 발생합니다. 코드로 밴드를 채우려면 `aData` 에 행을 쓰고 `aHandled` 를 true로 설정합니다. 그렇지 않으면 밴드는 **Source** 에서 채워집니다.

펼침 상태는 마스터 키별로 유지됩니다. 소스의 행이 바뀌면 열려 있는 모든 밴드가 UI 스레드에서 제자리에서 다시 채워집니다. 밴드의 헤더를 클릭하면 정렬되고, 밴드 위에서 마우스 휠을 돌리면 스크롤되며, 아래쪽 키는 밴드를 지나 이동합니다. 밴드 이미지는 모든 그리드 패키지가 참조하는 esegece.sgcGrid.Rendering을 통해 그려집니다.
Source가 컴포넌트이면 해당 컴포넌트를 해제할 때 **Source**가 지워집니다. 다른 개체인 경우에는 개체가 사라지기 전에 **Source**를 null로 설정하십시오.

TsgcGridDataStore

TsgcGridDataStore는 세부 행을 보관하는 비시각적 컴포넌트이며 **Columns**, **Data**, **IsgcUITable**을 갖습니다. 코드로 채우거나 **TsgcGridSource_DataSet**의 **Grid**로 설정하여 `DataTable`을 로드합니다. **Grid**와 **DataSet**을 설정한 뒤 **Active**를 true로 설정하면 **LiveUpdates**가 저장소를 동기화 상태로 유지합니다. .NET DataTable 바인딩, JSON 및 CSV를 참고하십시오.

예제

조직도를 트리로 표시하고, **TsgcGridDataStore**에서 가져온 주문 밴드가 있는 고객을 표시하는 예제입니다.

```
using System;
using System.Windows.Forms;
using esegece.sgcGrid;

namespace GridTreeAndDetail
{
    public class TreeAndDetailForm : Form
    {
        public readonly TsgcWinFormsGrid GridTree = new TsgcWinFormsGrid();
        public readonly TsgcWinFormsGrid GridCustomers = new TsgcWinFormsGrid();
        public readonly TsgcGridDataStore OrdersStore = new TsgcGridDataStore();

        public TreeAndDetailForm()
        {
            GridTree.Dock = DockStyle.Left;
            GridTree.Width = 400;
            GridCustomers.Dock = DockStyle.Fill;
```

```
        Controls.Add(GridCustomers);
        Controls.Add(GridTree);

        // tree rows: every row names its parent row
        GridTree.Columns.Add("id", "Id", TsgcGridDataType.gdtInteger).Visible = false;
        GridTree.Columns.Add("manager", "Manager", TsgcGridDataType.gdtInteger).Visible = false;
        GridTree.Columns.Add("name", "Name");
        GridTree.Columns.Add("title", "Title");
        GridTree.Data.AppendRow("1", 1, null, "Ann Smith", "CEO");
        GridTree.Data.AppendRow("2", 2, 1, "Ben Clarke", "CTO");
        GridTree.Data.AppendRow("3", 3, 2, "Carl Diaz", "Developer");
        GridTree.Tree.KeyField = "id";
        GridTree.Tree.ParentField = "manager";
        GridTree.Tree.TreeField = "name";
        GridTree.Tree.Expanded = true;
        GridTree.OnTreeExpandChange += (s, aRow, aExpanded) =>
            Text = aRow < 0 ? "All nodes" : GridTree.Data.CellText(aRow, 2);

        // master rows
        GridCustomers.Columns.Add("customerid", "Customer", TsgcGridDataType.gdtInteger);
        GridCustomers.Columns.Add("name", "Name");
        GridCustomers.Data.AppendRow("10", 10, "Acme");
        GridCustomers.Data.AppendRow("20", 20, "Brick");

        // detail rows in a store, matched by customerid
        OrdersStore.Columns.Add("orderid", "Order", TsgcGridDataType.gdtInteger);
        OrdersStore.Columns.Add("customerid", "Customer", TsgcGridDataType.gdtInteger).Visible = false;
        OrdersStore.Columns.Add("amount", "Amount", TsgcGridDataType.gdtCurrency).DisplayFormat = "#,##0.00";
        OrdersStore.Data.AppendRow("1", 1, 10, 120.0);
        OrdersStore.Data.AppendRow("2", 2, 10, 80.5);
        OrdersStore.Data.AppendRow("3", 3, 20, 42.0);

        GridCustomers.Detail.Source = OrdersStore;
        GridCustomers.Detail.MasterKeyField = "customerid";
        GridCustomers.Detail.DetailKeyField = "customerid";
        GridCustomers.Detail.Lines = 6;
        GridCustomers.Detail.Active = true;
        GridCustomers.OnDetailCellClick += (s, aRow, aDetailRow, aDetailCol) =>
            Text = GridCustomers.DetailData(aRow).CellText(aDetailRow, aDetailCol);
    }
}

public class DetailFromCode
{
    // no Source: the rows of every band come from OnGetDetailData
    public static void Setup(TsgcWinFormsGrid aGrid)
    {
        aGrid.Detail.Columns.Add("item", "Item");
        aGrid.Detail.Columns.Add("qty", "Qty", TsgcGridDataType.gdtInteger);
        aGrid.Detail.MasterKeyField = "customerid";
        aGrid.Detail.Active = true;
        aGrid.OnGetDetailData += GridGetDetailData;
    }

    private static void GridGetDetailData(object Sender, int aRow, string aMasterKey,
        TsgcGridData aData, ref bool aHandled)
    {
        aData.BeginUpdate();
        try
        {
            aData.Clear();
            aData.AppendRow(aMasterKey + "-1", "Pens", 10);
            aData.AppendRow(aMasterKey + "-2", "Paper", 3);
        }
        finally
        {
            aData.EndUpdate();
```

```
            }
            aHandled = true;
        }
    }
}
```

두 번째 클래스는 코드만으로 밴드를 채웁니다. **Source**가 없으면 밴드는 옵션의 **Columns**를 사용하고, 모든 행은 **OnGetDetailData**에서 가져옵니다.

컨트롤 없이

TsgcGridLineView 도 같은 옵션을 받습니다. **TreeOptions** 에 **TsgcGridTreeOptions** 를, **DetailOptions** 에 **TsgcGridDetailOptions**를 할당하고, **SetNodeExpanded**와 **SetDetailExpanded**로 상태를 변경합니다. Excel 내보내기는 이를 사용하여 컨트롤 없이 트리를 기록합니다. .NET Excel 내보내기를 참고하십시오.

참고 항목

- .NET 클래스
- .NET 그룹화와 요약
- .NET Excel 내보내기
- .NET DataTable 바인딩, JSON 및 CSV
- .NET 데모
- TsgcGridSource_DataSet
- 트리 행
- 마스터 디테일
- TsgcGridDataStore

.NET Excel 내보내기

이 기능은 Registered 에디션에 포함됩니다. Community 패키지에는 들어 있지 않습니다.
TsgcWinFormsGrid, **TsgcWpfGrid**, **TsgcAvaloniaGrid**는 Excel 설치나 외부 라이브러리 없이 뷰를 Excel 통합 문서(.xlsx)로 저장합니다. 파일은 esegece.sgcGrid 패키지의 **TsgcGridXLSXWriter**가 기록하며, 컨트롤 없이도 동작합니다.

기록되는 내용

- 표시 열을 순서대로, 캡션은 굵은 헤더 행으로, 그리고 열 너비를 기록합니다.
- 숫자, 날짜, 부울 값은 형식이 지정된 셀로 기록되며 열의 **DisplayFormat**에 해당하는 Excel 숫자 서식이 적용되고, 텍스트는 공유 문자열로 기록됩니다.
- 그룹은 캡션이 있는 그룹 행으로 기록되며, 열 요약용 `SUBTOTAL` 수식이 각 그룹의 행 위에 Excel 윤곽 수준과 함께 기록됩니다. 접힌 그룹은 숨겨진 행으로 기록되므로 통합 문서는 그리드에 보이는 그대로 열립니다.
- 열에 요약이 있으면 데이터 아래에 전체 데이터 범위에 대한 `SUBTOTAL`이 있는 바닥글 행이 기록됩니다.
- 트리 행은 트리 열의 들여쓰기와 윤곽 수준으로 기록되며, 접힌 노드의 행은 숨겨진 행이 됩니다.
- 고정 열과 헤더 행은 틀 고정으로 기록됩니다.

세부 정보 밴드는 기록되지 않습니다. 정렬과 필터링은 적용되며, 통합 문서에는 표시 행이 담깁니다.

SaveToXLSX

`SaveToXLSX(Stream aStream, string aSheetName = "Sheet1")`와 `SaveToXLSX(string aFileName, string aSheetName = "Sheet1")`는 화면의 열 너비와 **FrozenColumns**로 뷰를 기록합니다. UI 스레드에서 호출하십시오.

```
using System.Windows.Forms;
using esegece.sgcGrid;

namespace GridExcelExport
{
    public static class GridExport
    {
        public static void Export(TsgcWinFormsGrid aGrid)
        {
            using (SaveFileDialog oDialog = new SaveFileDialog())
            {
                oDialog.Filter = "Excel workbook (*.xlsx)|*.xlsx";
                if (oDialog.ShowDialog() == DialogResult.OK)
                    aGrid.SaveToXLSX(oDialog.FileName, "Orders");
            }
        }
    }
}
```

TsgcGridXLSXWriter

멤버	설명
SheetName	워크시트의 이름입니다. 기본값은 `Sheet1` 입니다. Excel이 거부하는 문자는 `_` 가 되며 이름은 31자로 잘립니다.
FrozenColumns	계속 표시되는 앞쪽 표시 열의 수입니다. 기본값은 0입니다.
Summaries	그룹 행과 바닥글의 `SUBTOTAL` 행을 기록합니다. 기본값은 true입니다.
FileTime	파일 항목의 날짜와 시간입니다. 기본값은 생성 시간입니다.

SetColumnWidth	`SetColumnWidth(aCol, aDIPs)` : 열(**Columns** 인덱스)의 너비를 장치 독립적 픽셀로 지정하며, **Width** 대신 사용됩니다. 음수 값은 **Width**를 복원합니다.
SaveToStream, SaveToFile	`SaveToStream(aData, aLines, aStream)` 와 `SaveToFile(aData, aLines, aFileName)` 은 `aData` 의 통합 문서를 기록합니다. `aLines` 는 그룹화, 트리, 펼침 상태를 제공합니다. null이면 열의 **GroupIndex**로 만든 자체 라인 뷰를 사용합니다.

오류는 **EsgcGridXLSXError** 를 발생시킵니다. `sgcGrid_XLSX.sgcGridExcelNumberFormat` 은 데이터 형식과 **DisplayFormat**에 사용되는 Excel 숫자 서식을 반환합니다.

컨트롤 없는 예제

첫 번째 그룹을 접은 그룹화된 통합 문서와 트리로 표현한 조직도를 기록하는 콘솔 프로그램입니다. .NET 그룹화와 요약의 예제에 있는 `CreateSales` 메서드를 사용합니다.

```
using System;
using System.IO;
using esegece.sgcGrid;

namespace GridExcelExport
{
    public static class ExportSamples
    {
        public static void Run(string aOutDir)
        {
            // grouped sales: group rows with SUBTOTAL formulas and an outline
            TsgcGridData oSales = GridGrouping.SalesGroups.CreateSales();
            TsgcGridLineView oSalesLines = new TsgcGridLineView(oSales);
            oSalesLines.SetGroupExpanded(0, false); // written as hidden rows

            TsgcGridXLSXWriter oWriter = new TsgcGridXLSXWriter();
            oWriter.SheetName = "Sales";
            oWriter.FrozenColumns = 1;
            oWriter.SetColumnWidth(0, 80); // in DIPs, instead of Columns[0].Width
            oWriter.SaveToFile(oSales, oSalesLines, Path.Combine(aOutDir, "sales.xlsx"));

            // a tree: the levels become Excel outline levels
            TsgcGridData oOrg = new TsgcGridData();
            oOrg.Columns.Add("id", "Id", TsgcGridDataType.gdtInteger).Visible = false;
            oOrg.Columns.Add("manager", "Manager", TsgcGridDataType.gdtInteger).Visible = false;
            oOrg.Columns.Add("name", "Name");
            oOrg.Columns.Add("title", "Title");
            oOrg.AppendRow("1", 1, null, "Ann Smith", "CEO");
            oOrg.AppendRow("2", 2, 1, "Ben Clarke", "CTO");
            oOrg.AppendRow("3", 3, 2, "Carl Diaz", "Developer");
            oOrg.AppendRow("4", 4, 1, "Dora Evans", "CFO");

            TsgcGridLineView oOrgLines = new TsgcGridLineView(oOrg);
            TsgcGridTreeOptions oTree = new TsgcGridTreeOptions();
            oTree.KeyField = "id";
            oTree.ParentField = "manager";
            oTree.TreeField = "name";
            oTree.Expanded = true;
            oOrgLines.TreeOptions = oTree;

            TsgcGridXLSXWriter oOrgWriter = new TsgcGridXLSXWriter();
            oOrgWriter.SheetName = "Org chart";
            using (FileStream oStream = new FileStream(Path.Combine(aOutDir, "orgchart.xlsx"), FileMode.Create))
                oOrgWriter.SaveToStream(oOrg, oOrgLines, oStream);
        }
    }
}
```

.NET 8에서 10만 행을 약 1.4초 만에 기록합니다.

참고 항목

- .NET 그룹화와 요약
- .NET 트리 행과 마스터 디테일
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 데모
- Excel 내보내기

.NET 데모

데모는 `.net\demos`의 `sgcGrid_Demos.sln` 솔루션에 있습니다. 모든 데모에는 README.md와 `--test` 스위치가 있으며, 이 스위치는 화면에 표시하지 않고 자체 점검을 실행한 뒤 성공하면 0으로 종료합니다. Delphi 데모의 `/test` 스위치와 같습니다.

데모	호스트	보여 주는 내용
LiveTicker	WinForms	워커 스레드에서 1000개의 심볼을 갱신하는 데모입니다. 변경 깜박임, 상승 및 하락 색, **TailMode**를 사용합니다.
LiveTickerWpf	WPF	**TsgcWpfGrid**에서의 같은 티커입니다.
LiveTickerAvalonia	Avalonia	**TsgcAvaloniaGrid**에서의 같은 티커입니다.
DataSetGrid	WinForms	**TsgcGridSource_DataSet**을 통해 바인딩한 DataTable과 JSON 및 CSV 가져오기와 내보내기입니다.
Editing	WinForms	**OnValidate**를 사용한 제자리 편집입니다. Registered 에디션 전용입니다.
GridAndChart	WinForms	그리드의 행과 선택이 sgcDashboard 타일 안에서 **TsgcChartDataSource_Table**을 통해 sgcCharts 차트를 구동합니다.
AskYourGrid	WinForms	규칙 기반 파서를 사용하는 **TsgcGridAIQuery**입니다. Registered 에디션 전용입니다.
LiveFeed	WinForms	같은 프로세스 안의 **TsgcWebSocketServer**가 데이터를 공급하는 **TsgcGridFeedSource_WebSocket**입니다. Registered 에디션 전용입니다.
Grouping	WinForms	지역과 카테고리로 그룹화한 주문 10만 건을 그룹 패널, 그룹 및 바닥글 요약, 고정 열 두 개, 작업자 스레드의 실시간 가격 갱신과 함께 보여 줍니다.
GroupingWpf	WPF	**TsgcWpfGrid**에서 실행하는 동일한 그룹화 데모입니다.
GroupingAvalonia	Avalonia	**TsgcAvaloniaGrid**에서 실행하는 동일한 그룹화 데모입니다.
TreeAndDetail	WinForms	조직도를 트리 행으로, 그리고 DataTable의 고객과 **TsgcGridDataStore**에서 가져온 주문의 세부 정보 밴드를 표시합니다. Registered 에디션 전용입니다.
ExcelExport	Console	컨트롤 없이 **TsgcGridXLSXWriter**로 그룹화된 매출 통합 문서와 트리로 표현한 조직도를 기록합니다. Windows와 Linux에서 실행됩니다. Registered 에디션 전용입니다.

데모 빌드

데모 zip은 기본적으로 Community 에디션의 NuGet 패키지를 사용합니다. Registered 패키지로 데모를 빌드하려면 **SgcGridEdition**을 설정합니다.

```
dotnet build sgcGrid_Demos.sln -p:SgcGridEdition=Registered
```

Community 에디션에서도 Registered 전용 데모는 빌드되며, 실행하면 제한 사항을 설명합니다.

참고 항목

- .NET 개요
- NuGet에서 설치
- WinForms 빠른 시작
- .NET 실시간 피드

- .NET 그리드에 질문하기
- .NET 그룹화와 요약
- .NET 트리 행과 마스터 디테일
- .NET Excel 내보내기
- Delphi 데모

문제 해결

Undeclared identifier StartEdit, OnValidate 또는 OnEdited

Community 에디션은 SGC_GRID_EDIT 없이 컴파일되므로 그리드의 편집 관련 멤버가 존재하지 않으며, 이를 사용하는 코드는 E2003 Undeclared identifier로 실패합니다. 편집 이벤트가 있는 상태로 Registered 빌드에서 저장한 폼은 로드할 때 Error reading 예외가 발생합니다. 편집 코드를 제거하거나 Registered 또는 Trial 빌드를 사용하십시오. 에디션을 참조하십시오.

sgcGridWS_Source 또는 sgcWebSocket 파일을 찾을 수 없습니다

피드 소스와 AI 쿼리는 sgcGridWS 브리지에 있습니다. Community 에디션은 이를 배포하지 않습니다. 다른 에디션에서는 브리지에 sgcWebSockets가 필요합니다. 검색 경로에 sgcWebSockets 소스를 추가하거나, 같은 버전의 sgcGridWS 패키지보다 먼저 sgcWebSockets 패키지를 설치하십시오. 플랫폼을 참조하십시오.

Delphi 7에서 브리지가 컴파일되지 않습니다

Delphi 7에서 sgcGridWS 유닛은 sgcWebSockets에 대해 컴파일되며, sgcWebSockets는 자신이 빌드된 Indy 버전을 필요로 합니다. 브리지를 빌드하기 전에 사용 중인 sgcWebSockets 빌드가 요구하는 Indy 버전을 설치하십시오.

C++Builder DataSetGrid 데모가 Win64에서 실패합니다

Win32는 MIDAS를 정적으로 링크하지만, C++Builder에는 Win64용 정적 MIDAS 라이브러리가 없습니다. 64비트 실행 파일 옆에 midas.dll을 복사하십시오.

값이 바뀌지 않거나 행이 중복됩니다

UpdateRow는 키로 행을 찾으며 키는 대소문자를 구분합니다. AppendRow와 UpdateRow에 같은 키를 사용하십시오. AutoInsert가 False이면 없는 키에 대한 갱신은 무시됩니다. 열과 일치하지 않는 필드 이름도 무시됩니다.

OnFilterRow가 잘못된 행을 읽습니다

OnFilterRow는 저장소 행을 받습니다. 뷰 행을 받는 Cells가 아니라 StoreCells 또는 StoreKeyOf로 읽으십시오. 필터링을 참조하십시오.

피드 핸들러가 컨트롤을 건드리면 그리드가 멈춥니다

OnFilterRow와 OnGetValue는 데이터 잠금을 잡은 상태에서 실행되고, 피드의 OnChange, OnRow, OnError는 쓰기 스레드에서 실행됩니다. 그곳에서는 메인 스레드를 기다리거나 컨트롤을 건드리지 말고 TThread.Queue를 사용하십시오.

숫자에 쉼표나 점이 뜻밖에 표시됩니다

값은 로캘을 따르는 Data.FormatSettings로 서식화됩니다. 소수점 구분 기호로 점을 쓰려면 Data.UseInvariantFormat을 호출하거나 FormatSettings를 설정하십시오.

선언되지 않은 식별자 Tree, Detail 또는 SaveToXLSX

트리 행, 마스터 디테일 행, TsgcGridDataStore, Excel 내보내기는 Community 에디션에서 제외되는 SGC_GRID_ADVANCED에서만 컴파일됩니다. 이를 사용하는 코드는 E2003 Undeclared identifier로 실패하며, Tree 또는 Detail 속성이 저장된 폼은 로드할 때 Error reading 예외를 발생시킵니다. 에디션을 참조하십시오.

트리가 평평한 목록으로 표시됩니다

Tree.KeyField와 Tree.ParentField는 모두 그리드 열을 가리켜야 하며(숨겨져 있어도 됩니다) 데이터는 가상이 아니어야 합니다. 부모 값은 키 텍스트와 대소문자를 구분하여 일치시키며, 부모를 찾지 못한 행은 루트가 됩니다. 트리 행을 참조하십시오.

그룹화가 적용되지 않습니다

트리 모드가 켜져 있는 동안 그룹화는 무시됩니다. 다시 그룹화하려면 Tree.KeyField 또는 Tree.ParentField를 지우십시오.

디테일 확장기가 표시되지 않습니다

행에 확장기가 표시되려면 Detail.Active가 True이고, Detail.MasterKeyField가 그리드 열을 가리키며, Detail.Source 또는 OnGetDetailData 핸들러가 설정되어 있고, 그 행의 마스터 키가 비어 있지 않아야 합니다. 가상 데이터에서는 밴드가 표시되지 않습니다. 마스터 디테일을 참조하십시오.

디테일 밴드가 비어 있습니다

Detail.DetailKeyField는 소스의 열을 가리켜야 하며, 그 값이 마스터 키 텍스트와 일치해야 합니다(대소문자 구분, 5, 5.0, '5'는 같은 키입니다). 가상 소스는 행을 반환하지 않습니다. OnGetDetailData를 사용하고 Source가 없으면 밴드가 열을 거기서 가져오므로 Detail.Columns를 설정하십시오.

Excel 파일에 디테일 행이 없습니다

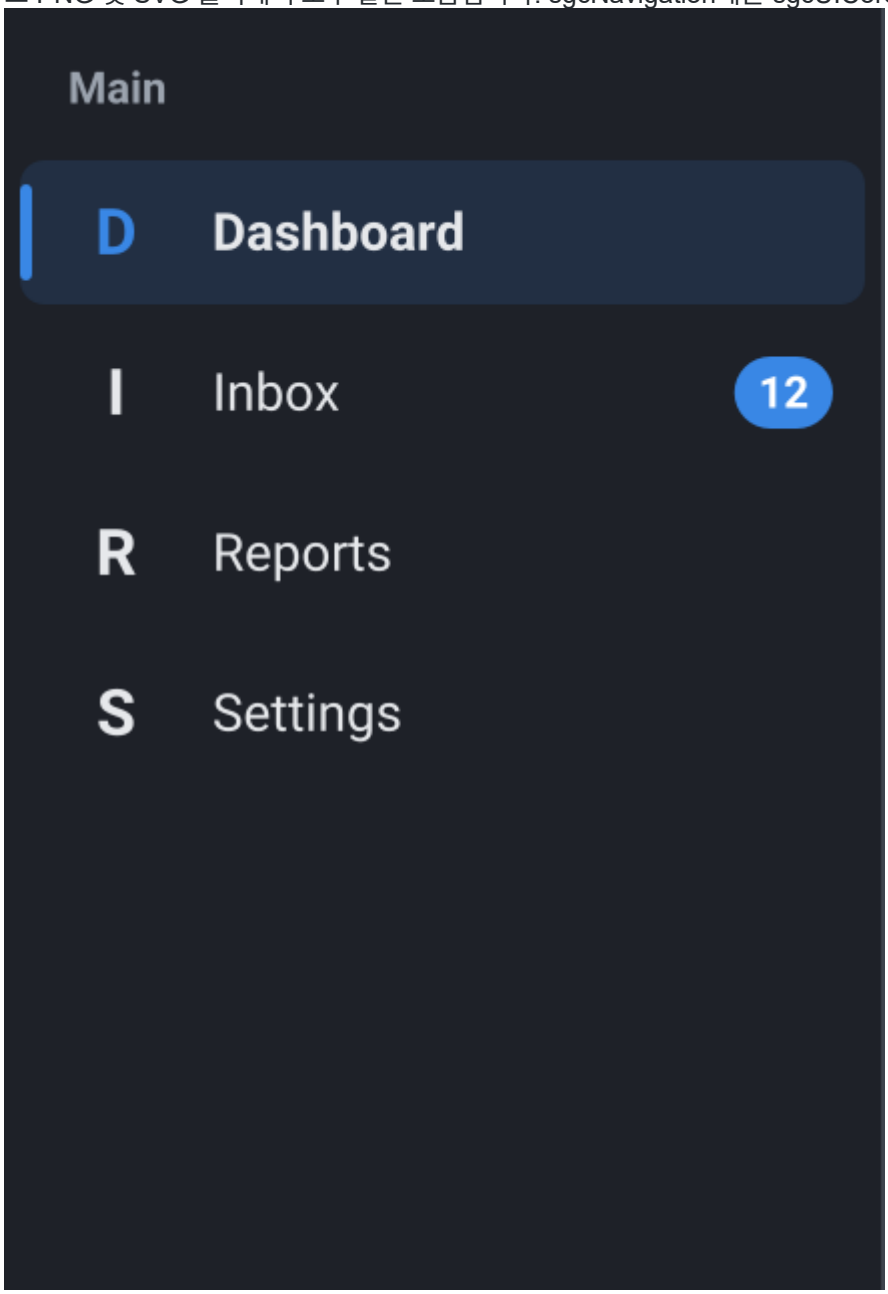
SaveToXLSX는 디테일 밴드를 쓰지 않습니다. 디테일 소스는 따로 내보내십시오. 예를 들어 TsgcGridDataStore의 Data에 대해 TsgcGridXLSXWriter를 사용합니다. Excel 내보내기를 참조하십시오.

SaveToXLSX가 EsgcGridXLSXError를 발생시킵니다

XLSX 워크시트에는 그룹 행과 헤더를 포함하여 최대 1,048,576행과 16,384열이 들어갑니다. 내보내기 전에 행을 필터링하거나 열을 숨기십시오. 예외가 발생하면 아무것도 쓰이지 않습니다.

소개

sgcNavigation 은 VCL 과 FireMonkey 용 Delphi 및 C++Builder 네 이 티 브 내 비 게 이 션 바 입 니 다. **TsgcVCLNavigationBar** 또는 **TsgcFMXNavigationBar** 컴포넌트 하나로 애플리케이션의 페이지를 아이콘, 캡션, 배지, 그룹 헤더가 있는 항목의 세로 목록으로 표시합니다. 사이드바 모드에서는 애니메이션과 함께 아이콘 레일로 접히는 평면 목록이 됩니다. 아코디언 모드에서는 그룹 헤더가 그 아래 항목을 펼치고 접습니다.
sgcNavigation은 eSeGeCe 비주얼 제품군의 일부이며 다른 제품의 틀을 이룹니다. 왼쪽에는 내비게이션 바가 있고, 오른쪽에는 **sgcDashboard** 대시보드, **sgcGrid** 그리드 또는 **sgcToolbar**가 있는 페이지가 놓입니다. 캔버스, 래스터라이저, 글꼴, SVG 라이터, 라이트 및 다크 테마를 갖춘 공유 런타임 패키지인 **sgcUICore**로 그려지므로, 바는 VCL, FireMonkey, 헤드리스 PNG 및 SVG 출력에서 모두 같은 모습입니다. sgcNavigation에는 sgcUICore가 필요하며 함께 설치됩니다.



에디션

sgcNavigation은 Registered, Trial, Community로 제공됩니다. Community 에디션은 항목의 Action 링크를 제외한 모든 기능을 갖추고 있습니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 프레임워크 중립 모델, sgcUICore 캔버스의 렌더러, VCL 및 FMX 컨트롤.
- 플랫폼: Delphi 버전, C++Builder 및 플랫폼.
- VCL 빠른 시작, FMX 빠른 시작, C++Builder 빠른 시작: 몇 분 만에 만드는 첫 내비게이션 바.
- 구성 요소: 내비게이션 바, 항목, 모델.
- 사이드바 모드, 아코디언 모드, 접기와 애니메이션, 이미지와 글리프, 배지, 키보드와 포커스, Action 링크, 테마, 헤드리스 렌더링.
- 데모: Delphi 및 C++Builder 데모 프로젝트.

아키텍처

sgcNavigation은 계층으로 구성됩니다. 각 계층은 아래 계층에만 의존하므로 모델과 렌더러는 VCL이나 FireMonkey 없이 컴파일됩니다.

계층	유닛	역할
모델	sgcNavigation_Model	TsgcNavigationModel은 Items(TsgcNavigationItem의 컬렉션), 모드, 접힘 상태, 크기, 테마, 상호작용 상태(선택, 핫, 눌림, 포커스된 항목)를 보관합니다. Layout 메서드는 표시되는 모든 항목의 행을 계산하고 HitTest는 한 지점 아래의 항목을 찾습니다.
렌더러	sgcNavigation_Render	sgcNavigationRender는 모든 sgcUICore 캔버스에 모델을 그립니다. sgcNavigationSaveToPNG와 sgcNavigationSaveToSVG는 이를 사용하여 창 없이 이미지를 씁니다. 헤드리스 렌더링을 참조하십시오.
VCL 컨트롤	sgcNavigation_VCL	TsgcVCLNavigationBar는 모델을 소유하고 그 속성을 게시하며 sgcUICore VCL 컨트롤 기반을 통해 그립니다. 마우스, 키보드, 힌트, 이미지 목록, 접기 애니메이션을 추가합니다.
FMX 컨트롤	sgcNavigation_FMX	TsgcFMXNavigationBar는 같은 속성, 메서드, 이벤트에 터치 입력을 더한 FireMonkey 컨트롤입니다.
sgcUICore	sgcUICore_Canvas, sgcUICore_VCL, sgcUICore_FMX, sgcUICore_Theme	공유 런타임: 벡터 캔버스와 래스터라이저, 프레임 페이싱(MaxFPS)이 있는 VCL 및 FMX 컨트롤 기반, 라이트 및 다크 테마 토큰.

모델과 컨트롤

내비게이션 바의 게시된 속성은 모델의 속성이며, 바의 **Model** 속성으로 나머지에 접근할 수 있습니다: ToggleGroup, HitTest, NextVisible, 상호작용 상태. 모델이 변경되면 새 프레임이 예약될 뿐 직접 그리지 않으므로, 연속된 많은 변경도 한 번의 다시 그리기로 처리됩니다.

그리기

모든 행은 렌더러가 그립니다: 이미지 또는 글리프가 있는 아이콘 슬롯, 남은 너비에 맞춘 캡션, 배지, 그리고 핫, 눌림, 선택, 포커스, 비활성 상태입니다. 모든 크기는 논리 픽셀에 모니터 배율을 곱한 값이므로 바는 어떤 DPI에서도 선명합니다. VCL 바는 PPI / 96을 배율로, FMX 바는 폼의 배율을 사용합니다.

참고 항목

- 플랫폼
- TsgcNavigationModel
- 테마

에디션

sgcNavigation은 모든 에디션이 같은 소스에서 빌드됩니다. 에디션은 빌드에 컴파일되는 기능만 바꿉니다.

에디션	포함 내용
Registered	모든 기능: VCL 및 FMX 내비게이션 바, 사이드바와 아코디언 모드, 애니메이션이 있는 접힌 아이콘 레일, 이미지, 글리프, 배지, 키보드 탐색, 테마, 항목의 Action 링크, 헤드리스 PNG 및 SVG 출력.
Trial	Registered와 같은 기능. SGC_TRIAL로 컴파일한 빌드는 빌드에 명시된 릴리스 기간이 지나면 평가판이라는 알림을 표시합니다. 그 밖에는 달라지는 것이 없습니다.
Community	항목의 Action 링크를 제외한 모든 기능.

Community에서 제외되는 기능

Community 에디션은 SGC_NAVIGATION_ACTIONS를 정의하지 않고 컴파일됩니다. Community 빌드에서는 TsgcNavigationItem의 **Action** 속성이 존재하지 않으므로 항목을 TAction에 연결할 수 없습니다. 대신 항목의 **OnClick** 또는 바의 **OnItemClick** 이벤트를 할당하십시오.
에디션은 sgcVer.inc의 컴파일 시점 선택(SGC_COMMUNITY, SGC_TRIAL도 함께 끕니다)이며 런타임 스위치가 아닙니다. Action을 사용하는 코드나 폼 파일은 Community 빌드에서 컴파일되거나 로드되지 않습니다. 문제 해결을 참조하십시오.

참고 항목

- Action 링크
- 플랫폼

설치

sgcNavigation은 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcNavigation**과 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcNavigation에 포함된 항목

sgcUICore는 레이아웃 제품과 sgcCharts의 공통 기반이며 sgcNavigation과 함께 제공됩니다. 자체 확인란은 없습니다. sgcNavigation을 선택하면 설치 프로그램이 자동으로 설치합니다. sgcNavigation은 sgcWebSockets 브리지 패키지가 없습니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드입니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Navigation`	sgcNavigation 데모입니다. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcNavigation도 포함됩니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 sgcNavigation이 포함됩니다. Community의 제품은 sgcCharts, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard이며, sgcPDF와 sgcMaps는 Community에 포함되지 않습니다. sgcNavigation의 Community 빌드에서 빠지는 기능은 에디션을 참조하십시오.

이전 독립 실행형 설치

설치 프로그램이 이전 독립 실행형 sgcUICore 설치(예: 독립 실행형 sgcCharts 설치 프로그램이 만든 것)를 발견하면, 그 설치의 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcNavigationD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcNavigation<V>` / `dclsgcNavigation<V>`	sgcNavigation 컴포넌트(런타임/디자인 타임).

의존 순서대로 빌드합니다. `sgcUICore`, 그다음 `sgcNavigation`, 그다음 `dclsgcNavigation`입니다. 그런 다음 `dclsgcNavigation` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder의 경우 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcNavigationC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

.NET

.NET에서는 달라지는 것이 없습니다. sgcNavigation은 여전히 자체 NuGet 패키지(`esegece.sgcNavigation`와 그 보조 패키지)로 배포됩니다. .NET 설치를 참조하십시오.

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE~XE8, 10~10.4, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE~XE8, 10~10.4, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows의 TsgcVCLNavigationBar, Win32 및 Win64, 모든 Delphi 버전.
FireMonkey	Delphi XE6 이상의 TsgcFMXNavigationBar(SGC_NAVIGATION_FMX), Win32, Win64, macOS, iOS, Android, 그리고 FMX for Linux(FMXLinux)를 사용하는 Linux. Images 속성에는 XE8 이상(FMX.ImgList)이, 항목 힌트에는 10.1 Berlin 이상이 필요합니다.
Linux64	프레임워크 중립 유닛, 모델, 헤드리스 PNG 및 SVG 출력이 있는 렌더러, Delphi 10.2 이상(SGC_NAVIGATION_LINUX).
.NET	WinForms, WPF, Avalonia용 동일한 API를 제공하는 C# 포트와 헤드리스 PNG 및 SVG 출력, .NET Framework 4.7.2와 .NET 8 및 9 지원. .NET 개요를 참조하십시오.

패키지

sgcNavigation에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcNavigationD13.dpk, C++Builder 13에서는 sgcNavigationC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcNavigation	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, sgcUICore
sgcNavigationVCL	런타임, VCL	sgcNavigation_VCL (TsgcVCLNavigationBar)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcNavigation
sgcNavigationFMX	런타임, FireMonkey, XE6부터	sgcNavigation_FMX (TsgcFMXNavigationBar)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcNavigation
dclsgcNavigation	디자인 타임, VCL	**SGC Navigation** 팔레트 페이지에 등록하는 컴포넌트는 TsgcVCLNavigationBar입니다.	designide, rtl, vcl, sgcNavigation, sgcNavigationVCL
dclsgcNavigationFMX	디자인 타임, FireMonkey, XE6부터	**SGC Navigation** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXNavigationBar입니다.	designide, rtl, fmx, sgcNavigation, sgcNavigationFMX, dclsgcNavigation

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcNavigationFMX에는 dclsgcNavigation가 필요하므로 dclsgcNavigation를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcNavigation	sgcNavigationVCL	sgcNavigationFMX	dclsgcNavigation, dclsgcNavigationFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcNavigation, sgcNavigationVCL	sgcNavigationFMX	dclsgcNavigation, dclsgcNavigationFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcNavigation만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcNavigation만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcNavigationC13.cbproj, sgcNavigationVCLC13.cbproj, sgcNavigationFMXC13.cbproj, dclsgcNavigationC13.cbproj, dclsgcNavigationFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXNavigationBar)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcNavigationVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcNavigation_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcNavigation는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 프레임워크에 독립적인 유닛입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.
VCL 바는 VCL 폼의 팔레트에만, FMX 바는 FireMonkey 폼의 팔레트에만 나타납니다. **Items** 속성은 표준 컬렉션 편집기로 편집합니다.

참고 항목

- 에디션
- 아키텍처
- 문제 해결

VCL 빠른 시작

디자인 타임에서

- **SGC Navigation** 팔레트 페이지에서 **TsgcVCLNavigationBar**를 폼에 놓습니다. 기본적으로 왼쪽에 정렬됩니다.
- 오브젝트 인스펙터에서 **Items**를 더블 클릭하고 항목을 추가합니다. 모든 항목에 **Caption**과 **Glyph**(문자 하나 또는 짧은 기호)를 설정하고, 그룹 헤더인 항목에는 **IsGroup**을 True로 설정합니다.
- 폼의 나머지 영역에 **Align**을 alClient로 설정한 TPanel 또는 TPageControl을 놓습니다.
- 클릭한 항목의 페이지를 표시하는 **OnItemClick** 핸들러를 작성하거나, 모든 항목의 **OnClick**을 할당합니다.
- 어두운 팔레트를 쓰려면 **Theme**을 utDark로, 아이콘 레일을 보려면 **Collapsed**를 True로 설정합니다.

코드에서

런타임에 만든 같은 바입니다. `sgcUI\Source`가 검색 경로에 있으면 이 프로그램은 그대로 컴파일됩니다.

참고 항목

- TsgcVCLNavigationBar
- TsgcNavigationItem
- 사이드바 모드
- 접기와 애니메이션
- 테마

FMX 빠른 시작

FireMonkey 컨트롤에는 Delphi XE6 이상이 필요합니다. VCL 컨트롤과 똑같이 사용합니다. 기본적으로 왼쪽에 정렬되는 **TsgcFMXNavigationBar**를 놓고, 컬렉션 편집기로 **Items**를 채우고, **OnItemClick**을 처리합니다. 행을 탭하면 클릭처럼 동작합니다.

코드에서

참고 항목

- TsgcFMXNavigationBar
- TsgcNavigationItem
- 플랫폼

C++Builder 빠른 시작

사용 중인 C++Builder 버전용 sgcNavigation 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 및 라이브러리 경로에 추가한 다음, Delphi와 똑같이 **SGC Navigation** 팔레트 페이지의 컴포넌트를 사용하십시오. 설치 프로그램이 두 경로를 IDE에 추가합니다. sgcNavigation과 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있으므로, sgcUICore에는 별도의 경로가 필요하지 않습니다.

코드에서

OnItemClick은 클로저 이벤트이므로 핸들러는 클래스의 __fastcall 멤버 함수입니다.

참고 항목

- VCL 빠른 시작
- 데모
- 플랫폼

데모

데모는 설치 폴더의 **Demos** 폴더에 있으며, Delphi 프로젝트는 Demos\Delphi 아래에, VCL 데모의 C++Builder 포트는 Demos\CBuilder 아래에 있습니다. 모든 데모 폴더에는 데모가 하는 일과 작동 방식을 설명하는 readme.html이 있습니다.

데모	구성 요소	보여 주는 내용
SidebarApp	TsgcVCLNavigationBar, TsgcVCLDashboard, TsgcVCLKPICard	아이콘, 배지, 비활성 항목, 애니메이션 접기가 있는 사이드바 모드의 내비게이션 바로, TPageControl의 페이지를 제어하며 그 중 하나는 sgcDashboard 대시보드입니다. Delphi 및 C++Builder, Win32 및 Win64. sgcDashboard가 필요합니다.
Accordion	TsgcVCLNavigationBar, TActionList	세 개의 그룹, 단일 또는 다중 펼침 스위치, TActionList의 액션에 연결된 항목이 있는 아코디언 모드의 내비게이션 바. Delphi 및 C++Builder, Win32 및 Win64. Action 링크는 Community 에디션에 포함되지 않으므로 이 데모는 Community에서 빌드되지 않습니다.
SidebarFMX	TsgcFMXNavigationBar, TsgcFMXDashboard, TsgcFMXKPICard	SidebarApp의 FireMonkey 버전: 글리프, 배지, 비활성 항목, 애니메이션 접기가 있으며 TTabControl의 탭을 제어하고 그중 하나는 sgcDashboard 대시보드입니다. Delphi, Win32 및 Win64. sgcDashboard가 필요합니다.

참고 항목

- VCL 빠른 시작
- 아코디언 모드
- Action 링크

TsgcVCLNavigationBar

유닛: sgcNavigation_VCL
TsgcVCLNavigationBar는 Windows 전용 VCL 내비게이션 바입니다. 기본적으로 왼쪽에 정렬되며 Items를 사이드바 모드 또는 아코디언 모드의 행으로 표시합니다. 마우스, 휠, 키보드, 힌트를 처리하며 행이 컨트롤보다 높으면 스크롤합니다. 모든 크기는 논리 픽셀에 PPI / 96을 곱한 값입니다.

속성

속성	형식	설명
Items	TsgcNavigationItems	표시 순서대로 나열된 항목과 그룹 헤더. TsgcNavigationItems를 참조하십시오.
Mode	TsgcNavigationMode	nbmSidebar(기본값)는 평면 목록, nbmAccordion은 펼치고 접는 그룹입니다.
Collapsed	Boolean	바를 ExpandedWidth 대신 CollapsedWidth 너비의 아이콘 레일로 표시합니다. 기본값 False.
ExpandedWidth	Integer	펼쳐진 바의 너비, 논리 픽셀. 기본값 220.
CollapsedWidth	Integer	아이콘 레일의 너비, 논리 픽셀. 기본값 56.
ItemHeight	Integer	항목 행의 높이, 논리 픽셀. 기본값 40.
GroupHeight	Integer	그룹 헤더 행의 높이, 논리 픽셀. 기본값 32.
Indent	Integer	아코디언 모드에서 그룹 항목의 들여쓰기, 논리 픽셀. 기본값 16.
AllowMultiExpand	Boolean	아코디언 모드에서 둘 이상의 그룹을 펼칠 수 있게 합니다. False(기본값)이면 그룹을 펼칠 때 다른 그룹이 접힙니다.
SelectedIndex	Integer	선택된 항목의 인덱스. 없으면 -1(기본값). 범위를 벗어난 인덱스는 -1이 됩니다.
Theme	TsgcUITheme	utLight(기본값), utDark, 또는 라이트 색상을 사용하는 utCustom.
AccentColor	TsgcUIColor	선택, 포커스, 배지의 색상($AARRGGBB). 0(기본값)이면 테마의 강조 색상을 사용합니다.
Images	TCustomImageList	항목 ImageIndex의 이미지 목록. 이미지는 자체 크기로 그려집니다.
AnimateCollapse	Boolean	런타임에 Collapsed가 바뀔 때 너비를 약 150ms 동안 애니메이션합니다. 기본값 True.
Model	TsgcNavigationModel (public, 읽기 전용)	바의 모델. TsgcNavigationModel을 참조하십시오.
ScrollY	Integer (public, 읽기 전용)	세로 스크롤 오프셋, 디바이스 픽셀.

일반적인 컨트롤 멤버도 게시됩니다. **Align**(기본값 alLeft), **Anchors**, **Constraints**, **Enabled**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **TabOrder**, **TabStop**(기본값 True), **Visible**, **OnClick**, **OnContextPopup**, **OnEnter**, **OnExit**, **OnKeyDown**, **OnKeyUp**, **OnMouseDown**, **OnMouseMove**, **OnMouseUp**, **OnResize**.

메서드

메서드	반환	설명
ItemAtPos(X, Y: Integer)	Integer	컨트롤의 한 지점에 있는 항목. 없으면 -1. 오버로드는 그 지점 아래의 부분(npItem 또는 npExpander)도 반환합니다.
ItemRect(aIndex: Integer)	TRect	스크롤이 반영된 컨트롤 좌표의 항목 행. 항목이 표시되지 않으면 빈 사각형.
ScrollIntoView(aIndex: Integer)		항목의 행이 보이도록 바를 스크롤합니다.
InitiateAction		Action이 있는 모든 항목이 스스로 갱신되게 합니다. 프레임워크가 유휴 시간에 호출합니다.

이벤트

이벤트	형식	설명
OnItemClick	TsgcNavigationItemEvent	활성화된 항목이 클릭, 탭되었거나 Enter 또는 Space로 활성화되었습니다. 항목은 이미 선택되어 있습니다. 항목의 OnClick 또는 Action은 이 이벤트 뒤에 실행됩니다.
OnSelectionChange	TNotifyEvent	사용자 또는 코드에 의해 SelectedIndex가 바뀌었습니다.
OnCollapsedChange	TNotifyEvent	Collapsed가 바뀌었습니다.

예제

참고 항목

- TsgcFMXNavigationBar
- TsgcNavigationItem
- TsgcNavigationModel
- 키보드와 포커스

TsgcFMXNavigationBar

유닛: sgcNavigation_FMX
TsgcFMXNavigationBar는 Delphi XE6 이상의 FireMonkey 내비게이션 바로, Windows, macOS, iOS, Android, 그리고 FMXLinux를 사용하는 Linux용입니다. TsgcVCLNavigationBar의 속성, 메서드, 이벤트를 갖추고 터치 입력을 추가합니다. 행을 탭하면 클릭처럼 동작합니다. 크기는 논리 단위에 폼의 배율을 곱한 값입니다. 10.1 Berlin부터는 컨트롤의 Hint가 커서 아래 항목을 따릅니다.

속성

속성	형식	설명
Items	TsgcNavigationItems	표시 순서대로 나열된 항목과 그룹 헤더. TsgcNavigationItems를 참조하십시오.
Mode	TsgcNavigationMode	nbmSidebar(기본값)는 평면 목록, nbmAccordion은 펼치고 접는 그룹입니다.
Collapsed	Boolean	바를 ExpandedWidth 대신 CollapsedWidth 너비의 아이콘 레일로 표시합니다. 기본값 False.
ExpandedWidth	Integer	펼쳐진 바의 너비, 논리 픽셀. 기본값 220.
CollapsedWidth	Integer	아이콘 레일의 너비, 논리 픽셀. 기본값 56.
ItemHeight	Integer	항목 행의 높이, 논리 픽셀. 기본값 40.
GroupHeight	Integer	그룹 헤더 행의 높이, 논리 픽셀. 기본값 32.
Indent	Integer	아코디언 모드에서 그룹 항목의 들여쓰기, 논리 픽셀. 기본값 16.
AllowMultiExpand	Boolean	아코디언 모드에서 둘 이상의 그룹을 펼칠 수 있게 합니다. False(기본값)이면 그룹을 펼칠 때 다른 그룹이 접힙니다.
SelectedIndex	Integer	선택된 항목의 인덱스. 없으면 -1(기본값). 범위를 벗어난 인덱스는 -1이 됩니다.
Theme	TsgcUITheme	utLight(기본값), utDark, 또는 라이트 색상을 사용하는 utCustom.
AccentColor	TsgcUIColor	선택, 포커스, 배지의 색상($AARRGGBB). 0(기본값)이면 테마의 강조 색상을 사용합니다.
Images	TCustomImageList	항목 ImageIndex의 이미지 목록. FMX에서는 XE8 이상이 필요하며 이미지는 16 x 16 논리 픽셀로 그려집니다.
AnimateCollapse	Boolean	런타임에 Collapsed가 바뀔 때 너비를 약 150ms 동안 애니메이션합니다. 기본값 True.
Model	TsgcNavigationModel (public, 읽기 전용)	바의 모델. TsgcNavigationModel을 참조하십시오.
ScrollY	Integer (public, 읽기 전용)	세로 스크롤 오프셋, 디바이스 픽셀.

일반적인 FMX 컨트롤 멤버도 게시됩니다. **Align**(기본값 Left), **Anchors**, **CanFocus**(기본값 True), **Enabled**, **HitTest**, **Margins**, **Padding**, **Opacity**, **PopupMenu**, **Position**, **Width**, **Height**, **TabOrder**, **Visible**, **ShowHint**, **TabStop**(10.1 이상)과 OnClick, OnEnter, OnExit, OnKey, OnMouse, OnPaint, OnResize 이벤트입니다.

메서드

메서드	반환	설명
ItemAtPos(X, Y: Single)	Integer	컨트롤의 한 지점에 있는 항목. 없으면 -1. 오버로드는 그 지점 아래의 부분(npItem 또는 npExpander)도 반환합니다.
ItemRect(aIndex: Integer)	TRectF	스크롤이 반영된 컨트롤 좌표의 항목 행. 항목이 표시되지 않으면 빈 사각형.
ScrollIntoView(aIndex: Integer)		항목의 행이 보이도록 바를 스크롤합니다.
InitiateAction		Action이 있는 모든 항목이 스스로 갱신되게 합니다. 프레임워크가 유휴 시간에 호출합니다.

이벤트

이벤트	형식	설명
OnItemClick	TsgcNavigationItemEvent	활성화된 항목이 클릭, 탭되었거나 Enter 또는 Space로 활성화되었습니다. 항목은 이미 선택되어 있습니다. 항목의 OnClick 또는 Action은 이 이벤트 뒤에 실행됩니다.
OnSelectionChange	TNotifyEvent	사용자 또는 코드에 의해 SelectedIndex가 바뀌었습니다.
OnCollapsedChange	TNotifyEvent	Collapsed가 바뀌었습니다.

예제

참고 항목

- TsgcVCLNavigationBar
- TsgcNavigationItem
- 플랫폼

TsgcNavigationItem

유닛: sgcNavigation_Model
TsgcNavigationItem은 내비게이션 바의 한 행으로, 애플리케이션의 한 페이지이거나 IsGroup이 True일 때는 그룹 헤더입니다. 항목은 바의 Items 컬렉션에 들어 있습니다.

속성

속성	형식	설명
Caption	string	행의 텍스트. 접힌 아이콘 레일에서는 숨겨집니다.
Glyph	string	짧은 텍스트 또는 기호(예: 문자 하나)로, 항목에 이미지가 없을 때 아이콘 슬롯 가운데에 그려집니다.
ImageIndex	Integer	바의 Images에 있는 이미지의 인덱스. 기본값 -1, 이미지 없음.
Badge	string	행 오른쪽에 강조 색상의 알약 모양으로, 레일에서는 아이콘 위의 점으로 그려지는 짧은 텍스트. 비어 있으면(기본값) 배지를 그리지 않습니다. 배지를 참조하십시오.
IsGroup	Boolean	항목을 그룹 헤더로 만듭니다. 헤더는 선택되지 않습니다. 아코디언 모드에서는 다음 헤더까지 그 아래 항목을 펼치고 접습니다. 기본값 False.
Expanded	Boolean	아코디언 모드에서 그룹 헤더의 항목을 표시할지 여부. 기본값 True.
Enabled	Boolean	비활성 항목은 흐리게 그려지며 선택하거나 클릭할 수 없습니다. 기본값 True.
Visible	Boolean	숨겨진 항목은 행을 갖지 않습니다. 기본값 True.
Hint	string	행의 힌트. 접힌 레일에서 Hint가 없는 항목은 Caption을 힌트로 표시합니다.
Tag	NativeInt	애플리케이션이 자유롭게 쓰는 값이며, XE2 이전에는 Integer입니다.
Action	TBasicAction	연결된 액션. Caption, Enabled, Visible, Hint, ImageIndex는 액션을 따르며, OnClick이 할당되지 않은 경우 클릭하면 액션이 실행됩니다. Community 에디션에는 없습니다. Action 링크를 참조하십시오.
BoundsRect	TsgcUIRect (public, 읽기 전용)	마지막 레이아웃의 행으로, 디바이스 픽셀 단위이며 스크롤 이전의 콘텐츠 좌표입니다.
Shown	Boolean (public, 읽기 전용)	마지막 레이아웃에서 항목이 행을 받았는지 여부.
Separator	Boolean (public, 읽기 전용)	행이 레일에서 그룹의 구분선인지 여부.
Level	Integer (public, 읽기 전용)	행의 들여쓰기 수준. 아코디언 그룹의 항목은 1입니다.

메서드

메서드	반환	설명
Click(aComponent: TComponent = nil)		항목을 실행합니다. OnClick이 할당되어 있으면 OnClick을, 아니면 Action을 aComponent를 ActionComponent로 하여 실행합니다. 바는 OnItemClick 뒤에 이를 호출합니다.
InitiateAction		연결된 Action이 스스로 갱신되게 합니다. Action이 없으면 아무 일도 하지 않습니다.

이벤트

이벤트	형식	설명
OnClick	TNotifyEvent	항목이 활성화되었습니다. Sender는 항목입니다.

예제

참고 항목

- TsgcNavigationItems
- 이미지와 글리프
- 배지
- Action 링크

TsgcNavigationItems

유닛: sgcNavigation_Model
TsgcNavigationItems는 내비게이션 바의 TsgcNavigationItem 컬렉션이며 TOwnedCollection이므로, 항목은 디자인 타임에 표준 컬렉션 편집기로 편집하고 폼 파일에 저장됩니다. 컬렉션의 순서가 행의 순서입니다. 그룹 헤더는 다음 헤더까지 뒤따르는 항목을 소유합니다.

메서드

메서드	반환	설명
Add	TsgcNavigationItem	끝에 빈 항목을 추가합니다.
AddItem(const aCaption: string; const aGlyph: string = ''; aIsGroup: Boolean = False)	TsgcNavigationItem	Caption과 Glyph가 있는 항목을 추가하고 (aIsGroup이 True이면 그룹 헤더) 그것을 반환합니다.
Items[Index: Integer]	TsgcNavigationItem	인덱스 위치의 항목. 기본 속성이므로 Nav.Items[2]는 세 번째 항목입니다.

상속된 TCollection 멤버도 사용할 수 있습니다: **Count**, **Clear**, **Delete**, **Insert**, **BeginUpdate**, **EndUpdate**. 많은 변경을 BeginUpdate와 EndUpdate로 감싸면 바가 한 번만 레이아웃됩니다.

예제

참고 항목

- TsgcNavigationItem
- 아코디언 모드

TsgcNavigationModel

유닛: sgcNavigation_Model
TsgcNavigationModel은 내비게이션 바의 프레임워크 중립 모델입니다. 모든 바는 하나를 소유하며 **Model** 속성으로 게시합니다. 컨트롤 없이 모델만 따로 만들어 바를 이미지로 렌더링하거나 레이아웃을 테스트할 수도 있습니다. VCL이나 FMX에 의존하지 않으며 모든 플랫폼에서 빌드됩니다. 메인 스레드에서 사용하도록 설계되었습니다.

속성

게시된 속성은 바의 속성과 같습니다: Items, Mode, Collapsed, ExpandedWidth, CollapsedWidth, ItemHeight, GroupHeight, Indent, AllowMultiExpand, SelectedIndex, Theme, AccentColor. TsgcVCLNavigationBar를 참조하십시오. 모델은 다음 public 속성을 추가합니다.

속성	형식	설명
HotIndex	Integer	마우스 아래의 항목. 없으면 -1.
PressedIndex	Integer	눌리고 있는 항목. 없으면 -1.
FocusedIndex	Integer	키보드 포커스가 있는 항목. 없으면 -1.
ShowFocus	Boolean	포커스 사각형을 그립니다. 컨트롤은 키보드를 사용하는 동안 이를 설정합니다.
LayoutWidth	Integer (읽기 전용)	마지막 Layout의 너비, 디바이스 픽셀.
LayoutScale	Double (읽기 전용)	마지막 Layout의 배율.

메서드

메서드	반환	설명
Layout(aWidth: Integer; aScale: Double = 1)		너비가 aWidth 디바이스 픽셀인 바의 행을 계산합니다. aScale은 모든 논리 크기에 곱해집니다.
ContentHeight	Double	마지막 Layout의 행 높이와 패딩의 합, 디바이스 픽셀.
HitTest(aX, aY: Double; out aPart: TsgcNavigationPart)	Integer	마지막 Layout의 콘텐츠 좌표에서 한 지점에 있는 항목. 없으면 -1. aPart는 npItem이거나, 아코디언 그룹의 확장 표시 위에서는 npExpander입니다. aPart가 없는 오버로드도 있습니다. 비활성 항목은 보고되고 레일 구분선은 보고되지 않습니다.
IsItemShown(aIndex: Integer)	Boolean	현재 Mode, Collapsed, 그룹 상태에서 항목이 행을 받는지 여부.
IsItemFocusable(aIndex: Integer)	Boolean	키보드가 항목에 멈출 수 있는지 여부. 표시되고 활성화되어 있으며 그룹 헤더가 아니어야 하고, 펼쳐진 아코디언은 예외입니다.
NextVisible(aIndex: Integer)	Integer	다음 포커스 가능한 항목. 없으면 -1. NextVisible(-1)은 첫 번째 항목입니다.
PrevVisible(aIndex: Integer)	Integer	이전 포커스 가능한 항목. 없으면 -1. PrevVisible(Items.Count)는 마지막 항목입니다.
GroupOf(aIndex: Integer)	Integer	항목이 속한 그룹 헤더. 항목이 그룹이거나 앞에 그룹이 없으면 -1.

ToggleGroup(aIndex: Integer)		그룹 헤더를 펼치거나 접습니다. AllowMultiExpand가 False이면 펼칠 때 다른 그룹이 접힙니다.
CurrentWidth	Integer	현재 Collapsed 상태의 논리 너비, ExpandedWidth 또는 CollapsedWidth.
GetAccent	TsgcUIColor	사용 중인 강조 색상. AccentColor이거나, 0이면 테마의 강조 색상입니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	모델이 변경되어 다시 그려야 합니다. 바가 이를 할당하므로 따로 만든 모델에서만 사용하십시오.

예제

참고 항목

- 헤드리스 렌더링
- 아키텍처
- TsgcVCLNavigationBar

사이드바 모드

사이드바 모드(**Mode** = nbmSidebar)가 기본값입니다. 바는 보이는 모든 항목을 위에서부터 쌓인 행으로 표시합니다. 그룹 헤더는 아래 항목의 제목을 붙이는 흐린 색의 정적 레이블이며, 선택되지 않고 클릭해도 아무 일도 일어나지 않습니다.

- 항목 행의 높이는 ItemHeight 픽셀이고 그룹 헤더 행의 높이는 GroupHeight 픽셀입니다.
- 활성화된 항목을 클릭하거나 탭하면 선택되고, OnItemClick이 발생한 다음 항목의 OnClick 또는 Action이 실행됩니다.
- 선택된 행에는 강조 색조 배경과 왼쪽 가장자리의 3픽셀 강조 막대가 표시됩니다.
- 행이 컨트롤보다 높으면 마우스 휠로 바를 스크롤하며, ScrollIntoView는 지정한 항목을 보여 줍니다.
- Collapsed는 사이드바를 아이콘 레일로 바꿉니다. 접기와 애니메이션을 참조하십시오.

예제

참고 항목

- 아코디언 모드
- 접기와 애니메이션
- TsgcVCLNavigationBar

아코디언 모드

아코디언 모드(**Mode** = nbmAccordion)에서는 그룹 헤더가 다음 헤더까지 뒤따르는 항목을 펼치고 접습니다. 그룹의 항목은 그룹이 보이고 **Expanded** 속성이 True인 동안에만 표시되며 **Indent**만큼 들여쓰기됩니다. 헤더 오른쪽 끝에는 갈매기 모양 확장 표시가 있습니다.

- 헤더 행이나 확장 표시를 클릭하면 그룹이 토글됩니다. 키보드에서는 Enter 또는 Space가 같은 동작을 하며, Left는 포커스된 그룹을 접고 Right는 펼칩니다.
- **AllowMultiExpand**가 False(기본값)이면 그룹을 펼칠 때 다른 그룹이 접히므로 한 번에 하나의 그룹만 열려 있습니다. 여러 그룹을 열어 두려면 True로 설정하십시오.
- 코드에서 **Model.ToggleGroup**은 같은 규칙으로 그룹을 토글하고, 헤더의 **Expanded**를 설정하면 다른 그룹에 영향을 주지 않고 열거나 닫습니다.
- 첫 번째 그룹 헤더 앞의 항목은 어떤 그룹에도 속하지 않으며 항상 표시됩니다.
- 접힌 레일에서는 그룹이 토글되지 않습니다. 헤더가 아닌 모든 보이는 항목이 표시되고 각 그룹은 구분선이 됩니다.

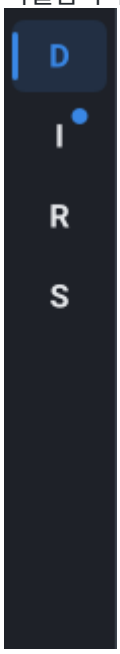
예제

참고 항목

- 사이드바 모드
- TsgcNavigationItems
- 데모

접기와 애니메이션

Collapsed는 바의 너비를 **ExpandedWidth**(기본값 220)와 **CollapsedWidth**(기본값 56) 사이에서 전환하며, 둘 다 논리 픽셀입니다. 접힌 바는 사이드바 모드와 아코디언 모드 모두에서 아이콘 레일입니다.



- 모든 항목의 아이콘 슬롯만 그려집니다. 캡션은 숨겨지며, Hint가 없는 항목의 힌트는 Caption입니다.
- 그룹 헤더는 구분선이 됩니다. 단, 첫 번째 행이 되거나 다른 구분선 뒤에 오는 경우는 제외됩니다.
- 배지는 아이콘 위의 점이 됩니다.
- **AnimateCollapse**가 True(기본값)이면 런타임에 너비가 약 150ms 동안 애니메이션됩니다. 디자인 타임이나 AnimateCollapse가 False이면 즉시 바뀝니다.
- **OnCollapsedChange**는 Collapsed가 바뀔 때 발생합니다.

바는 자신의 Width를 설정하므로 Align은 alLeft(VCL) 또는 Left(FMX)로 두고 오른쪽의 콘텐츠는 alClient를 사용하게 하십시오. 모델은 **CurrentWidth**로 너비를 보고합니다.

예제

참고 항목

- 사이드바 모드
- 배지
- TsgcVCLNavigationBar

이미지와 글리프

모든 행의 왼쪽에는 아이콘 슬롯이 있습니다. 슬롯에는 다음 우선순위로 표시됩니다.
- 둘 다 설정된 경우, 바의 **Images** 이미지 목록에 있는 **ImageIndex** 이미지,
- 그렇지 않으면 항목의 **Glyph** 텍스트(예: 문자 하나 또는 짧은 기호)를 가운데 정렬로,
- 그렇지 않으면 아무것도 표시하지 않습니다.

글리프는 이미지 목록이 필요 없고 테마 색상을 따르므로 데모에서는 문자를 사용합니다. 비활성 항목은 이미지를 불투명도 40%로 그립니다.

VCL

Images는 TCustomImageList이며 이미지는 자체 크기로 그려집니다. 마스크 이미지 목록과 32비트 이미지 목록 모두 동작합니다. 모든 이미지는 한 번 ARGB로 변환되어 이미지 목록이 바뀔 때까지 캐시됩니다.

FireMonkey

Images는 FMX.ImgList의 FMX TCustomImageList이며 Delphi XE8 이상이 필요합니다. 이미지는 16 x 16 논리 픽셀로 그려지고 프레임의 디바이스 크기에 맞춰 이미지 목록에서 가져오며, 이미지 목록이 바뀔 때까지 캐시됩니다.

예제

참고 항목

- TsgcNavigationItem
- 플랫폼

배지

Badge는 항목의 짧은 텍스트로, 예를 들어 읽지 않은 개수나 New라는 단어입니다. 행의 오른쪽에 정렬된 강조 색상의 알약 모양으로 그려지며, 텍스트는 강조 색상 위에서 더 잘 읽히는 검은색 또는 흰색입니다. 비어 있는 Badge는 아무것도 그리지 않습니다. 접힌 아이콘 레일에서는 배지가 아이콘 위의 점이 됩니다.
값이 바뀔 때마다 코드에서 Badge를 설정하십시오. 바는 스스로 다시 그립니다.

예제

참고 항목

- TsgcNavigationItem
- 테마
- 접기와 애니메이션

키보드와 포커스

바는 키보드 포커스를 받으며(VCL의 **TabStop**, FMX의 **CanFocus**, 둘 다 기본값 True) 포커스된 행을 포커스 가능한 항목 사이에서 이동합니다. 포커스 가능한 항목은 표시되고 활성화된 항목이며, 펼쳐진 아코디언에서는 그룹 헤더도 포함됩니다.

키	동작
Up, Down	포커스를 이전 또는 다음 포커스 가능한 행으로 옮기고 보이도록 스크롤합니다.
Home, End	포커스를 첫 번째 또는 마지막 포커스 가능한 행으로 옮깁니다.
Enter, Space	포커스된 행을 클릭처럼 활성화합니다. 항목을 선택하고 OnItemClick을 발생시키거나 아코디언 그룹을 토글합니다.
Left, Right	포커스된 아코디언 그룹을 접거나 펼칩니다. 그룹의 항목에서 Left를 누르면 포커스가 그룹 헤더로 이동합니다.

점선 포커스 사각형은 키보드를 사용하는 동안에만 표시됩니다. 선택과 포커스는 별개입니다. **SelectedIndex**는 사용자가 선택한 페이지이고 **Model.FocusedIndex**는 키보드가 있는 행입니다.

힌트

ShowHint가 True이면 힌트는 커서 아래 항목의 **Hint**이거나, 접힌 레일에서는 그 Caption이므로 다른 행으로 이동하면 그 행의 힌트가 표시됩니다. FMX에서 힌트는 10.1 Berlin 이상이 필요합니다.

예제

참고 항목

- TsgcVCLNavigationBar
- TsgcNavigationModel
- 아코디언 모드

Action 링크

항목은 **Action** 속성을 통해 모든 TBasicAction, 즉 VCL 또는 FireMonkey TAction에 연결됩니다. Action 링크는 Community 에디션에 포함되지 않습니다. 에디션을 참조하십시오.

- 항목의 **Caption**, **Enabled**, **Visible**, **Hint**, **ImageIndex**는 액션을 따릅니다.
- 항목을 클릭하면 바의 OnItemClick이 발생한 다음 바를 ActionComponent로 하여 액션이 실행됩니다. 항목에 OnClick 핸들러가 있으면 액션 대신 OnClick이 실행됩니다.
- 바가 모든 항목에 InitiateAction을 호출하므로 액션의 OnUpdate는 유휴 시간에 실행됩니다.
- Delphi XE3 이상에서는 액션이 모든 변경을 항목에 전달합니다. 이전 버전에서는 항목이 액션을 할당할 때와 InitiateAction마다 액션의 속성을 복사합니다.

예제

참고 항목

- TsgcNavigationItem
- 데모
- 에디션

테마

Theme은 sgcUICore 테마 토큰으로 바의 색상을 설정합니다. utLight(기본값), utDark, 또는 라이트 색상을 사용하는 utCustom입니다. 표면, 오른쪽 가장자리의 테두리, 텍스트, 흐린 그룹 헤더, 호버 및 눌림 오버레이, 강조 색상은 모두 테마에서 가져옵니다.
AccentColor는 테마의 강조 색상을 $AARRGGBB 색상으로 재정의하며, 선택 막대와 배경, 포커스 사각형, 배지에 사용됩니다. 0(기본값)이면 테마의 강조 색상을 유지합니다.
테마 토큰은 sgcDashboard 및 다른 sgcUICore 제품과 같으므로 Theme이 같은 바와 대시보드는 서로 어울립니다. sgcDashboard 타일에 호스팅된 바는 대시보드의 테마를 따릅니다. 타일이 호스팅하는 컨트롤의 게시된 Theme에 자신의 테마를 전달하기 때문입니다.

예제

참고 항목

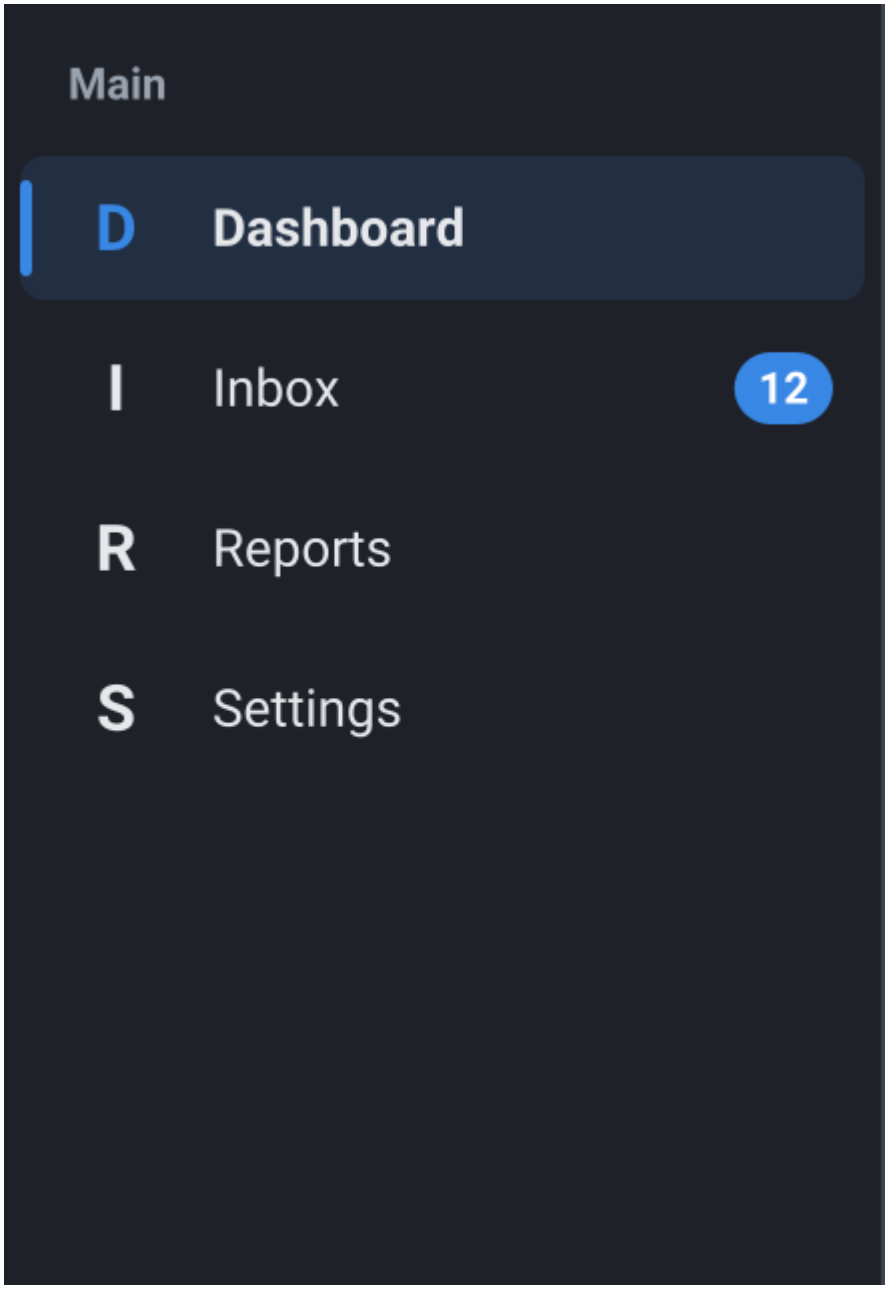
- 아키텍처
- 배지
- 헤드리스 렌더링

헤드리스 렌더링

sgcNavigation_Render의 렌더러는 창이나 VCL, FMX 없이 TsgcNavigationModel을 그리므로, 콘솔 애플리케이션, 서비스, 테스트, 웹 서버가 같은 바를 이미지로 만들 수 있습니다. 예를 들어 애플리케이션 문서나 비주얼 테스트에 쓸 수 있습니다. 이 유닛은 Linux64를 포함한 모든 플랫폼에서 빌드됩니다.

루틴	설명
sgcNavigationSaveToPNG(aModel, aFileName or aStream, aWidth, aHeight, aScale = 1, aImages = nil)	PNG 파일 또는 스트림을 씁니다. aWidth와 aHeight는 논리 픽셀이며 aScale이 이를 곱합니다. 배율 2에서 220 x 320은 440 x 640 이미지가 됩니다.
sgcNavigationSaveToSVG(aModel, aFileName or aStream, aWidth, aHeight, aScale = 1, aImages = nil)	같은 바의 SVG 파일 또는 스트림을 씁니다.
sgcNavigationRender(aModel, aCanvas, aWidth, aHeight, aScale = 1, aScrollY = 0, aImages = nil)	디바이스 픽셀 단위로 모든 sgcUICore 캔버스에 바를 그립니다.

aImages는 ImageIndex의 이미지를 반환하는 TsgcNavigationImageEvent 형식의 선택적 콜백입니다. 이것이 없으면 항목은 Glyph를 그립니다. 컨트롤의 모델은 **Model** 속성으로 얻을 수 있으므로 **sgcNavigationSaveToPNG(Nav.Model, ...)**은 컨트롤이 표시하는 내용을 임의의 크기로 저장합니다.



예제

참고 항목

- TsgcNavigationModel
- 플랫폼

.NET 개요

sgcNavigation .NET은 Delphi sgcNavigation 라이브러리를 관리 코드 C#으로 이식한 것이며, Delphi 라이브러리를 감싸는 래퍼가 아닙니다. C# 클래스는 **esegece.sgcNavigation** 네임스페이스에 있고 Delphi 이름을 그대로 사용하므로, **TsgcNavigationModel**, **TsgcNavigationItems**, **TsgcNavigationItem**, **TsgcNavigationMode**은 두 언어에서 똑같이 읽히며, 각 호스트의 내비게이션 바도 Delphi 명명 패턴을 따릅니다. 렌더링에는 **sgcUICore**의 .NET 포트를 사용하며, eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 라이트 및 다크 테마를 공유합니다. 따라서 내비게이션 바는 WinForms, WPF, Avalonia, 헤드리스 PNG 및 SVG 출력에서 모두 같게 보입니다.
내비게이션 바는 아이콘, 캡션, 선택적 배지가 있는 항목 목록을 표시합니다. 사이드바 모드에서는 모든 항목이 표시되고 그룹 헤더는 고정 레이블입니다. 아코디언 모드에서는 그룹 헤더를 클릭하면 해당 항목이 펼쳐지거나 접힙니다. 두 모드 모두 바가 아이콘 레일로 접히며, ExpandedWidth와 CollapsedWidth 사이에서 애니메이션이 적용됩니다.

패키지

이 포트는 다섯 개의 NuGet 패키지로 나뉘어 있으므로, 프로젝트는 사용하는 호스트만 가져옵니다.

PackageId	포함 내용	대상 프레임워크
esegece.sgcNavigation	모델(**TsgcNavigationModel**, **TsgcNavigationItems**, **TsgcNavigationItem**), 항목의 **Command** 링크, 렌더러 **sgcNavigation_Render**, 헤드리스 SVG 출력. 네이티브 종속성이 없습니다. esegece.sgcUICore에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcNavigation.Rendering	헤드리스 PNG 출력 **sgcNavigation_Rendering**. Windows, Linux, macOS에서 esegece.sgcUICore.Rendering의 SkiaSharp 캔버스에 그립니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcNavigation.WinForms	**TsgcWinFormsNavigationBar**.	net472, net8.0-windows, net9.0-windows
esegece.sgcNavigation.Wpf	**TsgcWpfNavigationBar**.	net472, net8.0-windows, net9.0-windows
esegece.sgcNavigation.Avalonia	**TsgcAvaloniaNavigationBar**. Avalonia 11을 사용하며 Windows, Linux, macOS에서 동작합니다.	net8.0, net9.0

세 개의 호스트 패키지는 esegece.sgcNavigation.Rendering에 종속되고, 이 패키지는 esegece.sgcNavigation에 종속됩니다. 따라서 `PackageReference` 한 줄로 호스트에 필요한 모든 것을 가져올 수 있습니다. NuGet으로 설치를 참조하십시오.

에디션

에디션	패키지 ID	포함 내용
Registered	esegece.sgcNavigation, esegece.sgcNavigation.Rendering, esegece.sgcNavigation.WinForms, esegece.sgcNavigation.Wpf, esegece.sgcNavigation.Avalonia	모든 기능: 세 호스트의 내비게이션 바, 사이드바 및 아코디언 모드, 아이콘 레일, 이미지, 배지, 키보드 탐색, 항목의 Command 링크, 테마, 헤드리스 PNG 및 SVG 출력.

Community	같은 ID에 .Community 접미사를 붙인 것입니다. 예: esegece.sgcNavigation.WinForms.Community. sgcUICore의 Community 패키지에 종속됩니다.	Registered와 같은 내용에서 항목의 **Command** 및 **CommandParameter** 속성을 뺀 것입니다. Delphi Community 에디션에 액션 링크가 없기 때문입니다. 항목은 OnClick으로 동작합니다.

Community 에디션은 연간 총매출이 1,000,000 EUR 미만이고 개발자가 5명 이하인 개인 및 조직이 상업적 및 비상업적 용도로 무료로 사용할 수 있습니다. 두 기준 중 하나라도 같거나 넘는 조직은 Registered 라이선스가 필요합니다. .NET 포트에는 Trial 에디션이 없습니다.

호스트

호스트	Delphi 대응	플랫폼
WinForms	VCL (TsgcVCLNavigationBar)	Windows, .NET Framework 4.7.2 및 .NET 8, 9.
WPF	FMX (TsgcFMXNavigationBar)	Windows, .NET Framework 4.7.2 및 .NET 8, 9. 바는 XAML에서 선언하고 항목은 코드에서 추가합니다.
Avalonia	FMX (TsgcFMXNavigationBar)	Windows, Linux, macOS, .NET 8 및 9, 마우스, 터치, 펜 지원. 바는 XAML에서 선언하고 항목은 코드에서 추가합니다.
Headless	sgcNavigation_Render	대상 프레임워크의 모든 플랫폼: SVG는 esegece.sgcNavigation, PNG는 esegece.sgcNavigation.Rendering.

모든 호스트의 API는 같습니다. Items, Mode, Collapsed, ExpandedWidth, CollapsedWidth, ItemHeight, GroupHeight, Indent, AllowMultiExpand, SelectedIndex, Theme, AccentColor, Images, AnimateCollapse, ScrollY, ItemAtPos, ItemRect, ScrollIntoView, 그리고 이벤트 OnItemClick, OnSelectionChange, OnCollapsedChange가 있습니다. 클래스 이름은 .NET 클래스를, Delphi에서 달라지는 점은 Delphi와 .NET의 차이점을 참조하십시오.

참고 항목

- NuGet으로 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET 데모
- 플랫폼

NuGet으로 설치

패키지는 nuget.org에 있습니다. 호스트 패키지를 추가하면 코어 패키지와 렌더링 패키지가 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcNavigation.WinForms
dotnet add package esegece.sgcNavigation.Wpf
dotnet add package esegece.sgcNavigation.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcNavigation.WinForms.Community
dotnet add package esegece.sgcNavigation.Wpf.Community
dotnet add package esegece.sgcNavigation.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcNavigation.Wpf" Version="2026.10.0" />
</ItemGroup>
```

한 프로젝트에서는 Registered 패키지나 Community 패키지 중 하나만 참조하고, 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있기 때문입니다. 항목의 **Command**를 사용하는 코드는 Community 패키지에 대해 컴파일되지 않습니다. 한 프로젝트에서 두 에디션을 모두 빌드하는 경우 `#if !SGC_COMMUNITY`로 감싸십시오.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서는 `xmlns:sgc="clr-namespace:esegece.sgcNavigation;assembly=esegece.sgcNavigation.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: Avalonia 11.3의 **Avalonia.Desktop** 및 **Avalonia.Themes.Fluent** 패키지를 사용하는 `net8.0` 또는 `net9.0` 프로젝트. 일반 Avalonia 데스크톱 애플리케이션과 같습니다. XAML에서는 `xmlns:sgc="using:esegece.sgcNavigation"`으로 네임스페이스를 매핑합니다.
- Headless: SVG에는 esegece.sgcNavigation만으로 충분합니다. PNG에는 esegece.sgcNavigation.Rendering을 추가하십시오. SkiaSharp와 Windows 및 Linux용 네이티브 자산이 함께 설치됩니다.

C#에서는 모든 호스트에 `using esegece.sgcNavigation;`이 필요하며, **TsgcUITheme**과 그 밖의 테마 형식에는 `using esegece.sgcUICore;`도 필요합니다.

참고 항목

- .NET 개요
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 데모

WinForms 빠른 시작

TsgcWinFormsNavigationBar는 WinForms 내비게이션 바이며, Delphi의 TsgcVCLNavigationBar에 대응합니다. **TsgcNavigationModel**을 소유하고, 그 속성을 노출하며, sgcUICore로 그립니다. 패키지는 **esegece.sgcNavigation.WinForms**입니다.

단계

- WinForms 프로젝트를 만들고 esegece.sgcNavigation.WinForms 패키지를 추가합니다.
- **TsgcWinFormsNavigationBar**를 만듭니다. 기본적으로 왼쪽에 도킹되며 너비는 **ExpandedWidth**를 따릅니다.
- **ImageList**를 **Images**에 할당하고 `Items.AddItem(aCaption)`으로 항목을 추가합니다. 각 항목의 **ImageIndex**, **Badge**, **Hint**를 설정합니다. `AddItem(aCaption, "", true)`는 그룹 헤더를 추가합니다.
- 선택한 항목의 페이지를 표시하려면 **OnSelectionChange**를, 모든 클릭에 반응하려면 **OnItemClick**을 할당합니다.
- 실행합니다. 클릭 또는 Enter로 항목을 선택하고, 화살표 키, Home, End로 포커스를 이동하며, **Collapsed**는 바를 아이콘 레일로 접습니다.

예제

사이드바, 배지, 그룹 헤더, Collapse 버튼이 있는 WinForms 애플리케이션의 전체 Program.cs입니다. esegece.sgcNavigation.WinForms를 참조하는 `net8.0-windows` 프로젝트에서 그대로 컴파일됩니다.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcNavigation;

namespace NavigationQuickStartWinForms
{
    internal class MainForm : Form
    {
        private readonly TsgcWinFormsNavigationBar Nav;
        private readonly Button btnCollapse;
        private readonly Label lblPage;

        public MainForm()
        {
            Text = "sgcNavigation WinForms";
            ClientSize = new Size(720, 460);

            lblPage = new Label { Dock = DockStyle.Fill, TextAlign = ContentAlignment.MiddleCenter };
            btnCollapse = new Button { Text = "Collapse", Dock = DockStyle.Top };
            btnCollapse.Click += btnCollapse_Click;

            // a sidebar on the left: the width follows ExpandedWidth and CollapsedWidth
            Nav = new TsgcWinFormsNavigationBar();
            Nav.Dock = DockStyle.Left;
            Nav.Mode = TsgcNavigationMode.nbmSidebar;
            Nav.AnimateCollapse = true;
            Nav.Images = DoBuildImages();
            DoAddItem("Dashboard", 0, "");
            DoAddItem("Inbox", 1, "12");
            DoAddItem("Reports", 2, "");
            // a group header is a static label in sidebar mode
            Nav.Items.AddItem("Admin", "", true);
            DoAddItem("Settings", 3, "").Hint = "Application settings";
            Nav.OnSelectionChange = NavSelectionChange;
            Nav.OnItemClick = NavItemClick;
            Nav.OnCollapsedChange = NavCollapsedChange;
            Nav.SelectedIndex = 0;
```

```
            // the last control added is docked first, so the bar takes the whole left side
            Controls.Add(lblPage);
            Controls.Add(btnCollapse);
            Controls.Add(Nav);
        }

        private TsgcNavigationItem DoAddItem(string aCaption, int aImageIndex, string aBadge)
        {
            TsgcNavigationItem oItem = Nav.Items.AddItem(aCaption);
            oItem.ImageIndex = aImageIndex;
            oItem.Badge = aBadge;
            return oItem;
        }

        private static ImageList DoBuildImages()
        {
            // four 16 x 16 icons, drawn at startup
            ImageList oImages = new ImageList();
            oImages.ColorDepth = ColorDepth.Depth32Bit;
            oImages.ImageSize = new Size(16, 16);
            Color[] vColors = { Color.SteelBlue, Color.SeaGreen, Color.DarkOrange, Color.SlateGray };
            foreach (Color vColor in vColors)
            {
                Bitmap oBitmap = new Bitmap(16, 16);
                using (Graphics oGraphics = Graphics.FromImage(oBitmap))
                using (SolidBrush oBrush = new SolidBrush(vColor))
                    oGraphics.FillEllipse(oBrush, 2, 2, 12, 12);
                oImages.Images.Add(oBitmap);
            }
            return oImages;
        }

        private void NavSelectionChange(object Sender)
        {
            // SelectedIndex is -1 when no item is selected
            if (Nav.SelectedIndex >= 0)
                lblPage.Text = Nav.Items[Nav.SelectedIndex].Caption + " page";
        }

        private void NavItemClick(object Sender, TsgcNavigationItem aItem)
        {
            Text = "sgcNavigation WinForms: " + aItem.Caption;
        }

        private void NavCollapsedChange(object Sender)
        {
            btnCollapse.Text = Nav.Collapsed ? "Expand" : "Collapse";
        }

        private void btnCollapse_Click(object sender, EventArgs e)
        {
            Nav.Collapsed = !Nav.Collapsed;
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new MainForm());
        }
    }
}
```

바는 도구 상자에도 있습니다. 항목은 디자이너가 직렬화하지 않으므로 항상 코드에서 추가합니다. Delphi와 .NET의 차이점을 참조하십시오.

참고 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- WPF 빠른 시작
- Avalonia 빠른 시작
- 이미지와 글리프

WPF 빠른 시작

TsgcWpfNavigationBar는 WPF 내비게이션 바입니다. 다른 컨트롤처럼 XAML에서 선언하고, 항목은 코드에서 추가합니다. 패키지는 **esegece.sgcNavigation.Wpf**입니다.

단계

- WPF 프로젝트(`dotnet new wpf`)를 만들고 esegece.sgcNavigation.Wpf 패키지를 추가합니다.
- `esegece.sgcNavigation` 네임스페이스를 매핑하고 **TsgcWpfNavigationBar**를 선언합니다. **Mode**, **AnimateCollapse**, **AllowMultiExpand** 등 단순한 속성은 XAML 특성으로 설정할 수 있습니다.
- 코드 비하인드에서 **ImageSource** 목록인 **Images**를 할당하고, 항목을 추가하고, 이벤트를 할당합니다. 이벤트는 WPF 이벤트가 아니라 대리자이므로 XAML 특성으로 설정할 수 없습니다.

예제

MainWindow.xaml은 바, Collapse 버튼, 선택한 페이지를 표시하는 텍스트를 선언합니다. 프로젝트의 App.xaml은 WPF 템플릿의 것이며 StartupUri가 MainWindow.xaml로 설정되어 있습니다.

```
<Window x:Class="NavigationQuickStartWpf.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcNavigation;assembly=esegece.sgcNavigation.Wpf"
        Title="sgcNavigation WPF" Width="720" Height="460">
    <DockPanel>
        <!-- the width follows ExpandedWidth and CollapsedWidth, in device independent units -->
        <sgc:TsgcWpfNavigationBar x:Name="Nav" DockPanel.Dock="Left" Mode="nbmSidebar" AnimateCollapse="True" />
        <Button x:Name="btnCollapse" DockPanel.Dock="Top" Content="Collapse" Click="btnCollapse_Click" />
        <TextBlock x:Name="lblPage" FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Center" />
    </DockPanel>
</Window>
```

MainWindow.xaml.cs는 세 개의 **DrawingImage** 벡터 아이콘을 만들고, 항목을 추가하고, 이벤트를 할당합니다.

```
using System.Collections.Generic;
using System.Windows;
using System.Windows.Media;
using esegece.sgcNavigation;

namespace NavigationQuickStartWpf
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            // the items and the events are set in code
            Nav.Images = new List<ImageSource>
            {
                DoIcon(Colors.SteelBlue, "M2,2 H7 V7 H2 Z M9,2 H14 V7 H9 Z M2,9 H7 V14 H2 Z M9,9 H14 V14 H9 Z"),
                DoIcon(Colors.SeaGreen, "M1,3 H15 V13 H1 Z"),
                DoIcon(Colors.DarkOrange, "M2,8 H5 V15 H2 Z M6,4 H9 V15 H6 Z M10,1 H13 V15 H10 Z")
            };
            DoAddItem("Dashboard", 0, "");
            DoAddItem("Inbox", 1, "3");
            DoAddItem("Reports", 2, "").Hint = "Monthly reports";
            Nav.OnSelectionChange = NavSelectionChange;
            Nav.OnCollapsedChange = NavCollapsedChange;
            Nav.SelectedIndex = 0;
        }

        private static ImageSource DoIcon(Color aColor, string aGeometry)
```

```
        {
            // a vector icon, drawn at 16 x 16 device independent units
            GeometryDrawing oDrawing = new GeometryDrawing(new SolidColorBrush(aColor), null,
                Geometry.Parse(aGeometry));
            DrawingImage oImage = new DrawingImage(oDrawing);
            oImage.Freeze();
            return oImage;
        }

        private TsgcNavigationItem DoAddItem(string aCaption, int aImageIndex, string aBadge)
        {
            TsgcNavigationItem oItem = Nav.Items.AddItem(aCaption);
            oItem.ImageIndex = aImageIndex;
            oItem.Badge = aBadge;
            return oItem;
        }

        private void NavSelectionChange(object Sender)
        {
            if (Nav.SelectedIndex >= 0)
                lblPage.Text = Nav.Items[Nav.SelectedIndex].Caption + " page";
        }

        private void NavCollapsedChange(object Sender)
        {
            btnCollapse.Content = Nav.Collapsed ? "Expand" : "Collapse";
        }

        private void btnCollapse_Click(object sender, RoutedEventArgs e)
        {
            Nav.Collapsed = !Nav.Collapsed;
        }
    }
}
```

Images 는 **DrawingImage** 뿐 아 니 라 **BitmapImage** 등 모 든 **ImageSource** 를 받 습 니 다. 목 록 이 **ObservableCollection<ImageSource>**이면 이미지가 추가, 제거, 교체될 때 바가 다시 그려집니다.

참고 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- WinForms 빠른 시작
- Avalonia 빠른 시작
- 이미지와 글리프

Avalonia 빠른 시작

TsgcAvaloniaNavigationBar는 Avalonia 내비게이션 바이며, Delphi의 TsgcFMXNavigationBar에 대응하고 Windows, Linux, macOS 에서 동작합니다. 마우스, 터치, 펜은 같은 포인터 경로를 거칩니다. 패키지는 **esegece.sgcNavigation.Avalonia**입니다.

단계

- .NET 8 프로젝트를 만들고 esegece.sgcNavigation.Avalonia, Avalonia.Desktop, Avalonia.Themes.Fluent 패키지를 추가합니다.
- `xmlns:sgc="using:esegece.sgcNavigation"`으로 네임스페이스를 매핑하고, **Mode**를 지정하여 **TsgcAvaloniaNavigationBar**를 선언합니다.
- 코드 비하인드에서 **Theme**를 설정하고, 그룹과 항목을 추가하고, 이벤트를 할당합니다. 이벤트는 Avalonia 이벤트가 아니라 대리자입니다.

예제

MainWindow.axaml은 한 번에 하나의 그룹만 열리는 아코디언을 선언합니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="using:esegece.sgcNavigation"
        x:Class="NavigationQuickStartAvalonia.MainWindow"
        Title="sgcNavigation Avalonia" Width="720" Height="460">
    <DockPanel>
        <!-- an accordion: a click on a group header expands or collapses the group -->
        <sgc:TsgcAvaloniaNavigationBar x:Name="Nav" DockPanel.Dock="Left" Mode="nbmAccordion"
                                       AllowMultiExpand="False" />
        <TextBlock x:Name="lblPage" Margin="24" FontSize="20" />
    </DockPanel>
</Window>
```

MainWindow.axaml.cs는 다크 테마를 설정하고, 세 개의 그룹과 그 항목을 추가하고, 클릭한 항목을 표시합니다.

```
using Avalonia.Controls;
using esegece.sgcNavigation;
using esegece.sgcUICore;

namespace NavigationQuickStartAvalonia
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            // Theme is a TsgcUITheme; the items and the events are set in code
            Nav.Theme = TsgcUITheme.utDark;
            DoAddGroup("Sales", "New order", "Customers", "Invoices");
            DoAddGroup("Inventory", "Products", "Stock");
            DoAddGroup("Admin", "Users", "Audit log");
            Nav.Items[1].Badge = "5";
            Nav.OnItemClick = NavItemClick;
        }

        private void DoAddGroup(string aCaption, params string[] aItems)
        {
            // the items that follow a group header belong to the group
            Nav.Items.BeginUpdate();
            try
            {
                Nav.Items.AddItem(aCaption, "", true);
```

```
                foreach (string vCaption in aItems)
                    Nav.Items.AddItem(vCaption);
            }
            finally
            {
                Nav.Items.EndUpdate();
            }
        }

        private void NavItemClick(object Sender, TsgcNavigationItem aItem)
        {
            // a group header does not fire OnItemClick, it expands or collapses its group
            lblPage.Text = aItem.Caption;
        }
    }
}
```

Program.cs는 Fluent 테마로 애플리케이션을 시작합니다.

```
using System;
using Avalonia;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;

namespace NavigationQuickStartAvalonia
{
    internal class App : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new MainWindow();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main(string[] args)
        {
            BuildAvaloniaApp().StartWithClassicDesktopLifetime(args);
        }

        public static AppBuilder BuildAvaloniaApp()
        {
            return AppBuilder.Configure<App>().UsePlatformDetect().LogToTrace();
        }
    }
}
```

Theme는 Avalonia 컨트롤의 `StyledElement.Theme` 속성을 가리므로, XAML 특성이 아니라 TsgcAvaloniaNavigationBar 형식의 변수에 코드로 설정하십시오. Images는 **IImage** 목록이며, **Bitmap** 또는 **DrawingImage**를 사용합니다.

참고 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점

- WinForms 빠른 시작
- WPF 빠른 시작
- 아코디언 모드

.NET 클래스

각 호스트에는 같은 멤버를 가진 내비게이션 바 클래스가 하나씩 있습니다. 프레임워크에 독립적인 형식은 세 호스트가 공유하며 Delphi 이름을 유지합니다.

호스트 클래스

호스트	클래스	기본 클래스	Delphi
WinForms	TsgcWinFormsNavigationBar	TsgcUIWinFormsControl	TsgcVCLNavigationBar
WPF	TsgcWpfNavigationBar	TsgcUIWpfControl	TsgcFMXNavigationBar
Avalonia	TsgcAvaloniaNavigationBar	TsgcUIAvaloniaControl	TsgcFMXNavigationBar

프레임워크에 독립적인 형식

형식	설명
TsgcNavigationModel	항목, 모드, 접힌 레일, 선택, 핫, 눌림, 포커스 항목, 행 레이아웃, 히트 테스트. 모든 바가 하나씩 소유합니다. TsgcNavigationModel을 참조하십시오.
TsgcNavigationItems	항목 컬렉션: Count, 인덱서, Add, AddItem(aCaption, aGlyph = "", aIsGroup = false), Insert, Delete, Clear, IndexOf, BeginUpdate, EndUpdate.
TsgcNavigationItem	하나의 항목: Caption, Glyph, ImageIndex, Badge, IsGroup, Expanded, Enabled, Visible, Hint, Tag (long), OnClick, Click(), 그리고 Registered 에디션에서는 Command와 CommandParameter. TsgcNavigationItem을 참조하십시오.
TsgcNavigationMode	nbmSidebar, nbmAccordion.
TsgcNavigationPart	npNone, npItem, npExpander: ItemAtPos가 반환하는 행의 부분.
TsgcNavigationItemEvent	`void (object Sender, TsgcNavigationItem aItem)`, OnItemClick의 형식.
TsgcUINotifyEvent	`void (object Sender)`, esegece.sgcUICore에서 제공.
sgcNavigation_Render, sgcNavigation_Rendering	렌더러와 헤드리스 SVG 및 PNG 출력. .NET 헤드리스 렌더링을 참조하십시오.

내비게이션 바

멤버	형식	설명
Model	TsgcNavigationModel	바의 모델.
Items	TsgcNavigationItems	항목과 그룹 헤더(순서대로).
Mode	TsgcNavigationMode	사이드바 또는 아코디언.
Collapsed	bool	바를 아이콘 레일로 접습니다. 너비가 ExpandedWidth와 CollapsedWidth 사이에서 이동

		하며, AnimateCollapse가 true이면 애니메이션이 적용됩니다.
ExpandedWidth, CollapsedWidth	int	펼친 바와 아이콘 레일의 너비. 기본값은 각각 220과 56입니다.
ItemHeight, GroupHeight, Indent	int	항목 행의 높이, 그룹 헤더 행의 높이, 아코디언 모드에서 그룹 항목의 들여쓰기.
AllowMultiExpand	bool	아코디언 모드에서 false이면 한 그룹을 열 때 다른 그룹이 닫힙니다.
SelectedIndex	int	선택한 항목. 없으면 -1.
Theme, AccentColor	TsgcUITheme, uint	바의 테마. AccentColor가 0이면 테마의 강조색을 사용합니다.
Images	ImageList, IList<ImageSource>, IList<IImage>	ImageIndex의 이미지. Delphi와 .NET의 차이점을 참조하십시오.
AnimateCollapse	bool	Collapsed를 약 150 ms 동안 애니메이션합니다.
ScrollY	int	세로 스크롤 오프셋(장치 픽셀 단위).
ItemAtPos(X, Y), ItemAtPos(X, Y, out aPart)	int	컨트롤의 한 지점에 있는 항목의 인덱스(없으면 -1)와 행의 부분.
ItemRect(aIndex)	Rectangle, Rect	컨트롤 좌표에서 항목의 행.
ScrollIntoView(aIndex)	void	항목의 행이 보이도록 바를 스크롤합니다.
OnItemClick	TsgcNavigationItemEvent	활성화된 항목에 대한 클릭, Enter 또는 Space. 항목의 OnClick 또는 Command가 실행되기 전에 발생합니다. 그룹 헤더에서는 발생하지 않습니다.
OnSelectionChange	TsgcUINotifyEvent	사용자 또는 코드에 의해 SelectedIndex가 변경되었습니다.
OnCollapsedChange	TsgcUINotifyEvent	Collapsed가 변경되었습니다.

위치와 크기는 WinForms(Rectangle)에서는 픽셀이고, WPF와 Avalonia(Rect)에서는 장치 독립 단위입니다.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작

Delphi와 .NET의 차이점

.NET API는 Delphi API를 이름 그대로 따릅니다. 다른 점은 다음과 같습니다.

액션 링크 대신 Command

Delphi는 항목의 Action 속성으로 TBasicAction에 연결합니다. Action 링크를 참조하십시오. .NET에는 TBasicAction이 없으므로, 항목에는 `System.Windows.Input.ICommand`인 **Command**와 **CommandParameter**가 있습니다. 둘 다 Delphi 액션 링크와 마찬가지로 Registered 에디션에만 포함됩니다.

- `CanExecute(CommandParameter)`가 항목의 **Enabled**를 결정합니다. 명령이나 매개 변수가 할당될 때와 `CanExecuteChanged`가 발생할 때마다 평가됩니다.
- 클릭, Enter, Space는 바의 OnItemClick을 발생시킨 다음, 실행할 수 있으면 명령을 실행합니다. 항목에 OnClick 처리기가 있으면 Delphi와 마찬가지로 명령 대신 OnClick이 실행됩니다.
- ICommand에는 캡션, 힌트, 이미지 인덱스가 없으므로 Caption, Hint, ImageIndex는 명령에서 동기화되지 않습니다. 이 값들은 항목의 속성으로 남습니다. Visible도 동기화되지 않습니다.
- 유휴 시 OnUpdate는 없습니다. 상태가 바뀌는 명령은 CanExecuteChanged를 발생시키고, 항목은 즉시 이를 따릅니다.

항목을 최소한의 명령에 연결하고, 실행하고, CanExecuteChanged로 비활성화하는 콘솔 프로그램입니다. esegece.sgcNavigation만 필요하며, 세 호스트의 바도 같은 방식으로 동작합니다.

```
using System;
using System.Windows.Input;
using esegece.sgcNavigation;

namespace NavigationCommands
{
    // a minimal ICommand; MVVM toolkits provide the same as RelayCommand
    internal class TExportCommand : ICommand
    {
        private bool FAllowed = true;

        public event EventHandler CanExecuteChanged;

        public bool Allowed
        {
            get { return FAllowed; }
            set
            {
                FAllowed = value;
                EventHandler vHandler = CanExecuteChanged;
                if (vHandler != null)
                    vHandler(this, EventArgs.Empty);
            }
        }

        public bool CanExecute(object parameter)
        {
            return FAllowed;
        }

        public void Execute(object parameter)
        {
            Console.WriteLine("Export " + parameter);
        }
    }

    internal static class Program
    {
        private static void Main()
        {
            TsgcNavigationModel oModel = new TsgcNavigationModel();
            TsgcNavigationItem oItem = oModel.Items.AddItem("Export");
```

```
            TExportCommand oCommand = new TExportCommand();

            // Enabled follows CanExecute, a click executes the command
            oItem.Command = oCommand;
            oItem.CommandParameter = "orders.csv";
            oItem.Click();

            // CanExecuteChanged disables the item
            oCommand.Allowed = false;
            Console.WriteLine("Enabled: " + oItem.Enabled);

            // the caption is the item's own, the command does not change it
            oItem.Caption = "Export orders";
            Console.WriteLine(oItem.Caption);
        }
    }
}
```

이미지

Delphi는 VCL에서는 TCustomImageList를, FireMonkey에서는 FMX.ImgList의 TCustomImageList를 사용합니다. 이미지와 글리프를 참조하십시오. .NET에서는 **Images** 속성이 각 호스트의 이미지 형식을 가지며, 항목의 **ImageIndex**는 그 목록의 인덱스입니다.

호스트	Images	그려지는 크기
WinForms	System.Windows.Forms.ImageList	이미지 목록의 ImageSize(픽셀 단위).
WPF	IList<ImageSource>: 비트맵과 DrawingImage 벡터 아이콘. ObservableCollection은 변경될 때 다시 그려집니다.	16 x 16 장치 독립 단위. 벡터 아이콘은 모든 DPI에서 선명합니다.
Avalonia	IList<IImage>: Bitmap 또는 DrawingImage. INotifyCollectionChanged가 있는 목록은 변경될 때 다시 그려집니다.	16 x 16 장치 독립 단위.

모든 이미지는 한 번만 ARGB로 변환되며 목록이 바뀔 때까지 캐시됩니다. WPF와 Avalonia에서는 DPI 배율이 바뀔 때까지도 캐시됩니다. 항목에 이미지가 없으면 **Glyph**의 텍스트가 아이콘 슬롯에 그려집니다.

힌트와 툴팁

Delphi에서는 ShowHint가 True일 때 바가 커서 아래 항목의 Hint를 표시하며, FMX는 10.1 Berlin부터 힌트를 표시합니다. .NET에서는 바가 항상 툴팁을 표시하며 ShowHint 속성은 없습니다. 커서 아래 항목의 Hint, 또는 캡션이 보이지 않는 접힌 레일에서는 항목의 Caption이 표시됩니다. WinForms는 내부 ToolTip 컴포넌트를 사용하고, WPF는 바의 ToolTip 속성을 설정하며, Avalonia는 ToolTip.Tip 연결 속성을 설정합니다. 바의 툴팁을 직접 설정하지 마십시오. 바가 대체합니다. 대신 항목의 Hint를 설정하십시오.

단위

- WinForms: ExpandedWidth, CollapsedWidth, ItemHeight, GroupHeight, Indent는 VCL 바가 PPI / 96을 사용하는 것과 마찬가지로 논리 픽셀에 DPI / 96을 곱한 값입니다. ItemAtPos, ItemRect, ScrollY는 클라이언트 픽셀입니다.
- WPF와 Avalonia: Width, ExpandedWidth, CollapsedWidth, ItemAtPos, ItemRect는 FMX와 마찬가지로 장치 독립 단위입니다. 프레임, 모델의 레이아웃, **ScrollY**는 장치 픽셀이며, 단위에 컨트롤의 DPI 배율(WPF) 또는 창의 렌더링 배율(Avalonia)을 곱한 값입니다. 항목의 BoundsRect도 장치 픽셀입니다.
- Headless: aWidth와 aHeight는 논리 픽셀이며 aScale이 여기에 곱해집니다. .NET 헤드리스 렌더링을 참조하십시오.

이벤트

OnItemClick, OnSelectionChange, OnCollapsedChange, 항목의 OnClick은 Delphi 이름을 가진 대리자 속성이며 C# `event` 멤버가 아닙니다. 처리기를 할당하면 Delphi 메서드 포인터처럼 이전 처리기가 대체됩니다. 처리기는 코드에서 할당하며, XAML로 설정할 수 없는 WPF와 Avalonia에서도 마찬가지입니다.

디자이너 동사 없음

Delphi에서는 Items 속성을 디자인 타임에 표준 컬렉션 편집기로 편집합니다. .NET 포트에는 디자이너 동사도 컬렉션 편집기도 없습니다. 바의 Items는 직렬화되지 않으므로 항상 코드에서, 보통 폼이나 창의 생성자에서 추가합니다. WinForms에서는 바가 도구 상자에 있고 단순한 속성은 속성 창에서 설정합니다. WPF와 Avalonia에서는 바를 XAML에서 선언합니다.

에디션

.NET 포트는 Registered 및 Community 패키지로 제공됩니다. .NET 개요를 참조하십시오. Community 패키지에는 Command와 CommandParameter가 없습니다. Trial 에디션은 없습니다.

데이터 바인딩

WPF와 Avalonia에서 사용자용 속성은 종속성 속성과 Avalonia 속성이므로 데이터 바인딩, 스타일 지정, 애니메이션이 가능합니다. .NET 데이터 바인딩을 참조하십시오.

참고 항목

- .NET 개요
- .NET 클래스
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- Action 링크
- 이미지와 글리프

.NET 데이터 바인딩

WPF 및 Avalonia 호스트에서 탐색 막대의 사용자용 속성은 종속성 속성(WPF)과 Avalonia 속성이므로 뷰 모델에 데이터 바인딩하거나 Style로 설정하거나 애니메이션을 적용할 수 있습니다. 이 항목은 WPF와 Avalonia에만 적용됩니다. WinForms 컨트롤은 변경되지 않았으며, 해당 속성은 이전과 같은 일반 CLR 속성입니다.

바인딩 가능한 속성

표의 모든 속성에는 접미사 Property가 붙은 정적 필드가 있습니다. 예를 들어 `TsgcWpfNavigationBar.SelectedIndexProperty`입니다. WPF에서 이 필드는 DependencyProperty입니다. Avalonia에서는 StyledProperty이며, SelectedIndex만 DirectProperty입니다. **TwoWay**로 표시된 속성은 막대도 변경하므로(클릭이나 키 입력 후의 SelectedIndex, 모델 변경 후의 둘 다) 기본적으로 TwoWay로 바인딩되며, 따라서 Mode가 없는 바인딩도 뷰 모델을 최신 상태로 유지합니다. 나머지 속성은 기본적으로 OneWay로 바인딩됩니다.

WPF 클래스	Avalonia 클래스	바인딩 가능한 속성
TsgcWpfNavigationBar	TsgcAvaloniaNavigationBar	Mode, **Collapsed (TwoWay)**, ExpandedWidth, CollapsedWidth, ItemHeight, GroupHeight, Indent, AllowMultiExpand, **SelectedIndex (TwoWay)**, Theme, AccentColor, AnimateCollapse

Avalonia에서 Theme은 sgcUICore 테마(TsgcUITheme)이며 StyledElement의 Avalonia 컨트롤 테마를 가립니다. 후자는 `StyledElement.ThemeProperty`를 통해 계속 접근할 수 있습니다.

동작

- **모델이 값을 유지합니다.** 모든 속성은 탐색 모델에 저장됩니다. 바인딩된 값은 모델로 들어가고, 속성은 모델이 유지하는 값을 다시 가져옵니다. 모델은 1보다 작은 너비나 높이를 거부하고 SelectedIndex를 항목 범위 안에 유지하며, 속성은 그 뒤에 사용 중인 값을 표시합니다. TwoWay 소스도 WPF와 Avalonia 모두에서 수정된 값을 받습니다. WPF에서는 디스패처가 대기 중인 작업을 실행한 뒤에 도착합니다. Avalonia에서 다시 쓰기는 바인딩이 실행 중일 때 소스가 보낸 값에 적용됩니다. 바인딩이 시작될 때 소스가 이미 가지고 있는 값은 컨트롤에서 수정되지만 소스에는 다시 기록되지 않습니다.
- **막대가 만든 변경은 뷰 모델에 도달합니다.** 항목 클릭, 키 입력, 모델의 변경, 코드에서의 변경은 SelectedIndex를 업데이트하고, 모델 또는 코드에서의 변경은 Collapsed를 업데이트합니다. 기본 TwoWay 바인딩에서는 뷰 모델이 따라갑니다.
- **이벤트는 모든 소스에서 발생합니다.** OnSelectionChange와 OnCollapsedChange는 입력, 코드, 바인딩 중 어느 것에서 온 변경이든 발생합니다.
- **항목 먼저, 그다음 바인딩.** SelectedIndex는 항목을 참조하므로 바인딩된 값이 도착하기 전에 항목을 추가하십시오. 예를 들어 예제처럼 항목을 추가한 뒤에 DataContext를 설정합니다.
- **스타일.** Style setter는 바인딩 가능한 모든 속성을 설정할 수 있습니다. 예를 들어 창에 있는 모든 탐색 막대의 Mode와 Theme입니다.

CLR 속성으로 남는 속성

속성	이유
Items	모델의 항목 트리이며 AddItem과 컬렉션의 다른 메서드로 만듭니다. 바인딩할 단일 값이 아닙니다.
Images	항목의 이미지 목록이며 참조로 공유되는 개체입니다.
Model	컨트롤 뒤에 있는 플랫폼 독립적인 모델이며 읽기 전용입니다.
ScrollY	스크롤 위치이며 휠을 한 단계 돌릴 때마다 바뀌므로 설정이 아닙니다.

OnSelectionChange, OnCollapsedChange 및 기타 이벤트	Delphi와 같은 이름의 델리게이트 속성으로, 코드에서 할당합니다. Delphi와 .NET의 차이점을 참조하십시오.

예제

아래 뷰 모델은 두 예제에서 공유됩니다. SelectedIndex와 Collapsed는 Mode 없이 바인딩되므로 TwoWay로 바인딩됩니다. 항목을 클릭하면 뷰 모델이 바뀌고, 확인란은 뷰 모델을 통해 막대를 접습니다.

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace NavigationBinding
{
    public class MainViewModel : INotifyPropertyChanged
    {
        private int FSelectedIndex = -1;
        private bool FCollapsed;

        public event PropertyChangedEventHandler PropertyChanged;

        // bound to SelectedIndex, TwoWay by default
        public int SelectedIndex
        {
            get { return FSelectedIndex; }
            set { DoSet(ref FSelectedIndex, value); }
        }

        // bound to Collapsed, TwoWay by default
        public bool Collapsed
        {
            get { return FCollapsed; }
            set { DoSet(ref FCollapsed, value); }
        }

        private void DoSet<T>(ref T aField, T aValue, [CallerMemberName] string aName = null)
        {
            if (EqualityComparer<T>.Default.Equals(aField, aValue))
                return;
            aField = aValue;
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(aName));
        }
    }
}
```

WPF

MainWindow.xaml은 막대를 바인딩합니다. App.xaml은 WPF 템플릿의 것입니다.

```
<Window x:Class="NavigationBinding.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcNavigation;assembly=esegece.sgcNavigation.Wpf"
        Title="sgcNavigation binding" Width="720" Height="460">
    <DockPanel>
        <!-- SelectedIndex and Collapsed bind TwoWay by default -->
        <sgc:TsgcWpfNavigationBar x:Name="Nav" DockPanel.Dock="Left" Mode="nbmSidebar"
                                  SelectedIndex="{Binding SelectedIndex}" Collapsed="{Binding Collapsed}" />
        <StackPanel Margin="24">
            <CheckBox Content="Collapsed" IsChecked="{Binding Collapsed}" />
            <TextBlock FontSize="20" Text="{Binding SelectedIndex}" />
        </StackPanel>
```

```
    </DockPanel>
</Window>
```

MainWindow.xaml.cs는 항목을 추가한 다음 뷰 모델을 DataContext로 설정합니다.

```
using System.Windows;

namespace NavigationBinding
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Nav.Items.AddItem("Dashboard");
            Nav.Items.AddItem("Inbox");
            Nav.Items.AddItem("Reports");
            // set after the items, so the bound SelectedIndex finds its item
            DataContext = new MainViewModel();
        }
    }
}
```

Avalonia

MainWindow.axaml은 `x:DataType`을 뷰 모델로 지정하므로 바인딩은 컴파일된 바인딩입니다. App 클래스는 Avalonia 빠른 시작의 것입니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:NavigationBinding"
        xmlns:sgc="using:esegece.sgcNavigation"
        x:Class="NavigationBinding.MainWindow"
        x:DataType="local:MainViewModel"
        Title="sgcNavigation binding" Width="720" Height="460">
    <DockPanel>
        <!-- SelectedIndex and Collapsed bind TwoWay by default -->
        <sgc:TsgcAvaloniaNavigationBar x:Name="Nav" DockPanel.Dock="Left" Mode="nbmSidebar"
                                       SelectedIndex="{Binding SelectedIndex}" Collapsed="{Binding Collapsed}" />
        <StackPanel Margin="24">
            <CheckBox Content="Collapsed" IsChecked="{Binding Collapsed}" />
            <TextBlock FontSize="20" Text="{Binding SelectedIndex}" />
        </StackPanel>
    </DockPanel>
</Window>
```

MainWindow.axaml.cs는 항목을 추가한 다음 뷰 모델을 DataContext로 설정합니다.

```
using Avalonia.Controls;

namespace NavigationBinding
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            Nav.Items.AddItem("Dashboard");
            Nav.Items.AddItem("Inbox");
            Nav.Items.AddItem("Reports");
            // set after the items, so the bound SelectedIndex finds its item
            DataContext = new MainViewModel();
        }
    }
}
```

참고 항목

- Delphi와 .NET의 차이점
- .NET 클래스
- WPF 빠른 시작
- Avalonia 빠른 시작

.NET 헤드리스 렌더링

내비게이션 바는 창 없이 서버, 테스트, 보고서에서 그릴 수 있습니다. **sgcNavigation_Render.sgcNavigationSaveToSVG**는 esegece.sgcNavigation 패키지만으로 SVG 파일 또는 스트림을 작성하고, **sgcNavigation_Rendering.sgcNavigationSaveToPNG**는 esegece.sgcNavigation.Rendering으로 PNG를 작성합니다. 후자는 Windows, Linux, macOS의 SkiaSharp에 그립니다. SVG 출력은 Delphi 헤드리스 렌더링의 출력과 바이트 단위로 동일합니다.

메서드

메서드	패키지	설명
sgcNavigation_Render.sgcNavigationSaveToSVG(aModel, aFileName or aStream, aWidth, aHeight, aScale = 1, aImages = null)	esegece.sgcNavigation	aModel의 바를 SVG로 작성합니다.
sgcNavigation_Rendering.sgcNavigationSaveToPNG(aModel, aFileName or aStream, aWidth, aHeight, aScale = 1, aImages = null)	esegece.sgcNavigation.Rendering	aModel의 바를 투명 배경의 PNG로 작성합니다.
sgcNavigation_Render.sgcNavigationRender(aModel, aCanvas, aWidth, aHeight, aScale = 1, aScrollY = 0, aImages = null)	esegece.sgcNavigation	임의의 sgcUICore 캔버스에 바를 그립니다.

aWidth와 aHeight는 논리 픽셀입니다. 이미지 크기는 aWidth x aScale 곱하기 aHeight x aScale 픽셀이므로, 배율 2는 고DPI 화면이나 인쇄에 적합한 선명한 이미지를 만듭니다. aImages는 선택 사항인 **TsgcNavigationImageEvent**, `bool (int aImageIndex, out TsgcUIARGBBuffer aImage)`로, ImageIndex의 이미지를 **TsgcUIARGBBuffer**로 반환합니다. 지정하지 않으면 항목의 Glyph가 아이콘 슬롯에 그려집니다.

예제

esegece.sgcNavigation.Rendering을 참조하는 콘솔 프로그램(`net8.0`)입니다. 배지, 그룹 헤더, 비활성 항목이 있는 사이드바, 접힌 아이콘 레일, 그리고 같은 항목을 닫힌 그룹이 있는 다크 아코디언으로 PNG와 SVG로 작성합니다.

```
using System;
using esegece.sgcNavigation;
using esegece.sgcUICore;

namespace NavigationHeadless
{
    internal static class Program
    {
        private static void Main()
        {
            // a sidebar with text glyphs in the icon slot, a badge and a disabled item
            TsgcNavigationModel oModel = new TsgcNavigationModel();
            oModel.Items.AddItem("Dashboard", "D");
            oModel.Items.AddItem("Inbox", "I").Badge = "12";
            oModel.Items.AddItem("Reports", "R");
            oModel.Items.AddItem("Admin", "", true);
            oModel.Items.AddItem("Billing", "B").Enabled = false;
            oModel.Items.AddItem("Settings", "S");
            oModel.SelectedIndex = 1;
```

```
            // aWidth and aHeight are logical pixels, the PNG is twice as large
            sgcNavigation_Rendering.sgcNavigationSaveToPNG(oModel, "sidebar.png", 220, 300, 2);
            sgcNavigation_Render.sgcNavigationSaveToSVG(oModel, "sidebar.svg", 220, 300);

            // the collapsed icon rail, CollapsedWidth wide
            oModel.Collapsed = true;
            sgcNavigation_Rendering.sgcNavigationSaveToPNG(oModel, "rail.png", oModel.CollapsedWidth, 300);

            // the same items as a dark accordion with the Admin group closed
            oModel.Collapsed = false;
            oModel.Mode = TsgcNavigationMode.nbmAccordion;
            oModel.Theme = TsgcUITheme.utDark;
            oModel.Items[3].Expanded = false;
            sgcNavigation_Rendering.sgcNavigationSaveToPNG(oModel, "accordion.png", 220, 300);
            Console.WriteLine("sidebar.png, sidebar.svg, rail.png and accordion.png written");
        }
    }
}
```

참고 항목

- .NET 개요
- .NET 클래스
- 헤드리스 렌더링
- TsgcNavigationModel

.NET 데모

.NET 데모는 제품의 `demos` 폴더와 데모 zip에 솔루션 `sgcNavigation_Demos.sln`과 함께 들어 있습니다. 각 데모에는 README.md가 있습니다.

데모	호스트	설명
SidebarApp	WinForms	왼쪽의 사이드바로 애플리케이션의 페이지를 전환합니다. 다섯 개의 항목(Dashboard, Inbox, Reports, Billing, Settings)은 ImageList에서 아이콘을 가져오고, Inbox와 Reports는 배지를 표시하며, Billing은 비활성화되어 있습니다. Collapse 버튼은 바를 애니메이션과 함께 아이콘 레일로 접습니다. OnSelectionChange는 선택한 항목의 Tag에 저장된 페이지를 표시하고, OnItemClick은 클릭한 항목을 상태 줄에 쓰고, OnCollapsedChange는 버튼 캡션을 갱신합니다. 첫 페이지에는 실시간 KPI 카드 두 개가 있는 sgcDashboard 대시보드가 있으며, Theme 콤보로 바, 대시보드, 페이지를 라이트와 다크 사이에서 전환합니다.
SidebarAppWpf	WPF	DrawingImage 벡터 아이콘이 있는 TsgcWpfNavigationBar와 XAML에서 선언한 TsgcWpfDashboard로 만든 같은 애플리케이션.
SidebarAppAvalonia	Avalonia	IImage 벡터 아이콘이 있는 TsgcAvaloniaNavigationBar로 만든 같은 애플리케이션. Windows, Linux, macOS에서 동작합니다.
Accordion	WinForms	Sales, Inventory, Admin 그룹이 있는 아코디언입니다. "Allow multi expand"를 선택 해제하면 그룹을 열 때 다른 그룹이 닫힙니다. New order, Export, Audit log는 항목의 Command로 명령에 연결되어 있습니다. 클릭하면 명령이 실행되고 Export 항목은 CanExecute를 따르며, "Export command enabled" 확인란으로 CanExecute를 바꿀 수 있습니다. ICommand에는 캡션이 없으므로 Rename 버튼은 항목 자체의 이름을 바꿉니다. Community 에디션에서는 항목이 OnClick으로 동작합니다.
AccordionWpf	WPF	TsgcWpfNavigationBar로 만든 같은 아코디언.
AccordionAvalonia	Avalonia	TsgcAvaloniaNavigationBar로 만든 같은 아코디언.

빌드 및 실행

데모에는 .NET 8 SDK 이상이 필요합니다. 데모는 해당 폴더에서 실행합니다.

```
dotnet run
```

제품 트리 안에서는 데모가 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip에서는 NuGet 패키지(기본적으로 Community 에디션)를 복원합니다. Registered 패키지를 사용하려면 **SgcNavigationEdition**을 설정합니다.

```
dotnet build -p:SgcNavigationEdition=Registered
```

자체 점검

모든 데모에는 `--test` 스위치가 있습니다. 창을 화면 밖에서 열고, 항목을 클릭하고, 그룹을 열고 닫고, 바를 접고, 테마를 전환하며, 점검마다 PASS 또는 FAIL 한 줄을 파일에 쓰고, 실패한 점검의 수를 종료 코드로 하여 종료합니다. 따라서 0은 모든 점검을 통과했다는 뜻입니다.

```
dotnet run -- --test result.txt
```

참고 항목

- .NET 개요
- NuGet으로 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- 데모

문제 해결

Undeclared identifier Action 또는 "Property Action does not exist"

Community 에디션은 SGC_NAVIGATION_ACTIONS 없이 컴파일되므로 항목의 Action 속성이 존재하지 않습니다. 이를 사용하는 코드는 E2003 Undeclared identifier로 실패하며, Registered 빌드에서 저장한 폼에 액션에 연결된 항목이 있으면 로드할 때 "Error reading" 예외가 발생합니다. 항목의 OnClick 또는 바의 OnItemClick 이벤트를 사용하거나 Registered 또는 Trial 빌드를 사용하십시오. 에디션을 참조하십시오.

그룹 헤더를 클릭해도 선택되지 않음

그룹 헤더는 선택되지 않습니다. 사이드바 모드에서는 정적 레이블이고, 펼쳐진 아코디언에서는 클릭하면 그룹이 토글됩니다. 페이지에는 IsGroup이 False인 항목을 사용하십시오. 아코디언 모드를 참조하십시오.

항목이 클릭에 반응하지 않음

비활성 항목(Enabled False)은 흐리게 그려지며 클릭과 키보드를 무시합니다. Action이 있는 항목은 액션의 Enabled를 따르므로 액션의 OnUpdate도 확인하십시오.

FireMonkey에서 이미지가 표시되지 않음

FMX 바의 Images 속성에는 Delphi XE8 이상이 필요합니다. 이전 버전에서는 항목이 Glyph를 그립니다. 이미지와 글리프를 참조하십시오.

FMX 바에 힌트가 없음

FMX 바의 항목 힌트에는 Delphi 10.1 Berlin 이상이 필요하며 ShowHint가 True여야 합니다.

바가 애니메이션 없이 새 너비로 바로 바뀜

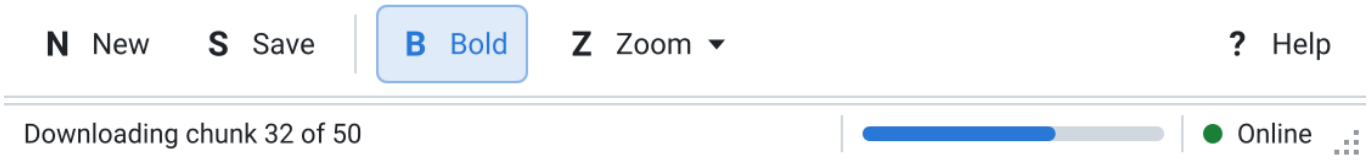
접기 애니메이션은 런타임에서만, 그리고 AnimateCollapse가 True인 동안에만 실행됩니다. 디자인 타임에는 너비가 즉시 바뀝니다.

SidebarApp, SidebarFMX 또는 Accordion 데모가 컴파일되지 않음

SidebarApp과 SidebarFMX는 대시보드를 호스팅하므로 sgcDashboard가 설치되어 있어야 합니다. Accordion은 Community 에디션에 포함되지 않는 Action 링크를 사용합니다. 데모를 참조하십시오.

소개

sgcToolbar는 VCL과 FireMonkey용 Delphi 및 C++Builder 네이티브 툴바와 상태 표시줄입니다. **TsgcVCLToolbar**와 **TsgcFMXToolbar**는 버튼, 토글, 라디오 그룹, 드롭다운 버튼, 레이블, 구분선, 스페이서를 한 줄로 표시하고, 공간에 맞지 않는 항목은 오버플로 메뉴로 옮깁니다. **TsgcVCLStatusBar**와 **TsgcFMXStatusBar**는 텍스트 패널, 진행률 표시줄, 색상 상태 표시기를 보여 주며, 작업자 스레드가 스레드 안전 setter로 이를 직접 갱신할 수 있습니다.
sgcToolbar는 eSeGeCe 비주얼 제품군의 일부이며, 다른 제품에 프레임을 제공합니다. **sgcDashboard** 대시보드, **sgcGrid** 그리드 또는 **sgcNavigation** 내비게이션 바를 호스팅하는 창의 위쪽에는 툴바가, 아래쪽에는 상태 표시줄이 놓입니다. 이 컨트롤은 캔버스, 래스터라이저, 글꼴, SVG 작성기, 라이트 및 다크 테마를 갖춘 공유 런타임 패키지인 **sgcUICore**로 그려지므로, VCL, FireMonkey, 헤드리스 PNG 및 SVG 출력에서 바의 모양이 동일합니다. sgcToolbar에는 sgcUICore가 필요하며, 함께 설치됩니다.



에디션

sgcToolbar는 Registered, Trial, Community 에디션으로 제공됩니다. Community 에디션은 오버플로 메뉴와 항목의 Action 링크를 제외한 모든 기능을 갖추고 있습니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 프레임워크에 독립적인 모델, sgcUICore 캔버스 위의 렌더러, VCL 및 FMX 컨트롤.
- 플랫폼: 지원하는 Delphi 버전, C++Builder, 플랫폼.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder: 몇 분 만에 만드는 첫 툴바와 상태 표시줄.
- 구성 요소: 툴바, 상태 표시줄, 항목, 패널, 모델.
- 항목 종류, 토글과 라디오 그룹, 드롭다운 메뉴, 오버플로, Action 링크, 이미지와 글리프, 키보드.
- 상태 패널, 진행률과 표시기, 스레드 안전 갱신, 크기 조절 핸들.
- 테마와 헤드리스 렌더링.
- 데모: Delphi 및 C++Builder 데모 프로젝트.

아키텍처

sgcToolbar는 계층으로 구성되어 있습니다. 각 계층은 아래 계층에만 의존하므로, 모델과 렌더러는 VCL이나 FireMonkey 없이 컴파일됩니다.

계층	유닛	역할
모델	sgcToolbar_Model	TsgcToolbarModel은 Items(TsgcToolbarItem 컬렉션), ShowCaptions, 크기, 테마, 상호작용 상태(핫, 눌림, 포커스된 항목)를 보관합니다. TsgcStatusBarModel은 Panels(TsgcStatusBarPanel 컬렉션), 테마, 크기 조절 핸들을 보관합니다. 두 모델의 Layout 메서드는 모든 항목이나 패널의 사각형을 계산하고, HitTest는 한 지점 아래에 있는 항목이나 패널을 찾습니다.
렌더러	sgcToolbar_Render	RenderToolbar와 RenderStatusBar는 모델을 임의의 sgcUICore 캔버스에 그립니다. 모델의 SaveToPNG와 SaveToSVG 메서드는 이를 사용해 창 없이 이미지를 씁니다. 헤드리스 렌더링을 참조하십시오.
VCL 컨트롤	sgcToolbar_VCL	TsgcVCLToolbar와 TsgcVCLStatusBar는 모델을 소유하고, 그 속성을 게시하며, sgcUICore VCL 컨트롤 기반 클래스를 통해 그립니다. 마우스, 키보드, 힌트, 이미지 목록, 팝업 메뉴, 크기 조절 핸들이 추가됩니다.
FMX 컨트롤	sgcToolbar_FMX	TsgcFMXToolbar와 TsgcFMXStatusBar는 동일한 속성, 메서드, 이벤트에 터치 입력이 더해진 FireMonkey 컨트롤입니다.
sgcUICore	sgcUICore_Canvas, sgcUICore_VCL, sgcUICore_FMX, sgcUICore_Theme	공유 런타임입니다. 벡터 캔버스와 래스터라이저, 프레임 페이싱(MaxFPS)을 갖춘 VCL 및 FMX 컨트롤 기반 클래스, 라이트 및 다크 테마 토큰을 제공합니다.

모델과 컨트롤

바의 게시된 속성은 그 모델의 속성이며, 바의 **Model** 속성으로 나머지에 접근할 수 있습니다. Layout, HitTest, 오버플로 목록, 상호작용 상태가 여기에 해당합니다. 모델이 변경되면 새 프레임이 예약될 뿐 직접 그리지 않으므로, 연속된 여러 변경도 한 번의 다시 그리기로 처리됩니다.

그리기

모든 항목과 패널은 렌더러가 그립니다. 이미지 또는 글리프가 있는 아이콘 슬롯, 캡션, 드롭다운 화살표, 핫, 눌림, 내려감, 포커스, 비활성 상태, 구분선, 오버플로 버튼, 진행률 표시줄, 표시기 점, 크기 조절 핸들이 대상입니다. 모든 크기는 논리 픽셀에 모니터 배율을 곱한 값이므로, 어떤 DPI에서도 바가 선명합니다. VCL 컨트롤은 PPI / 96을 배율로 사용하고, FMX 컨트롤은 폼의 배율을 사용합니다.

높이

툴바의 높이는 버튼을 따릅니다. 논리 픽셀 단위로 ButtonSize에 Padding의 두 배를 더한 값이며, 그 밖의 높이는 무시됩니다. 상태 표시줄은 지정한 Height를 유지하며, 기본값은 24 논리 픽셀입니다.

관련 항목

- 플랫폼
- TsgcToolbarModel
- TsgcStatusBarModel
- 테마

에디션

sgcToolbar는 모든 에디션이 동일한 소스에서 빌드됩니다. 에디션은 빌드가 컴파일하는 기능만 바꿉니다.

에디션	포함 내용
Registered	전체 기능입니다. VCL 및 FMX 툴바와 상태 표시줄, 모든 항목 종류, 토글과 라디오 그룹, 드롭다운 메뉴, 오버플로 메뉴, 항목의 Action 링크, 이미지와 글리프, 키보드 탐색, 텍스트, 진행률, 표시기 패널, 스레드 안전 setter, 크기 조절 핸들, 테마, 헤드리스 PNG 및 SVG 출력이 포함됩니다.
Trial	Registered와 동일한 기능입니다. SGC_TRIAL로 컴파일한 빌드는 빌드에 명시된 릴리스 기간이 지나면 평가판이라는 알림을 표시합니다. 그 외에는 달라지는 것이 없습니다.
Community	오버플로 메뉴와 항목의 Action 링크를 제외한 모든 기능입니다.

Community에서 제외되는 기능

Community 에디션은 SGC_TOOLBAR_ADVANCED를 정의하지 않은 상태로 컴파일됩니다. Community 빌드에서는 다음과 같습니다.

- 툴바에 오버플로 버튼이 없습니다. 공간에 맞지 않는 항목은 오른쪽 가장자리에서 잘리며, 툴바의 **OverflowMenu** 속성은 존재하지 않습니다. 오버플로를 참조하십시오.
- **Action** 속성이 TsgcToolbarItem에 존재하지 않으므로, 항목을 TAction에 연결할 수 없습니다. 대신 항목의 **OnClick** 또는 툴바의 **OnItemClick** 이벤트를 할당하십시오. Action 링크를 참조하십시오.

에디션은 실행 시 전환이 아니라 sgcVer.inc에서 정하는 컴파일 시점의 선택입니다(SGC_COMMUNITY는 SGC_TRIAL도 끕니다). Action 또는 OverflowMenu를 사용하는 코드나 폼 파일은 Community 빌드에서 컴파일되지 않거나 로드되지 않습니다. 문제 해결을 참조하십시오.

관련 항목

- 오버플로
- Action 링크
- 플랫폼

설치

sgcToolbar는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcToolbar**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcToolbar에 포함된 항목

sgcUICore는 레이아웃 제품과 sgcCharts의 공통 기반이며 sgcToolbar와 함께 제공됩니다. 자체 확인란은 없습니다. sgcToolbar를 선택하면 설치 프로그램이 자동으로 설치합니다. sgcToolbar는 sgcWebSockets 브리지 패키지가 없습니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드입니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Toolbar`	sgcToolbar 데모입니다. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcToolbar도 포함됩니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 sgcToolbar가 포함됩니다. Community의 제품은 sgcCharts, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard이며, sgcPDF와 sgcMaps는 Community에 포함되지 않습니다. sgcToolbar의 Community 빌드에서 빠지는 기능은 에디션을 참조하십시오.

이전 독립 실행형 설치

설치 프로그램이 이전 독립 실행형 sgcUICore 설치(예: 독립 실행형 sgcCharts 설치 프로그램이 만든 것)를 발견하면, 그 설치의 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcToolbarD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcToolbar<V>` / `dclsgcToolbar<V>`	sgcToolbar 컴포넌트(런타임/디자인 타임).

의존 순서대로 빌드합니다. `sgcUICore`, 그다음 `sgcToolbar`, 그다음 `dclsgcToolbar`입니다. 그런 다음 `dclsgcToolbar` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder의 경우 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcToolbarC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

.NET

.NET에서는 달라지는 것이 없습니다. sgcToolbar는 여전히 자체 NuGet 패키지(`esegece.sgcToolbar`와 그 보조 패키지)로 배포됩니다. .NET 설치를 참조하십시오.

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE~XE8, 10~10.4, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE~XE8, 10~10.4, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows의 TsgcVCLToolbar와 TsgcVCLStatusBar, Win32와 Win64, 모든 Delphi 버전.
FireMonkey	Delphi XE6부터 TsgcFMXToolbar와 TsgcFMXStatusBar(SGC_TOOLBAR_FMX), Win32, Win64, macOS, iOS, Android, 그리고 FMX for Linux(FMXLinux)를 사용하는 Linux. Images 속성에는 XE8 이상(FMX.ImgList)이, 힌트에는 10.1 Berlin 이상이 필요합니다.
Linux64	Delphi 10.2부터 프레임워크에 독립적인 유닛, 모델, 헤드리스 PNG 및 SVG 출력을 지원하는 렌더러(SGC_TOOLBAR_LINUX).
.NET	Delphi와 같은 API를 가진 C# 이식판으로, TsgcWinFormsToolbar와 TsgcWinFormsStatusBar, TsgcWpfToolbar와 TsgcWpfStatusBar, TsgcAvaloniaToolbar와 TsgcAvaloniaStatusBar, 그리고 헤드리스 PNG 및 SVG 출력을 .NET Framework 4.7.2와 .NET 8 및 9에서 제공합니다. .NET 개요를 참조하십시오.

패키지

sgcToolbar에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcToolbarD13.dpk, C++Builder 13에서는 sgcToolbarC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcToolbar	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, sgcUICore
sgcToolbarVCL	런타임, VCL	sgcToolbar_VCL (TsgcVCLToolbar, TsgcVCLStatusBar)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcToolbar
sgcToolbarFMX	런타임, FireMonkey, XE6부터	sgcToolbar_FMX (TsgcFMXToolbar, TsgcFMXStatusBar)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcToolbar
dclsgcToolbar	디자인 타임, VCL	**SGC Toolbar** 팔레트 페이지에 등록하는 컴포넌트는 TsgcVCLToolbar 및 TsgcVCLStatusBar입니다.	designide, rtl, vcl, sgcToolbar, sgcToolbarVCL
dclsgcToolbarFMX	디자인 타임, FireMonkey, XE6부터	**SGC Toolbar** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXToolbar 및 TsgcFMXStatusBar입니다.	designide, rtl, fmx, sgcToolbar, sgcToolbarFMX, dclsgcToolbar

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcToolbarFMX에는 dclsgcToolbar가 필요하므로 dclsgcToolbar를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcToolbar	sgcToolbarVCL	sgcToolbarFMX	dclsgcToolbar, dclsgcToolbarFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcToolbar, sgcToolbarVCL	sgcToolbarFMX	dclsgcToolbar, dclsgcToolbarFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcToolbar만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcToolbar만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcToolbarC13.cbproj, sgcToolbarVCLC13.cbproj, sgcToolbarFMXC13.cbproj, dclsgcToolbarC13.cbproj, dclsgcToolbarFMXC13.cbproj 입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXToolbar 및 TsgcFMXStatusBar)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcToolbarVCL은 Windows 전용입니다.

- 컨트롤을 폼에 놓거나 해당 유닛(sgcToolbar_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcToolbar는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 프레임워크에 독립적인 유닛입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.
VCL 바는 VCL 폼의 팔레트에만, FMX 바는 FireMonkey 폼의 팔레트에만 나타납니다. **Items**와 **Panels** 속성은 표준 컬렉션 편집기로 편집합니다.

관련 항목

- 에디션
- 아키텍처
- 문제 해결

빠른 시작 VCL

디자인 타임

- **SGC Toolbar** 팔레트 페이지에서 **TsgcVCLToolbar**를 폼에 올려 놓습니다. 기본적으로 위쪽에 정렬됩니다.
- 오브젝트 인스펙터에서 **Items**를 더블 클릭하고 항목을 추가합니다. 모든 버튼의 **Caption**과 **Glyph**(문자 하나 또는 짧은 기호)를 설정하고, 필요한 곳에는 **Kind**를 tikSeparator, tikToggle, tikDropdown, tikLabel, tikSpacer 중 하나로 설정합니다.
- **TsgcVCLStatusBar**를 올려 놓습니다. 기본적으로 아래쪽에 정렬됩니다. **Panels**를 더블 클릭하고 패널을 추가하며, 남는 너비를 차지할 패널에는 **Spring**을 True로 설정합니다.
- 툴바의 **OnItemClick** 핸들러를 작성하거나, 모든 항목의 **OnClick**을 할당합니다.
- 다크 팔레트를 사용하려면 두 바의 **Theme**을 utDark로 설정합니다.

코드에서

같은 바를 실행 시에 생성합니다. `sgcUI\Source`가 검색 경로에 있으면 이 프로그램은 그대로 컴파일됩니다.

관련 항목

- TsgcVCLToolbar
- TsgcVCLStatusBar
- TsgcToolbarItem
- 항목 종류
- 상태 패널

빠른 시작 FMX

FireMonkey 컨트롤에는 Delphi XE6 이상이 필요합니다. 사용법은 VCL 컨트롤과 똑같습니다. 기본적으로 위쪽에 정렬되는 **TsgcFMXToolbar**와 아래쪽에 정렬되는 **TsgcFMXStatusBar**를 올려 놓고, 컬렉션 편집기로 **Items**와 **Panels**를 채운 다음 **OnItemClick**을 처리합니다. 버튼을 탭하면 클릭과 같이 동작합니다.

코드에서

관련 항목

- TsgcFMXToolbar
- TsgcFMXStatusBar
- 플랫폼

빠른 시작 C++Builder

사용 중인 C++Builder 버전에 맞는 sgcToolbar 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 및 라이브러리 경로에 추가한 다음, **SGC Toolbar** 팔레트 페이지의 컴포넌트를 Delphi와 똑같이 사용합니다. 설치 프로그램이 두 경로를 IDE에 추가합니다. sgcToolbar와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있으므로, sgcUICore에는 별도의 경로가 필요하지 않습니다.

코드에서

OnItemClick은 클로저 이벤트이므로, 핸들러는 클래스의 __fastcall 멤버 함수입니다.

관련 항목

- 빠른 시작 VCL
- 데모
- 플랫폼

데모

데모는 설치 폴더의 **Demos** 폴더에 있으며, Delphi 프로젝트는 Demos\Delphi에, VCL 데모의 C++Builder 포팅은 Demos\CBuilder에 있습니다. 모든 데모 폴더에는 데모가 하는 일과 동작 방식을 설명하는 readme.html이 있습니다.

데모	구성 요소	보여 주는 내용
EditorShell	TsgcVCLToolbar, TsgcVCLStatusBar, TActionList	작은 텍스트 편집기의 셸입니다. 액션에 연결된 New, Open, Save 버튼, Bold와 Italic 토글, Left, Center, Right 라디오 그룹, Zoom 드롭다운, 스페이서, Dark 테마 토글, 오버플로 메뉴, 그리고 캐럿 위치와 저장 상태 표시기가 있는 상태 표시줄을 갖추고 있습니다. Delphi와 C++Builder, Win32와 Win64. Action 링크와 오버플로 메뉴는 Community 에디션에 포함되지 않으므로, 이 데모는 Community 에디션으로 빌드되지 않습니다.
EditorShellFMX	TsgcFMXToolbar, TsgcFMXStatusBar, TActionList	FireMonkey용 EditorShell 데모로, 항목, 액션, 패널이 동일합니다. Delphi, Win32와 Win64. 이 데모도 Registered 또는 Trial 빌드가 필요합니다.
LiveStatus	TsgcVCLStatusBar, TsgcVCLToolbar, TThread	작업자 스레드가 다운로드를 시뮬레이션하고, Synchronize 없이 SetPanelText, SetProgress, SetIndicator로 스프링 텍스트 패널, 진행률 패널, 표시기 패널, AutoSize 카운터를 직접 갱신합니다. Delphi와 C++Builder, Win32와 Win64.

관련 항목

- 빠른 시작 VCL
- 스레드 안전 갱신
- Action 링크

TsgcVCLToolbar

유닛: sgcToolbar_VCL
TsgcVCLToolbar는 Windows 전용 VCL 툴바입니다. 기본적으로 위쪽에 정렬되며 Items를 왼쪽에서 오른쪽으로 표시합니다. 항목 종류를 참조하십시오. 마우스, 키보드, 항목의 힌트, 드롭다운 메뉴, 오버플로 메뉴를 처리합니다. 높이는 ButtonSize와 Padding을 따르며 PPI / 96을 곱한 값입니다.

속성

속성	형식	설명
Items	TsgcToolbarItems	표시 순서대로 나열된 항목입니다. TsgcToolbarItems를 참조하십시오.
ShowCaptions	Boolean	아이콘 옆에 캡션을 그립니다. False이면 아이콘이 있는 버튼은 아이콘만 표시하고 캡션은 힌트가 됩니다. Glyph와 ImageIndex가 없는 항목은 항상 캡션을 표시합니다. 기본값 True.
ButtonSize	Integer	버튼의 높이이자 캡션이 없는 버튼의 너비이며, 논리 픽셀 단위입니다. 기본값 32.
Spacing	Integer	두 항목 사이의 간격이며, 논리 픽셀 단위입니다. 기본값 4.
Padding	Integer	버튼 주위의 여백이며, 논리 픽셀 단위입니다. 바의 높이는 ButtonSize에 Padding의 두 배를 더한 값입니다. 기본값 6.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom이며, utCustom은 라이트 색상을 사용합니다. 테마를 참조하십시오.
AccentColor	TsgcUIColor	포커스 링과 눌린 토글의 색상이며, $AARRGGBB 형식입니다. 0(기본값)이면 테마의 강조색을 사용합니다.
Images	TCustomImageList	항목의 ImageIndex가 가리키는 이미지 목록입니다. 모든 이미지는 한 번 스트레이트 ARGB로 변환되어 목록이 바뀔 때까지 캐시됩니다. 이미지와 글리프를 참조하십시오.
Model	TsgcToolbarModel (public, 읽기 전용)	바의 모델입니다. TsgcToolbarModel을 참조하십시오.
OverflowMenu	TPopupMenu (public, 읽기 전용)	오버플로 버튼의 메뉴이며, 표시될 때마다 다시 만들어집니다. Community 에디션에는 없습니다. 오버플로를 참조하십시오.
MaxFPS	Integer	sgcUICore 컨트롤 기반 클래스의 프레임 페이싱 제한입니다.

일반적인 컨트롤 멤버도 게시됩니다. **Align**(기본값 alTop), **Anchors**, **Constraints**, **Enabled**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **TabOrder**, **TabStop**(기본값 False), **Visible**, OnClick, OnContextPopup, OnEnter, OnExit, OnMouseDown, OnMouseMove, OnMouseUp, OnResize.

메서드

메서드	반환값	설명
ItemAtPos(X, Y: Integer)	Integer	컨트롤의 한 지점에 있는 항목의 인덱스이며, 오버플로 버튼이면 CS_TOOLBAR_HIT_OVERFLOW(-2), 아무것도 없으면 CS_TOOLBAR_HIT_NONE(-1)입니다. 구분선과 스페이서는 결코 히트되지 않습니다.

ItemRect(aIndex: Integer)	TRect	컨트롤 좌표로 나타낸 항목의 사각형이며, 항목이 바에 없으면 빈 사각형입니다.
OverflowRect	TRect	오버플로 버튼의 사각형이며, 오버플로가 없으면 비어 있습니다.
ClickItem(aIndex: Integer)		클릭한 것처럼 항목을 활성화합니다. 토글을 뒤집고, OnItemClick을 발생시키고, 항목의 OnClick 또는 Action을 실행하며, 드롭다운의 메뉴를 표시합니다.
InitiateAction		Action이 있는 모든 항목이 스스로 갱신하도록 합니다. 프레임워크가 유휴 시간에 호출합니다.

이벤트

이벤트	형식	설명
OnItemClick	TsgcToolbarItemEvent	활성화된 버튼, 토글 또는 드롭다운이 클릭, 탭, 오버플로 메뉴에서 선택, 또는 Enter나 Space로 활성화되었습니다. 토글의 Down은 이미 바뀌어 있습니다. 항목의 OnClick 또는 Action은 이 이벤트 다음에 실행됩니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcFMXToolbar
- TsgcToolbarItem
- TsgcToolbarModel
- 빠른 시작 VCL

TsgcFMXToolbar

유닛: sgcToolbar_FMX (Delphi XE6 이상)
TsgcFMXToolbar는 TsgcVCLToolbar와 속성, 메서드, 이벤트가 같은 FireMonkey 툴바입니다. 기본적으로 위쪽에 정렬되며 높이는 논리 단위로 ButtonSize에 Padding의 두 배를 더한 값입니다. 마우스와 터치, 포커스를 받을 수 있을 때의 키보드를 처리하고, 드롭다운 항목과 오버플로 버튼의 FireMonkey TPopupMenu를 표시합니다.

속성

속성	형식	설명
Items	TsgcToolbarItems	표시 순서대로 나열된 항목입니다. TsgcToolbarItems를 참조하십시오.
ShowCaptions	Boolean	아이콘 옆에 캡션을 그립니다. False이면 아이콘이 있는 버튼은 아이콘만 표시하고 캡션은 힌트가 됩니다. Glyph와 ImageIndex가 없는 항목은 항상 캡션을 표시합니다. 기본값 True.
ButtonSize	Integer	버튼의 높이이자 캡션이 없는 버튼의 너비이며, 논리 픽셀 단위입니다. 기본값 32.
Spacing	Integer	두 항목 사이의 간격이며, 논리 픽셀 단위입니다. 기본값 4.
Padding	Integer	버튼 주위의 여백이며, 논리 픽셀 단위입니다. 바의 높이는 ButtonSize에 Padding의 두 배를 더한 값입니다. 기본값 6.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom이며, utCustom은 라이트 색상을 사용합니다. 테마를 참조하십시오.
AccentColor	TsgcUIColor	포커스 링과 눌린 토글의 색상이며, $AARRGGBB 형식입니다. 0(기본값)이면 테마의 강조색을 사용합니다.
Images	TCustomImageList	항목의 ImageIndex가 가리키는 이미지 목록입니다. XE8 이상이 필요하며, 이미지는 16 x 16 논리 픽셀로 그려집니다. 이미지와 글리프를 참조하십시오.
Model	TsgcToolbarModel (public, 읽기 전용)	바의 모델입니다. TsgcToolbarModel을 참조하십시오.
OverflowMenu	TPopupMenu (public, 읽기 전용)	오버플로 버튼의 메뉴이며, 표시될 때마다 다시 만들어집니다. Community 에디션에는 없습니다. 오버플로를 참조하십시오.
MaxFPS	Integer	sgcUICore 컨트롤 기반 클래스의 프레임 페이싱 제한입니다.

일반적인 FireMonkey 컨트롤 멤버도 게시됩니다. **Align**(기본값 Top), **Anchors**, **CanFocus**(기본값 False), **Cursor**, **Enabled**, **HitTest**, **Locked**, **Margins**, **Opacity**, **PopupMenu**, **Position**, **TabOrder**, **Visible**, **Width**, 그리고 10.1 Berlin부터 **TabStop**, **ParentShowHint**, **ShowHint**. 이벤트에는 OnClick, OnEnter, OnExit, OnKeyDown, OnKeyUp, 마우스 이벤트, OnPainting, OnPaint, OnResize가 포함됩니다.

메서드

메서드	반환값	설명
ItemAtPos(X, Y: Single)	Integer	컨트롤의 한 지점에 있는 항목의 인덱스이며, 오버플로 버튼이면 CS_TOOLBAR_HIT_OVERFLOW(-2), 아무것도 없으면

		CS_TOOLBAR_HIT_NONE(-1)입니다. 구분선과 스페이서는 결코 히트되지 않습니다.
ItemRect(aIndex: Integer)	TRectF	컨트롤 좌표로 나타낸 항목의 사각형이며, 항목이 바에 없으면 빈 사각형입니다.
OverflowRect	TRectF	오버플로 버튼의 사각형이며, 오버플로가 없으면 비어 있습니다.
ClickItem(aIndex: Integer)		클릭한 것처럼 항목을 활성화합니다. 토글을 뒤집고, OnItemClick을 발생시키고, 항목의 OnClick 또는 Action을 실행하며, 드롭다운의 메뉴를 표시합니다.
InitiateAction		Action이 있는 모든 항목이 스스로 갱신하도록 합니다. 프레임워크가 유휴 시간에 호출합니다.

이벤트

이벤트	형식	설명
OnItemClick	TsgcToolbarItemEvent	활성화된 버튼, 토글 또는 드롭다운이 클릭, 탭, 오버플로 메뉴에서 선택, 또는 Enter나 Space로 활성화되었습니다. 토글의 Down은 이미 바뀌어 있습니다. 항목의 OnClick 또는 Action은 이 이벤트 다음에 실행됩니다.

예제

관련 항목

- TsgcVCLToolbar
- 빠른 시작 FMX
- 플랫폼

TsgcVCLStatusBar

유닛: sgcToolbar_VCL
TsgcVCLStatusBar는 Windows 전용 VCL 상태 표시줄입니다. 기본적으로 아래쪽에 정렬되며 Panels를 표시합니다. 고정, AutoSize, Spring 방식의 텍스트 패널, 진행률 패널, 표시기 패널이 있습니다. 마우스 아래에 있는 패널의 Hint를 표시하며, 크기 조절 핸들로 폼의 크기를 바꿀 수 있습니다. 모든 스레드가 세 개의 스레드 안전 setter로 패널을 갱신할 수 있습니다.

속성

속성	형식	설명
Panels	TsgcStatusBarPanels	왼쪽에서 오른쪽으로 나열된 패널입니다. TsgcStatusBarPanels를 참조하십시오.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom이며, utCustom은 라이트 색상을 사용합니다. 표시기 색상은 테마를 따릅니다.
AccentColor	TsgcUIColor	진행률 표시줄의 색상이며, $AARRGGBB 형식입니다. 0(기본값)이면 테마의 강조색을 사용합니다.
SizeGrip	Boolean	오른쪽 끝에 크기 조절 핸들을 그립니다. 폼에 크기 조절 가능한 테두리가 있고 최대화되지 않은 경우 부모 폼의 크기를 바꿉니다. 기본값 True. 크기 조절 핸들을 참조하십시오.
Height	Integer	바의 높이이며, 기본값은 24 논리 픽셀입니다.
Model	TsgcStatusBarModel (public, 읽기 전용)	바의 모델입니다. TsgcStatusBarModel을 참조하십시오.
MaxFPS	Integer	sgcUICore 컨트롤 기반 클래스의 프레임 페이싱 제한입니다.

일반적인 컨트롤 멤버도 게시됩니다. **Align**(기본값 alBottom), **Anchors**, **Constraints**, **Enabled**, **ParentShowHint**, **PopupMenu**, **ShowHint**, **Visible**, OnClick, OnContextPopup, OnMouseDown, OnMouseMove, OnMouseUp, OnResize.

메서드

메서드	반환값	설명
SetPanelText(aIndex: Integer; const aText: string)		패널의 Text를 설정합니다. 스레드 안전하며, 잘못된 인덱스는 무시됩니다. 스레드 안전 갱신을 참조하십시오.
SetProgress(aIndex, aValue: Integer)		패널의 Progress(0~100)를 설정합니다. 스레드 안전합니다.
SetIndicator(aIndex: Integer; aState: TsgcStatusIndicatorState)		패널의 IndicatorState를 설정합니다. 스레드 안전합니다.
PanelAtPos(X, Y: Integer)	Integer	컨트롤의 한 지점에 있는 패널의 인덱스이며, 없으면 -1입니다.
PanelRect(aIndex: Integer)	TRect	컨트롤 좌표로 나타낸 패널의 사각형입니다.
GripRect	TRect	크기 조절 핸들의 사각형이며, SizeGrip이 False이면 비어 있습니다.

이벤트

이벤트	형식	설명
OnPanelClick	TsgcStatusBarPanelEvent	패널이 클릭되거나 탭되었습니다. aPanel은 해당 패널입니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcFMXStatusBar
- TsgcStatusBarPanel
- 스레드 안전 갱신
- TsgcStatusBarModel

TsgcFMXStatusBar

유닛: sgcToolbar_FMX (Delphi XE6 이상)
TsgcFMXStatusBar는 TsgcVCLStatusBar와 속성, 메서드, 이벤트가 같은 FireMonkey 상태 표시줄입니다. 기본적으로 아래쪽에 정렬됩니다. 스레드 안전 setter는 다시 그리기를 메인 스레드에 큐잉합니다. 크기 조절 핸들은 그려지지만 폼의 크기를 바꾸지는 않습니다.

속성

속성	형식	설명
Panels	TsgcStatusBarPanels	왼쪽에서 오른쪽으로 나열된 패널입니다. TsgcStatusBarPanels를 참조하십시오.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom이며, utCustom은 라이트 색상을 사용합니다. 표시기 색상은 테마를 따릅니다.
AccentColor	TsgcUIColor	진행률 표시줄의 색상이며, $AARRGGBB 형식입니다. 0(기본값)이면 테마의 강조색을 사용합니다.
SizeGrip	Boolean	오른쪽 끝에 크기 조절 핸들을 그립니다. FireMonkey에는 비클라이언트 히트 테스트가 없으므로 핸들이 폼의 크기를 바꾸지 않습니다. 기본값 True. 크기 조절 핸들을 참조하십시오.
Height	Single	바의 높이이며, 기본값은 24 논리 픽셀입니다.
Model	TsgcStatusBarModel (public, 읽기 전용)	바의 모델입니다. TsgcStatusBarModel을 참조하십시오.
MaxFPS	Integer	sgcUICore 컨트롤 기반 클래스의 프레임 페이싱 제한입니다.

일반적인 FireMonkey 컨트롤 멤버도 게시됩니다. **Align**(기본값 Bottom), **Anchors**, **Cursor**, **Enabled**, **HitTest**, **Locked**, **Margins**, **Opacity**, **PopupMenu**, **Position**, **Visible**, **Width**, 그리고 10.1 Berlin부터 **Size**, **ParentShowHint**, **ShowHint**. 이벤트에는 OnClick, 마우스 이벤트, OnPainting, OnPaint, OnResize가 포함됩니다.

메서드

메서드	반환값	설명
SetPanelText(aIndex: Integer; const aText: string)		패널의 Text를 설정합니다. 스레드 안전하며, 잘못된 인덱스는 무시됩니다. 스레드 안전 갱신을 참조하십시오.
SetProgress(aIndex, aValue: Integer)		패널의 Progress(0~100)를 설정합니다. 스레드 안전합니다.
SetIndicator(aIndex: Integer; aState: TsgcStatusIndicatorState)		패널의 IndicatorState를 설정합니다. 스레드 안전합니다.
PanelAtPos(X, Y: Single)	Integer	컨트롤의 한 지점에 있는 패널의 인덱스이며, 없으면 -1입니다.
PanelRect(aIndex: Integer)	TRectF	컨트롤 좌표로 나타낸 패널의 사각형입니다.
GripRect	TRectF	크기 조절 핸들의 사각형이며, SizeGrip이 False이면 비어 있습니다.

이벤트

이벤트	형식	설명
OnPanelClick	TsgcStatusBarPanelEvent	패널이 클릭되거나 탭되었습니다. aPanel은 해당 패널입니다.

예제

FireMonkey 빠른 시작의 상태 표시줄 부분으로, 스프링 텍스트 패널과 진행률 패널입니다.

관련 항목

- TsgcVCLStatusBar
- 빠른 시작 FMX
- 크기 조절 핸들

TsgcToolbarItem

유닛: sgcToolbar_Model
TsgcToolbarItem은 툴바의 항목 하나이며, TsgcToolbarItems의 TCollectionItem입니다. VCL과 FMX 툴바가 같은 클래스를 사용합니다.

속성

속성	형식	설명
Kind	TsgcToolbarItemKind	tikButton(기본값), tikToggle, tikDropdown, tikSeparator, tikSpacer 또는 tikLabel입니다. 항목 종류를 참조하십시오.
Caption	string	항목의 텍스트입니다.
Glyph	string	항목에 이미지가 없을 때 아이콘 슬롯에 그려지는 짧은 텍스트이며, 예를 들어 Bold에는 B를 씁니다.
ImageIndex	Integer	바의 Images에서의 인덱스이며, -1(기본값)이면 없음입니다.
Hint	string	항목의 힌트입니다. 비어 있고 캡션이 숨겨져 있으면 Caption이 힌트가 됩니다.
Enabled	Boolean	비활성 항목은 흐리게 그려지며 마우스와 키보드를 무시합니다. 기본값 True.
Visible	Boolean	숨겨진 항목은 공간을 차지하지 않습니다. 기본값 True.
Down	Boolean	tikToggle 항목의 상태입니다. GroupIndex가 0보다 크면 Down을 설정할 때 같은 그룹의 다른 항목이 해제됩니다. 기본값 False.
GroupIndex	Integer	0보다 큰 같은 GroupIndex를 가진 토글은 라디오 그룹을 이룹니다. 기본값 0. 토글과 라디오 그룹을 참조하십시오.
Width	Integer	논리 픽셀 단위의 너비이며, 0(기본값)이면 자동입니다. 스페이서에서는 최소 너비입니다.
Tag	Integer	코드에서 자유롭게 사용할 수 있는 값입니다.
DropdownMenu	TComponent	tikDropdown 항목의 팝업 메뉴입니다. TsgcVCLToolbar에서는 VCL TPopupMenu, TsgcFMXToolbar에서는 FireMonkey TPopupMenu입니다. 메뉴가 해제되면 바가 참조를 지웁니다. 드롭다운 메뉴를 참조하십시오.
Action	TBasicAction	연결된 액션입니다. Caption, Enabled, Visible, Hint, ImageIndex, Checked(토글의 Down)가 액션을 따르며, OnClick이 할당되어 있지 않으면 클릭 시 액션이 실행됩니다. Community 에디션에는 없습니다. Action 링크를 참조하십시오.
BoundsRect	TsgcUIRect (public, 읽기 전용)	마지막 레이아웃의 사각형이며, 디바이스 픽셀 단위입니다.
Shown	Boolean (public, 읽기 전용)	마지막 레이아웃에서 항목이 바에 배치되었으면 True입니다.
InOverflow	Boolean (public, 읽기 전용)	마지막 레이아웃에서 항목이 오버플로 목록으로 옮겨졌으면 True입니다.

메서드

메서드	반환값	설명
Click(aComponent: TComponent = nil)		OnClick이 할당되어 있으면 이를 실행하고, 아니면 aComponent를 ActionComponent로 하여 연결된 Action을 실행합니다. 토글을 뒤집거나 OnItemClick을 발생시키지 않으며, 그런 동작에는 바의 ClickItem을 사용하십시오.
InitiateAction		연결된 Action이 스스로 갱신하도록 합니다. Action이 없으면 아무 동작도 하지 않습니다.
IsButton	Boolean	클릭할 수 있는 종류인 tikButton, tikToggle, tikDropdown이면 True입니다.

이벤트

이벤트	형식	설명
OnClick	TNotifyEvent	항목이 활성화되었습니다. Sender는 항목입니다. 바의 OnItemClick 다음에 실행됩니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcToolbarItems
- 항목 종류
- 토글과 라디오 그룹

TsgcToolbarItems

유닛: sgcToolbar_Model
TsgcToolbarItems는 TsgcToolbarItem의 컬렉션이며, 툴바와 TsgcToolbarModel이 사용합니다. 항목이 변경되면 바가 갱신됩니다.

속성

속성	형식	설명
Items[Index: Integer]	TsgcToolbarItem	Index 위치의 항목입니다. 기본 속성이므로 Toolbar.Items[0]이 첫 번째 항목입니다.
Count	Integer	항목의 개수입니다.

메서드

메서드	반환값	설명
Add	TsgcToolbarItem	끝에 tikButton 항목을 추가하고 반환합니다.
Delete(Index: Integer)		항목을 제거합니다.
Clear		모든 항목을 제거합니다.

여러 항목을 한 번의 다시 그리기로 변경하려면, 변경 작업을 모델의 BeginUpdate와 EndUpdate로 감싸십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcToolbarItem
- TsgcToolbarModel

TsgcStatusBarPanel

유닛: sgcToolbar_Model
TsgcStatusBarPanel은 상태 표시줄의 패널 하나이며, TsgcStatusBarPanels의 TCollectionItem입니다. Text, Progress, IndicatorState는 모델의 잠금으로 보호됩니다.

속성

속성	형식	설명
Kind	TsgcStatusPanelKind	spkText(기본값), spkProgress 또는 spkIndicator입니다. 진행률과 표시기를 참조하십시오.
Text	string	텍스트 패널 또는 표시기 패널의 텍스트입니다.
Width	Integer	AutoSize도 Spring도 설정되지 않았을 때의 논리 픽셀 단위 너비입니다. 기본값 100.
AutoSize	Boolean	너비가 텍스트를 따르며, 표시기의 점도 포함합니다. 기본값 False.
Spring	Boolean	스프링 패널은 다른 패널이 쓰고 남은 너비를 나누어 가집니다. 기본값 False. 상태 패널을 참조하십시오.
Alignment	TAlignment	taLeftJustify(기본값), taCenter 또는 taRightJustify입니다.
Progress	Integer	spkProgress 패널의 값이며 0~100입니다. 기본값 0.
IndicatorState	TsgcStatusIndicatorState	spkIndicator 패널의 상태이며 sisOff(기본값), sisOk, sisWarning 또는 sisError입니다.
Hint	string	패널 위에 표시되는 힌트입니다.
Tag	Integer	코드에서 자유롭게 사용할 수 있는 값입니다.
BoundsRect	TsgcUIRect (public, 읽기 전용)	마지막 레이아웃의 사각형이며, 디바이스 픽셀 단위입니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcStatusBarPanels
- 상태 패널
- 진행률과 표시기

TsgcStatusBarPanels

유닛: sgcToolbar_Model
TsgcStatusBarPanels는 TsgcStatusBarPanel의 컬렉션이며, 상태 표시줄과 TsgcStatusBarModel이 사용합니다.

속성

속성	형식	설명
Items[Index: Integer]	TsgcStatusBarPanel	Index 위치의 패널입니다. 기본 속성이므로 StatusBar.Panels[0]이 첫 번째 패널입니다.
Count	Integer	패널의 개수입니다.

메서드

메서드	반환값	설명
Add	TsgcStatusBarPanel	끝에 spkText 패널을 추가하고 반환합니다.
Delete(Index: Integer)		패널을 제거합니다.
Clear		모든 패널을 제거합니다.

패널의 추가, 삭제, 구성은 메인 스레드에서 수행하십시오. 작업자 스레드에서는 바의 SetPanelText, SetProgress, SetIndicator를 사용하십시오. 스레드 안전 갱신을 참조하십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcStatusBarPanel
- TsgcStatusBarModel

TsgcToolbarModel

유닛: sgcToolbar_Model
TsgcToolbarModel은 툴바의 프레임워크에 독립적인 모델입니다. 모든 툴바가 하나씩 소유하며 **Model** 속성으로 접근할 수 있고, 코드에서 만든 모델은 창 없이 툴바를 그립니다. 크기는 논리 픽셀이며, Layout이 여기에 배율을 곱하고 계산되는 모든 사각형은 디바이스 픽셀 단위입니다.

속성

속성	형식	설명
Items, ShowCaptions, ButtonSize, Spacing, Padding, Theme, AccentColor		툴바의 게시된 속성입니다. TsgcVCLToolbar를 참조하십시오.
HotIndex, PressedIndex, FocusedIndex	Integer	상호작용 상태이며, 항목 인덱스, 오버플로 버튼이면 CS_TOOLBAR_HIT_OVERFLOW, 없으면 -1입니다.
HasOverflow	Boolean (읽기 전용)	마지막 레이아웃에서 항목이 오버플로 목록으로 옮겨졌으면 True입니다.
OverflowCount	Integer (읽기 전용)	오버플로 목록의 항목 수입니다.
OverflowItem[Index: Integer]	TsgcToolbarItem (읽기 전용)	오버플로 목록의 항목입니다.
OverflowRect	TsgcUIRect (읽기 전용)	마지막 레이아웃의 오버플로 버튼이며, 디바이스 픽셀 단위입니다.
Owner	TPersistent	모델을 소유하는 컴포넌트이며, 디자인 타임의 이름 경로에 사용됩니다.

메서드

메서드	반환값	설명
Layout(aWidth, aHeight: Integer; aScale: Double; aMeasure or aCanvas)		aWidth x aHeight 디바이스 픽셀 크기의 바에 대한 항목 사각형을 계산합니다. 텍스트는 aMeasure(TsgcToolbarMeasureEvent) 또는 aCanvas로 측정하며, aMeasure가 nil이면 글자당 글꼴 크기의 0.55배로 추정합니다. 스페이서는 남는 너비를 나누어 가집니다. 오버플로 메뉴가 있으면 공간에 맞지 않는 항목이 오버플로 목록으로 옮겨지고, 없으면 잘립니다.
HitTest(aX, aY: Double)	Integer	마지막 레이아웃에서 한 지점에 있는 항목이며, CS_TOOLBAR_HIT_OVERFLOW 또는 CS_TOOLBAR_HIT_NONE일 수 있습니다.
Render(aCanvas; aWidth, aHeight: Integer; aScale: Double = 1)		바를 sgcUICore 캔버스에 그리며, 이미지에는 OnGetImage를 사용합니다.
SaveToPNG, SaveToSVG(aFileName or		바를 이미지로 씁니다. 헤드리스 렌더링을 참조하십시오.

aStream; aWidth, aHeight: Integer; aScale: Double = 1)		
BeginUpdate, EndUpdate		변경을 하나의 OnChange로 묶습니다.
GetAccent	TsgcUIColor	AccentColor이며, 0이면 테마의 강조색입니다.
ShowsCaption(aItem)	Boolean	항목이 캡션을 그리면 True이며, ShowCaptions이거나 항목에 아이콘이 없는 경우입니다.
HasIcon(aItem)	Boolean	항목에 Glyph 또는 ImageIndex가 있으면 True입니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	모델이 변경되었습니다. 바는 이 이벤트로 다시 그립니다.
OnGetImage	TsgcToolbarImageEvent	Render와 헤드리스 출력을 위해 ImageIndex의 이미지를 TsgcUIImage로 반환합니다.

예제

테스트나 서버처럼 컨트롤 없이 Layout과 HitTest를 사용합니다.

관련 항목

- TsgcVCLToolbar
- 헤드리스 렌더링
- 아키텍처

TsgcStatusBarModel

유닛: sgcToolbar_Model
TsgcStatusBarModel은 상태 표시줄의 프레임워크에 독립적인 모델입니다. 모든 상태 표시줄이 하나씩 소유하며 **Model** 속성으로 접근할 수 있습니다. 세 개의 setter는 스레드 안전합니다. 모델의 잠금을 획득하고, 값을 변경하고, 호출한 스레드에서 OnChange를 발생시키며, 값이 실제로 바뀐 경우에만 그렇게 합니다. 그 밖의 모든 속성은 메인 스레드용입니다.

속성

속성	형식	설명
Panels, Theme, AccentColor, SizeGrip		상태 표시줄의 게시된 속성입니다. TsgcVCLStatusBar를 참조하십시오.
GripRect	TsgcUIRect (읽기 전용)	마지막 레이아웃의 크기 조절 핸들이며 디바이스 픽셀 단위이고, SizeGrip이 False이면 비어 있습니다.
Owner	TPersistent	모델을 소유하는 컴포넌트이며, 디자인 타임의 이름 경로에 사용됩니다.

메서드

메서드	반환값	설명
SetPanelText, SetProgress, SetIndicator		스레드 안전한 setter입니다. 잘못된 인덱스는 무시됩니다.
Layout(aWidth, aHeight: Integer; aScale: Double; aMeasure or aCanvas)		패널 사각형을 계산합니다. 고정 너비를 먼저 배정하고, 다음으로 AutoSize 패널을 텍스트에 맞추며, 남은 너비를 Spring 패널이 0 미만이 되지 않게 나누어 가집니다. SizeGrip이 True이면 오른쪽에 크기 조절 핸들 공간이 확보됩니다.
HitTest(aX, aY: Double)	Integer	마지막 레이아웃에서 한 지점에 있는 패널이며, 없으면 -1입니다.
Render, SaveToPNG, SaveToSVG		TsgcToolbarModel과 마찬가지로 바를 캔버스에 그리거나 이미지로 씁니다.
GetIndicatorColor(aState)	TsgcUIColor	현재 테마에서 표시기 상태의 색상입니다.
GetAccent	TsgcUIColor	AccentColor이며, 0이면 테마의 강조색입니다.
BeginUpdate, EndUpdate		변경을 하나의 OnChange로 묶습니다.
Lock, Unlock		Text, Progress, IndicatorState를 보호하는 잠금입니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	모델이 변경되었습니다. 스레드 안전 setter 다음에는 호출한 스레드에서 발생하며, 바가 다시 그리기를 메인 스레드로 마샬링합니다.

예제

관련 항목

- TsgcVCLStatusBar
- 스레드 안전 갱신
- 헤드리스 렌더링

항목 종류

Kind 속성은 TsgcToolbarItem이 그려지는 방식과 반응하는 방식을 결정합니다. 항목은 컬렉션의 순서대로 왼쪽에서 오른쪽으로 배치되며, 항목 사이에는 Spacing 논리 픽셀의 간격이 있습니다.

Kind	동작
tikButton	기본값입니다. 클릭하면 바의 OnItemClick이 발생하고, 이어서 항목의 OnClick 또는 Action이 실행됩니다.
tikToggle	눌린 상태를 유지하는 버튼입니다. 클릭하면 OnItemClick이 발생하기 전에 Down이 뒤집힙니다. GroupIndex가 0보다 크면 라디오 버튼이 됩니다. 토글과 라디오 그룹을 참조하십시오.
tikDropdown	작은 화살표가 있는 버튼으로, 버튼 아래에 DropdownMenu를 표시합니다. 드롭다운 메뉴를 참조하십시오.
tikSeparator	가는 세로선입니다. 클릭되지 않습니다.
tikSpacer	바의 남는 너비를 다른 스페이서와 나누어 차지하는 빈 항목입니다. 뒤에 오는 항목을 오른쪽으로 밀어냅니다. Width는 최소 너비입니다.
tikLabel	버튼이 없는 캡션이며, 예를 들어 편집 컨트롤 앞에 사용합니다. 클릭되지 않습니다.

버튼은 아이콘(ImageIndex의 이미지 또는 Glyph 텍스트)과 Caption을 그립니다. ShowCaptions가 False이면 아이콘만 그려지고 캡션은 힌트가 됩니다. 아이콘이 없는 항목은 항상 캡션을 그립니다. Width가 0보다 크면 항목의 너비가 논리 픽셀 단위로 고정됩니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcToolbarItem
- 이미지와 글리프
- 오버플로

토글과 라디오 그룹

tikToggle 항목에는 강조색 틴트로 그려지는 **Down** 상태가 있습니다. 클릭하면 Down이 뒤집힌 다음 OnItemClick이 발생하므로, 핸들러는 새 상태를 읽게 됩니다.
0보다 큰 같은 **GroupIndex**를 공유하는 토글은 라디오 그룹을 이룹니다. 클릭이든 코드든, 그중 하나의 Down을 설정하면 나머지는 해제됩니다. 이미 눌려 있는 라디오 토글을 클릭해도 눌린 채로 남으므로, 그룹의 한 항목은 항상 선택된 상태를 유지합니다. Action에 연결된 토글은 액션의 Checked를 따릅니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- 항목 종류
- Action 링크
- TsgcToolbarItem

드롭다운 메뉴

tikDropdown 항목은 캡션 뒤에 작은 화살표를 그리고, **DropdownMenu** 속성의 팝업 메뉴를 버튼 아래에 표시합니다. TsgcVCLToolbar에서는 VCL TPopupMenu, TsgcFMXToolbar에서는 FireMonkey TPopupMenu입니다. 항목이 프레임워크에 독립적으로 유지되도록 이 속성은 TComponent로 선언되어 있습니다. 메뉴가 해제되면 바가 참조를 지웁니다.
VCL에서는 버튼을 누르는 순간 메뉴가 열립니다. OnItemClick과 항목의 OnClick이 먼저 발생하므로, 핸들러가 메뉴가 표시되기 전에 메뉴를 채우거나 체크할 수 있습니다. 오버플로 목록에 있는 드롭다운 항목은 오버플로 버튼 아래에 메뉴를 표시합니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- 항목 종류
- 오버플로
- 문제 해결

오버플로

항목이 바의 너비에 맞지 않으면 레이아웃이 마지막 항목들을 오버플로 목록으로 옮기고, 오른쪽 끝에 이중 쉐브론으로 그려진 오버플로 버튼을 확보합니다. 이 버튼을 누르면 바의 **OverflowMenu**가 표시됩니다. 이는 목록의 항목마다 하나의 메뉴 항목을 가진 채 실행 시에 만들어지는 팝업 메뉴입니다. 메뉴 항목을 선택하면 클릭과 똑같이 해당 항목이 활성화됩니다. 토글이 뒤집히고, OnItemClick이 발생하며, 드롭다운은 메뉴를 표시합니다.
마지막 레이아웃의 오버플로 목록은 모델을 통해 접근할 수 있습니다. **HasOverflow**, **OverflowCount**, **OverflowItem**[i], 그리고 모든 항목의 **InOverflow** 속성입니다. ItemAtPos는 오버플로 버튼 위에서 CS_TOOLBAR_HIT_OVERFLOW를 반환합니다.

Community 에디션

오버플로 메뉴는 Community 에디션(SGC_TOOLBAR_ADVANCED)에 포함되지 않습니다. 이 에디션에서는 공간에 맞지 않는 항목이 오른쪽 가장자리에서 잘리며 OverflowMenu 속성이 존재하지 않습니다. 에디션을 참조하십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcToolbarModel
- 드롭다운 메뉴
- 에디션

Action 링크

Action 속성은 모델이 어느 프레임워크에도 의존하지 않은 채, 항목을 임의의 TBasicAction, 즉 VCL 또는 FireMonkey TAction에 연결합니다. 그러면 항목은 액션의 **Caption**, **Enabled**, **Visible**, **Hint**, **ImageIndex**, **Checked**를 따르며, Checked는 tikToggle 항목의 Down을 제어합니다. FireMonkey TAction은 Text를 Caption으로 제공합니다.
항목에 OnClick 핸들러가 없으면 클릭 시 바를 ActionComponent로 하여 액션이 실행됩니다. 바는 유휴 시간에 InitiateAction을 호출하므로 연결된 모든 액션의 OnUpdate가 실행되고, 비활성 액션은 해당 버튼을 비활성화합니다. Delphi XE3부터는 액션이 변경 사항을 항목에 푸시합니다. XE3 이전에는 항목이 유휴 시간에 액션의 속성을 복사합니다.

Community 에디션

Action 링크는 Community 에디션(SGC_TOOLBAR_ADVANCED)에 포함되지 않으며, Action 속성이 존재하지 않습니다. 대신 항목의 OnClick 또는 바의 OnItemClick을 사용하십시오. 에디션을 참조하십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcToolbarItem
- 토글과 라디오 그룹
- 문제 해결

이미지와 글리프

클릭할 수 있는 모든 항목에는 16 논리 픽셀 크기의 아이콘 슬롯이 있습니다. 슬롯에는 바의 **Images** 목록에서 **ImageIndex** 위치의 이미지가 표시되고, 항목에 이미지가 없으면 **Glyph**가 표시됩니다. Glyph는 문자나 기호 같은 짧은 텍스트입니다. 둘 다 없는 항목은 캡션만 표시합니다.

- VCL에서 Images는 TCustomImageList입니다. 모든 이미지는 알파를 복원하여 한 번 스트레이트 ARGB로 변환되며, 이미지 목록이 바뀔 때까지 캐시됩니다.
- FMX에서 Images는 FMX.ImgList의 FireMonkey TCustomImageList이며 Delphi XE8 이상이 필요합니다. 이미지는 프레임의 디바이스 크기로 가져옵니다. 이전 버전에서는 항목이 Glyph를 그립니다.
- 헤드리스 출력은 모델의 OnGetImage 이벤트가 이미지를 반환하지 않는 한 Glyph를 그립니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- 항목 종류
- 헤드리스 렌더링
- 플랫폼

상태 패널

상태 표시줄은 Panels를 왼쪽에서 오른쪽으로 세 단계에 걸쳐 배치합니다.

- AutoSize도 Spring도 없는 **고정** 패널은 논리 픽셀 단위의 **Width**(기본값 100)를 차지합니다.
- **AutoSize** 패널은 텍스트의 너비에 표시기의 점을 더한 너비를 차지합니다.
- **Spring** 패널은 남은 너비를 0 미만이 되지 않게 나누어 가집니다. 주 메시지에는 스프링 패널 하나를 사용하십시오.

패널 사이는 가는 선으로 구분됩니다. 패널의 텍스트는 **Alignment**로 정렬되며, 마우스가 패널 위에 머물고 ShowHint가 True이면 **Hint**가 표시됩니다. OnPanelClick은 패널에 대한 클릭을 알려 줍니다. SizeGrip이 True이면 오른쪽 끝에 핸들 공간이 확보됩니다. 크기 조절 핸들을 참조하십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcStatusBarPanel
- 진행률과 표시기
- 스레드 안전 갱신

진행률과 표시기

텍스트 패널(spkText) 외에도 패널은 진행률 표시줄이나 상태 표시기가 될 수 있습니다.

Kind	그리는 내용
spkProgress	**Progress** 퍼센트(0~100)까지 채워진 막대이며, 상태 표시줄의 AccentColor 또는 테마의 강조색으로 그려집니다.
spkIndicator	색상 점과 그 뒤에 이어지는 패널의 **Text**입니다. 색상은 **IndicatorState**에서 정해지며, 연결이나 데이터 피드의 상태를 나타내기에 적합합니다.

IndicatorState	라이트 테마	다크 테마
sisOff	테마의 흐린 텍스트 색상	테마의 흐린 텍스트 색상
sisOk	녹색 $FF1A7F37	녹색 $FF3FB950
sisWarning	황색 $FFBF8700	황색 $FFD29922
sisError	빨간색 $FFCF222E	빨간색 $FFF85149

TsgcStatusBarModel의 GetIndicatorColor는 현재 테마에서 상태의 색상을 반환합니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- 상태 패널
- 스레드 안전 갱신
- 테마

스레드 안전 갱신

실시간 애플리케이션은 다운로드, 데이터 피드, 긴 계산 같은 작업자 스레드에서 상태 표시줄을 갱신합니다. 상태 표시줄의 **SetPanelText**, **SetProgress**, **SetIndicator**는 Synchronize나 Queue 없이 어떤 스레드에서든 호출할 수 있습니다.

- 모델은 패널의 Text, Progress, IndicatorState를 잠금으로 보호합니다.
- setter는 호출한 스레드에서 모델의 OnChange를 발생시키며, 값이 실제로 바뀐 경우에만 그렇게 하므로 같은 값을 반복해도 비용이 들지 않습니다.
- 바는 다시 그리기 요청으로 응답합니다. 작업자 스레드에서 VCL 바는 자신의 창에 메시지를 게시하기만 하고 FMX 바는 다시 그리기를 메인 스레드에 큐잉하므로, 프레임워크가 메인 스레드 밖에서 건드려지는 일이 없습니다.
- 범위를 벗어난 인덱스는 무시되므로, 패널보다 오래 살아남은 스레드도 예외를 발생시키지 않습니다.

그 밖의 모든 속성과 패널의 추가, 삭제, 구성은 메인 스레드에서 수행해야 합니다. 상태 표시줄을 소유한 폼이 소멸되기 전에 작업자가 중지되도록 하십시오.

예제

LiveStatus 데모처럼 50개 청크의 다운로드를 시뮬레이션하는 작업자 스레드입니다. 패널 0은 스프링 텍스트 패널, 패널 1은 진행률 패널, 패널 2는 표시기입니다.

관련 항목

- TsgcStatusBarModel
- 진행률과 표시기
- 데모

크기 조절 핸들

SizeGrip이 True(기본값)이면 상태 표시줄은 오른쪽 끝에 작은 영역을 확보하고 그곳에 핸들 점을 그립니다. **GripRect**는 그 사각형을 반환하며, SizeGrip이 False이면 비어 있습니다.

- VCL에서 TsgcVCLStatusBar는 핸들의 비클라이언트 히트 테스트에 응답하므로, 핸들을 드래그하면 부모 폼의 크기가 바뀝니다. 폼에 크기 조절 가능한 테두리가 있고 최대화되지 않은 경우에만 그렇게 동작하며, 디자인 타임에는 동작하지 않습니다.
- FMX에서 TsgcFMXStatusBar는 핸들을 그리기만 합니다. FireMonkey에는 비클라이언트 히트 테스트가 없으므로 핸들이 폼의 크기를 바꾸지 않습니다. 폼의 크기를 조절할 수 없다면 SizeGrip을 False로 설정하십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- 상태 패널
- TsgcVCLStatusBar

키보드

VCL에서 **TabStop**이 True이거나 FMX에서 **CanFocus**가 True이면 Tab 키로 툴바에 이동할 수 있습니다. 둘 다 기본값은 False이므로, 툴바가 아래쪽 편집기의 포커스를 빼앗지 않습니다. 툴바에 포커스가 있으면 포커스된 버튼에 포커스 링이 표시됩니다.

키	동작
Left, Right	이전 또는 다음의 활성화된 버튼, 토글, 드롭다운으로 이동합니다.
Home, End	첫 번째 또는 마지막 항목으로 이동합니다.
Enter, Space	클릭한 것처럼 포커스된 항목을 활성화합니다.

ClickItem은 코드에서 항목을 활성화하며, 예를 들어 폼의 키보드 단축키에서 호출할 수 있습니다. 상태 표시줄에는 키보드 인터페이스가 없습니다.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

- TsgcVCLToolbar
- TsgcFMXToolbar

테마

바는 제품군의 다른 제품과 같은 sgcUICore 테마 토큰에서 색상을 가져옵니다. **Theme**은 utLight(기본값), utDark 또는 utCustom이며, utCustom은 라이트 색상을 사용합니다. 각 바는 자체 Theme을 가지므로 툴바와 상태 표시줄에 각각 설정하십시오.

- 배경은 Surface 색상이며, 툴바 아래와 상태 표시줄 위에 Border 선이 있습니다.
- 핫 및 눌린 버튼에는 오버레이가, 눌린 토글에는 강조색 틴트가, 포커스된 버튼에는 포커스 링이, 비활성 항목에는 흐린 텍스트 색상이 적용됩니다.
- $AARRGGBB 형식의 **AccentColor**는 테마의 강조색을 대체합니다. 툴바에서는 포커스 링과 눌림 틴트, 상태 표시줄에서는 진행률 표시줄에 해당합니다. 0이면 테마의 강조색을 사용합니다.
- 표시기 색상은 테마를 따릅니다. 진행률과 표시기를 참조하십시오.

예제

이 예제는 **Toolbar**(TsgcVCLToolbar), **StatusBar**(TsgcVCLStatusBar), **Memo**(TMemo)가 있는 VCL 폼의 메서드입니다.

관련 항목

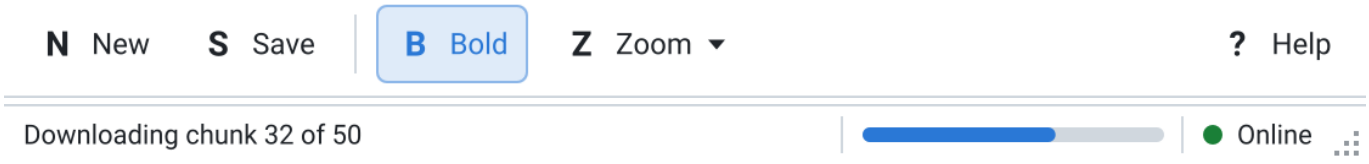
- 진행률과 표시기
- 아키텍처

헤드리스 렌더링

sgcToolbar_Render의 렌더러는 창도 VCL이나 FMX도 없이 TsgcToolbarModel 또는 TsgcStatusBarModel을 그립니다. 따라서 콘솔 애플리케이션, 서비스, 테스트, 웹 서버가 같은 바를 이미지로 만들 수 있으며, 애플리케이션 문서나 시각적 테스트에 활용할 수 있습니다. 이 유닛은 Linux64를 포함한 모든 플랫폼에서 빌드됩니다.

루틴	설명
Model.SaveToPNG(aFileName or aStream, aWidth, aHeight, aScale = 1)	PNG 파일 또는 스트림을 씁니다. aWidth와 aHeight는 논리 픽셀이고 aScale이 여기에 곱해집니다. 560 x 44에 배율 2이면 1120 x 88 이미지가 됩니다.
Model.SaveToSVG(aFileName or aStream, aWidth, aHeight, aScale = 1)	같은 바의 SVG 파일 또는 스트림을 씁니다.
Model.Render(aCanvas, aWidth, aHeight, aScale = 1)	바를 임의의 sgcUICore 캔버스에 디바이스 픽셀 단위로 그립니다.
RenderToolbar(aModel, aCanvas, aWidth, aHeight, aScale, aImages)	sgcToolbar_Render의 렌더러입니다. aImages는 ImageIndex의 이미지를 반환하며, nil이면 Glyph를 그립니다.
RenderStatusBar(aModel, aCanvas, aWidth, aHeight, aScale)	상태 표시줄의 렌더러입니다.

툴바의 높이는 ButtonSize에 Padding의 두 배를 더한 값이며, 기본값에서는 44 논리 픽셀입니다. 컨트롤의 모델은 **Model** 속성으로 접근할 수 있으므로, **Toolbar.Model.SaveToPNG(...)**는 컨트롤이 표시하는 내용을 임의의 크기로 저장합니다.



예제

관련 항목

- TsgcToolbarModel
- TsgcStatusBarModel
- 플랫폼

.NET 개요

sgcToolbar .NET은 Delphi sgcToolbar 라이브러리를 감싼 래퍼가 아니라 C#으로 이식한 관리형 라이브러리입니다. C# 클래스는 **esegece.sgcToolbar** 네임스페이스에 있으며 Delphi 이름을 그대로 유지하므로 **TsgcToolbarModel**, **TsgcToolbarItem**, **TsgcToolbarItemKind**, **TsgcStatusBarModel**, **TsgcStatusBarPanel**이 두 언어에서 똑같이 읽히고, 각 호스트의 툴바와 상태 표시줄도 Delphi 명명 패턴을 따릅니다. 이들은 **sgcUICore**의 .NET 이식판으로 그려지며, eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 라이트 및 다크 테마를 사용하므로 바는 WinForms, WPF, Avalonia, 헤드리스 PNG 및 SVG 출력에서 모두 똑같이 보입니다.
툴바는 버튼, 토글, 라디오 그룹, 드롭다운 버튼, 구분선, 스페이서, 레이블을 담고, 들어가지 않는 항목은 오버플로 메뉴로 옮깁니다. 상태 표시줄은 고정, 자동 또는 스프링 너비의 텍스트, 진행률, 표시기 패널을 담으며, 패널은 어느 스레드에서나 갱신할 수 있습니다.

패키지

이식판은 다섯 개의 NuGet 패키지로 나뉘어 있으므로, 프로젝트는 사용하는 호스트만 가져옵니다.

PackageId	포함 내용	대상 프레임워크
esegece.sgcToolbar	모델(**TsgcToolbarModel**, **TsgcStatusBarModel**), 항목과 패널, 항목의 **Command** 링크, 렌더러 **sgcToolbar_Render**, 헤드리스 SVG 출력. 네이티브 종속성은 없습니다. esegece.sgcUICore에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcToolbar.Rendering	헤드리스 PNG 출력, **sgcToolbar_Raster**의 **SaveToPNG** 확장 메서드. esegece.sgcUICore.Rendering의 SkiaSharp 캔버스에 그리며 Windows, Linux, macOS에서 동작합니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcToolbar.WinForms	**TsgcWinFormsToolbar**와 **TsgcWinFormsStatusBar**.	net472, net8.0-windows, net9.0-windows
esegece.sgcToolbar.Wpf	**TsgcWpfToolbar**와 **TsgcWpfStatusBar**.	net472, net8.0-windows, net9.0-windows
esegece.sgcToolbar.Avalonia	**TsgcAvaloniaToolbar**와 **TsgcAvaloniaStatusBar**. Avalonia 11을 사용하며 Windows, Linux, macOS에서 동작합니다.	net8.0, net9.0

세 호스트 패키지는 esegece.sgcToolbar.Rendering에 종속되고, 이 패키지는 esegece.sgcToolbar에 종속되므로 `PackageReference` 한 줄이면 호스트에 필요한 모든 것이 들어옵니다. NuGet 설치를 참조하십시오.

에디션

에디션	패키지 ID	포함 사항
Registered	esegece.sgcToolbar, esegece.sgcToolbar.Rendering, esegece.sgcToolbar.WinForms, esegece.sgcToolbar.Wpf, esegece.sgcToolbar.Avalonia	전부: 세 호스트의 툴바와 상태 표시줄, 모든 항목 및 패널 종류, 라디오 그룹, 드롭다운 메뉴, 오버플로 메뉴, 항목의 Command 링크, 이미지, 힌트, 키보드 탐색, 스레드 안전 상태 세터, 크기 조절 핸들, 테마, 헤드리스 PNG 및 SVG 출력.

Community	같은 ID에 .Community 접미사를 붙인 것으로, 예를 들어 esegece.sgcToolbar.WinForms.Community입니다. sgcUICore의 Community 패키지에 종속됩니다.	Registered와 같은 내용에서 오버플로 메뉴와 항목의 **Command** 및 **CommandParameter** 속성을 뺀 것으로, Delphi Community 에디션과 같습니다. 들어가지 않는 항목은 오른쪽 끝에서 잘리며, 항목은 OnClick을 실행합니다.

Community 에디션은 연간 총매출이 1,000,000 EUR 미만이고 개발자가 5명 이하인 개인과 조직이라면 상업적 용도와 비상업적 용도 모두 무료입니다. 두 기준 중 하나라도 같거나 넘는 조직은 Registered 라이선스가 필요합니다. .NET 이식판에는 Trial 에디션이 없습니다.

호스트

Host	Delphi 대응	플랫폼
WinForms	VCL (TsgcVCLToolbar, TsgcVCLStatusBar)	Windows, .NET Framework 4.7.2, .NET 8 및 9.
WPF	FMX (TsgcFMXToolbar, TsgcFMXStatusBar)	Windows, .NET Framework 4.7.2, .NET 8 및 9. 바는 XAML로 선언하고 항목과 패널은 코드에서 추가합니다.
Avalonia	FMX (TsgcFMXToolbar, TsgcFMXStatusBar)	Windows, Linux, macOS, .NET 8 및 9. 마우스, 터치, 펜을 지원합니다. 바는 XAML로 선언하고 항목과 패널은 코드에서 추가합니다.
헤드리스	모델의 Render 및 SaveToSVG 메서드	대상 프레임워크의 모든 플랫폼. SVG는 esegece.sgcToolbar, PNG는 esegece.sgcToolbar.Rendering으로 출력합니다.

모든 툴바는 같은 API를 가집니다. Items, ShowCaptions, ButtonSize, Spacing, Padding, Theme, AccentColor, Images, OverflowMenu, ItemAtPos, ItemRect, OverflowRect, ClickItem, OnItemClick입니다. 모든 상태 표시줄에는 Panels, SizeGrip, Theme, AccentColor, PanelAtPos, PanelRect, GripRect, 스레드 안전한 SetPanelText, SetProgress, SetIndicator, 그리고 OnPanelClick이 있습니다. 클래스 이름은 .NET 클래스를, Delphi와 달라지는 점은 Delphi와 .NET의 차이점을 참조하십시오.

관련 항목

- NuGet 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET 데모
- 플랫폼

NuGet 설치

패키지는 nuget.org에 있습니다. 호스트 패키지를 추가하면 코어 패키지와 렌더링 패키지가 종속성으로 함께 들어옵니다.

Registered 에디션

```
dotnet add package esegece.sgcToolbar.WinForms
dotnet add package esegece.sgcToolbar.Wpf
dotnet add package esegece.sgcToolbar.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcToolbar.WinForms.Community
dotnet add package esegece.sgcToolbar.Wpf.Community
dotnet add package esegece.sgcToolbar.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcToolbar.Wpf" Version="2026.10.0" />
</ItemGroup>
```

프로젝트에서는 Registered 패키지나 Community 패키지 중 하나만 참조하고 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있기 때문입니다. 항목의 **Command**나 툴바의 **OverflowMenu** 속성을 사용하는 코드는 Community 패키지에서는 컴파일되지 않습니다. 하나의 프로젝트에서 두 에디션을 모두 빌드한다면 `#if !SGC_COMMUNITY`로 감싸십시오.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`가 지정된 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`가 지정된 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서는 `xmlns:sgc="clr-namespace:esegece.sgcToolbar;assembly=esegece.sgcToolbar.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: Avalonia 11.3의 **Avalonia.Desktop** 및 **Avalonia.Themes.Fluent** 패키지를 사용하는 `net8.0` 또는 `net9.0` 프로젝트로, 다른 Avalonia 데스크톱 애플리케이션과 같습니다. XAML에서는 `xmlns:sgc="using:esegece.sgcToolbar"`로 네임스페이스를 매핑합니다.
- 헤드리스: SVG에는 esegece.sgcToolbar만으로 충분합니다. PNG에는 esegece.sgcToolbar.Rendering을 추가하며, 이 패키지는 SkiaSharp와 Windows 및 Linux용 네이티브 자산을 가져옵니다.

C#에서는 모든 호스트에 `using esegece.sgcToolbar;`가 필요하며, **TsgcUITheme**과 그 밖의 테마 형식에는 `using esegece.sgcUICore;`도 필요합니다.

관련 항목

- .NET 개요
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 데모

빠른 시작 WinForms

TsgcWinFormsToolbar 와 **TsgcWinFormsStatusBar** 는 WinForms 바 이 며, Delphi 의 TsgcVCLToolbar 및 TsgcVCLStatusBar에 해당합니다. 각각 모델을 하나씩 소유하고, 그 속성을 노출하며, sgcUICore를 통해 그립니다. 패키지는 **esegece.sgcToolbar.WinForms**입니다.

단계

- WinForms 프로젝트를 만들고 esegece.sgcToolbar.WinForms 패키지를 추가합니다.
- **TsgcWinFormsToolbar**를 만듭니다. 기본적으로 위쪽에 도킹되며 높이는 **ButtonSize**와 **Padding**을 따릅니다.
- `Items.Add()`로 항목을 추가하고 **Kind**, **Caption**, **Glyph** 또는 **ImageIndex**, **Hint**를 설정합니다. 같은 **GroupIndex**를 가진 **tikToggle**은 라디오 그룹이 되고, **tikDropdown**은 그 **DropdownMenu**의 **ContextMenuStrip**을 표시합니다.
- 아래쪽에 도킹되는 **TsgcWinFormsStatusBar**를 만들고 `Panels.Add()`로 패널을 추가합니다.
- 모든 클릭에 반응하려면 **OnItemClick**을, 항목별로 반응하려면 각 항목의 **OnClick**을 할당합니다.

예제

툴바와 상태 표시줄이 있는 WinForms 편집기의 Program.cs 전체입니다. 버튼 두 개, Bold 토글, Left와 Center 라디오 그룹, Zoom 드롭다운, 그리고 스프링 텍스트 패널, 진행률 패널, 표시기가 있습니다. esegece.sgcToolbar.WinForms를 참조하는 `net8.0-windows` 프로젝트에서 그대로 컴파일됩니다.

```
using System;
using System.Windows.Forms;
using esegece.sgcToolbar;

namespace ToolbarQuickStartWinForms
{
    internal class MainForm : Form
    {
        private readonly TsgcWinFormsToolbar FToolbar;
        private readonly TsgcWinFormsStatusBar FStatusBar;
        private readonly TextBox FEditor;

        public MainForm()
        {
            Text = "sgcToolbar WinForms";
            Width = 720;
            Height = 420;

            FEditor = new TextBox();
            FEditor.Multiline = true;
            FEditor.Dock = DockStyle.Fill;
            FEditor.TextChanged += EditorTextChanged;

            // Dock is Top by default, the height follows ButtonSize and Padding
            FToolbar = new TsgcWinFormsToolbar();
            FToolbar.ShowCaptions = true;
            FToolbar.OnItemClick = ToolbarItemClick;
            DoAddButton(TsgcToolbarItemKind.tikButton, "New", "N", "Create a new document");
            DoAddButton(TsgcToolbarItemKind.tikButton, "Save", "S", "Save the document");
            DoAddButton(TsgcToolbarItemKind.tikSeparator, "", "", "");
            DoAddButton(TsgcToolbarItemKind.tikToggle, "Bold", "B", "Bold text");
            // a radio group: the toggles with the same GroupIndex release each other
            DoAddButton(TsgcToolbarItemKind.tikToggle, "Left", "L", "Align left").GroupIndex = 1;
            DoAddButton(TsgcToolbarItemKind.tikToggle, "Center", "C", "Center").GroupIndex = 1;
            FToolbar.Items[4].Down = true;
            // a dropdown item shows a ContextMenuStrip below the button
            ContextMenuStrip oZoom = new ContextMenuStrip();
            foreach (string vZoom in new string[] { "50%", "100%", "200%" })
```

```
            oZoom.Items.Add(vZoom, null, ZoomClick);
        DoAddButton(TsgcToolbarItemKind.tikDropdown, "Zoom", "Z", "Zoom level").DropdownMenu = oZoom;

        // Dock is Bottom by default: a spring text panel, a progress and an indicator
        FStatusBar = new TsgcWinFormsStatusBar();
        TsgcStatusBarPanel oPanel = FStatusBar.Panels.Add();
        oPanel.Spring = true;
        oPanel.Text = "Ready";
        oPanel = FStatusBar.Panels.Add();
        oPanel.Kind = TsgcStatusPanelKind.spkProgress;
        oPanel = FStatusBar.Panels.Add();
        oPanel.Kind = TsgcStatusPanelKind.spkIndicator;
        oPanel.AutoSize = true;
        oPanel.Text = "Saved";
        oPanel.IndicatorState = TsgcStatusIndicatorState.sisOk;

        Controls.Add(FEditor);
        Controls.Add(FToolbar);
        Controls.Add(FStatusBar);
    }

    private TsgcToolbarItem DoAddButton(TsgcToolbarItemKind aKind, string aCaption, string aGlyph,
        string aHint)
    {
        TsgcToolbarItem oItem = FToolbar.Items.Add();
        oItem.Kind = aKind;
        oItem.Caption = aCaption;
        oItem.Glyph = aGlyph;
        oItem.Hint = aHint;
        return oItem;
    }

    private void ToolbarItemClick(object Sender, TsgcToolbarItem aItem)
    {
        if (aItem.Caption == "New")
            FEditor.Clear();
        else if (aItem.Caption == "Save")
            FStatusBar.SetIndicator(2, TsgcStatusIndicatorState.sisOk);
        else if (aItem.Caption == "Bold")
            FEditor.Font = new System.Drawing.Font(FEditor.Font,
                aItem.Down ? System.Drawing.FontStyle.Bold : System.Drawing.FontStyle.Regular);
        FStatusBar.SetPanelText(0, aItem.Caption + " clicked");
    }

    private void ZoomClick(object sender, EventArgs e)
    {
        FStatusBar.SetPanelText(0, "Zoom " + ((ToolStripItem)sender).Text);
    }

    private void EditorTextChanged(object sender, EventArgs e)
    {
        FStatusBar.SetIndicator(2, TsgcStatusIndicatorState.sisWarning);
        FStatusBar.SetProgress(1, Math.Min(100, FEditor.TextLength));
    }
}

internal static class Program
{
    [STAThread]
    private static void Main()
    {
        Application.EnableVisualStyles();
        Application.Run(new MainForm());
    }
}
```

바는 도구 상자에도 있습니다. 항목과 패널은 디자이너가 직렬화하지 않으므로 항상 코드에서 추가합니다. Delphi와 .NET의 차이점을 참조하십시오. 화살표 키로 버튼 사이에서 포커스를 이동하려면 **TabStop**을 True로 설정합니다.

관련 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- 항목 종류

빠른 시작 WPF

TsgcWpfToolbar와 **TsgcWpfStatusBar**는 WPF 바입니다. 다른 컨트롤처럼 XAML에서 선언하고, 항목과 패널은 코드에서 추가합니다. 패키지는 **esegece.sgcToolbar.Wpf**입니다.

단계

- WPF 프로젝트(`dotnet new wpf`)를 만들고 esegece.sgcToolbar.Wpf 패키지를 추가합니다.
- `esegece.sgcToolbar` 네임스페이스를 매핑하고 **TsgcWpfToolbar**와 **TsgcWpfStatusBar**를 선언합니다. **ShowCaptions**, **ButtonSize**, **SizeGrip** 등 단순한 속성은 XAML 특성으로 설정할 수 있습니다.
- 코드 비하인드에서 **ImageSource** 목록인 **Images**를 할당하고, 항목과 패널을 추가하고, **OnItemClick**을 할당합니다. 이것은 WPF 이벤트가 아니라 대리자이므로 XAML 특성으로 설정할 수 없습니다.

예제

MainWindow.xaml은 툴바, 상태 표시줄, 텍스트 상자를 선언합니다. 프로젝트의 App.xaml은 WPF 템플릿의 것이며 StartupUri가 MainWindow.xaml로 설정되어 있습니다.

```
<Window x:Class="ToolbarQuickStartWpf.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcToolbar;assembly=esegece.sgcToolbar.Wpf"
        Title="sgcToolbar WPF" Width="720" Height="420">
    <DockPanel>
        <!-- the height of both bars follows their properties, in device independent units -->
        <sgc:TsgcWpfToolbar x:Name="Toolbar" DockPanel.Dock="Top" ShowCaptions="True" />
        <sgc:TsgcWpfStatusBar x:Name="StatusBar" DockPanel.Dock="Bottom" SizeGrip="True" />
        <TextBox x:Name="Editor" AcceptsReturn="True" TextChanged="Editor_TextChanged" />
    </DockPanel>
</Window>
```

MainWindow.xaml.cs는 **DrawingImage** 벡터 아이콘 세 개를 만들고, 버튼, Wrap 토글, WPF **ContextMenu**가 있는 Zoom 드롭다운, 그리고 텍스트 패널과 표시기를 추가합니다.

```
using System.Collections.Generic;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using esegece.sgcToolbar;

namespace ToolbarQuickStartWpf
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            // the images, the items, the panels and the events are set in code
            Toolbar.Images = new List<ImageSource>
            {
                DoIcon(Colors.SteelBlue, "M3,1 H10 L13,4 V15 H3 Z"),
                DoIcon(Colors.SeaGreen, "M2,2 H14 V14 H2 Z M5,2 V6 H11 V2"),
                DoIcon(Colors.DarkOrange, "M7,1 H9 V7 H15 V9 H9 V15 H7 V9 H1 V7 H7 Z")
            };
            DoAddButton(TsgcToolbarItemKind.tikButton, "New", 0);
            DoAddButton(TsgcToolbarItemKind.tikButton, "Save", 1);
            DoAddButton(TsgcToolbarItemKind.tikToggle, "Wrap", 2).Hint = "Wrap long lines";
            // a dropdown item shows a WPF ContextMenu below the button
            ContextMenu oMenu = new ContextMenu();
            foreach (string vZoom in new string[] { "50%", "100%", "200%" })
            {
```

```
            MenuItem oEntry = new MenuItem();
            oEntry.Header = vZoom;
            oEntry.Click += ZoomClick;
            oMenu.Items.Add(oEntry);
        }
        DoAddButton(TsgcToolbarItemKind.tikDropdown, "Zoom", -1).DropdownMenu = oMenu;
        Toolbar.OnItemClick = ToolbarItemClick;

        TsgcStatusBarPanel oPanel = StatusBar.Panels.Add();
        oPanel.Spring = true;
        oPanel.Text = "Ready";
        oPanel = StatusBar.Panels.Add();
        oPanel.Kind = TsgcStatusPanelKind.spkIndicator;
        oPanel.AutoSize = true;
        oPanel.Text = "Saved";
        oPanel.IndicatorState = TsgcStatusIndicatorState.sisOk;
    }

    private static ImageSource DoIcon(Color aColor, string aGeometry)
    {
        // a vector icon, drawn at 16 x 16 device independent units
        GeometryDrawing oDrawing = new GeometryDrawing(new SolidColorBrush(aColor), null,
            Geometry.Parse(aGeometry));
        DrawingImage oImage = new DrawingImage(oDrawing);
        oImage.Freeze();
        return oImage;
    }

    private TsgcToolbarItem DoAddButton(TsgcToolbarItemKind aKind, string aCaption, int aImageIndex)
    {
        TsgcToolbarItem oItem = Toolbar.Items.Add();
        oItem.Kind = aKind;
        oItem.Caption = aCaption;
        oItem.ImageIndex = aImageIndex;
        return oItem;
    }

    private void ToolbarItemClick(object Sender, TsgcToolbarItem aItem)
    {
        if (aItem.Caption == "Save")
            StatusBar.SetIndicator(1, TsgcStatusIndicatorState.sisOk);
        else if (aItem.Caption == "Wrap")
            Editor.TextWrapping = aItem.Down ? TextWrapping.Wrap : TextWrapping.NoWrap;
        StatusBar.SetPanelText(0, aItem.Caption + " clicked");
    }

    private void ZoomClick(object sender, RoutedEventArgs e)
    {
        StatusBar.SetPanelText(0, "Zoom " + ((MenuItem)sender).Header);
    }

    private void Editor_TextChanged(object sender, TextChangedEventArgs e)
    {
        StatusBar.SetIndicator(1, TsgcStatusIndicatorState.sisWarning);
    }
  }
}
```

Images 는 **BitmapImage** 든 **DrawingImage** 든 모 든 **ImageSource** 를 받 습 니 다. 목 록 이 **ObservableCollection<ImageSource>**이면 이미지가 추가, 제거 또는 교체될 때 툴바가 다시 그려집니다.

관련 항목

- .NET 개요
- .NET 클래스

- Delphi와 .NET의 차이점
- 빠른 시작 WinForms
- 빠른 시작 Avalonia
- 이미지와 글리프

빠른 시작 Avalonia

TsgcAvaloniaToolbar 와 **TsgcAvaloniaStatusBar** 는 Avalonia 바 이 며, Delphi 의 TsgcFMXToolbar 및 TsgcFMXStatusBar에 해당하고 Windows, Linux, macOS에서 동작합니다. 마우스, 터치, 펜은 같은 포인터 경로를 거칩니다. 패키지는 **esegece.sgcToolbar.Avalonia**입니다.

단계

- .NET 8 프로젝트를 만들고 esegece.sgcToolbar.Avalonia, Avalonia.Desktop, Avalonia.Themes.Fluent 패키지를 추가합니다.
- `xmlns:sgc="using:esegece.sgcToolbar"`로 네임스페이스를 매핑하고 **TsgcAvaloniaToolbar**와 **TsgcAvaloniaStatusBar**를 선언합니다.
- 코드 비하인드에서 **Theme**을 설정하고, 항목과 패널을 추가하고, **OnItemClick**을 할당합니다. 이것은 Avalonia 이벤트가 아니라 대리자입니다.

예제

MainWindow.axaml은 두 바와 텍스트를 선언합니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="using:esegece.sgcToolbar"
        x:Class="ToolbarQuickStartAvalonia.MainWindow"
        Title="sgcToolbar Avalonia" Width="720" Height="420">
    <DockPanel>
        <sgc:TsgcAvaloniaToolbar x:Name="Toolbar" DockPanel.Dock="Top" ShowCaptions="True" />
        <sgc:TsgcAvaloniaStatusBar x:Name="StatusBar" DockPanel.Dock="Bottom" />
        <TextBlock x:Name="lblMessage" Margin="24" FontSize="20" />
    </DockPanel>
</Window>
```

MainWindow.axaml.cs는 다크 테마를 설정하고, Grid와 List 라디오 그룹, 뒤따르는 항목을 오른쪽으로 밀어내는 스페이서, 레이블, Refresh 버튼, 스프링 텍스트 패널, 진행률 패널을 추가합니다.

```
using Avalonia.Controls;
using esegece.sgcToolbar;
using esegece.sgcUICore;

namespace ToolbarQuickStartAvalonia
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            // Theme is a TsgcUITheme; the items, the panels and the events are set in code
            Toolbar.Theme = TsgcUITheme.utDark;
            StatusBar.Theme = TsgcUITheme.utDark;
            DoAddButton(TsgcToolbarItemKind.tikToggle, "Grid", "#").GroupIndex = 1;
            DoAddButton(TsgcToolbarItemKind.tikToggle, "List", "=").GroupIndex = 1;
            Toolbar.Items[0].Down = true;
            DoAddButton(TsgcToolbarItemKind.tikSpacer, "", "");
            DoAddButton(TsgcToolbarItemKind.tikLabel, "Filter:", "");
            DoAddButton(TsgcToolbarItemKind.tikButton, "Refresh", "R").Hint = "Load the items again";
            Toolbar.OnItemClick = ToolbarItemClick;

            TsgcStatusBarPanel oPanel = StatusBar.Panels.Add();
            oPanel.Spring = true;
            oPanel.Text = "Grid view";
            oPanel = StatusBar.Panels.Add();
            oPanel.Kind = TsgcStatusPanelKind.spkProgress;
```

```
            oPanel.Progress = 40;
        }

        private TsgcToolbarItem DoAddButton(TsgcToolbarItemKind aKind, string aCaption, string aGlyph)
        {
            TsgcToolbarItem oItem = Toolbar.Items.Add();
            oItem.Kind = aKind;
            oItem.Caption = aCaption;
            oItem.Glyph = aGlyph;
            return oItem;
        }

        private void ToolbarItemClick(object Sender, TsgcToolbarItem aItem)
        {
            lblMessage.Text = aItem.Caption + " clicked";
            if (aItem.GroupIndex == 1)
                StatusBar.SetPanelText(0, aItem.Caption + " view");
        }
    }
}
```

Program.cs는 Fluent 테마로 애플리케이션을 시작합니다.

```
using System;
using Avalonia;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;

namespace ToolbarQuickStartAvalonia
{
    internal class App : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new MainWindow();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main(string[] args)
        {
            BuildAvaloniaApp().StartWithClassicDesktopLifetime(args);
        }

        public static AppBuilder BuildAvaloniaApp()
        {
            return AppBuilder.Configure<App>().UsePlatformDetect().LogToTrace();
        }
    }
}
```

Theme은 Avalonia 컨트롤의 `StyledElement.Theme` 속성을 가리므로, XAML 특성이 아니라 바 형식의 변수에서 코드로 설정하십시오. Images는 **IImage** 목록이며 **Bitmap** 또는 **DrawingImage**를 담습니다.

관련 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 토글과 라디오 그룹

.NET 클래스

각 호스트에는 같은 멤버를 가진 툴바 클래스 하나와 상태 표시줄 클래스 하나가 있습니다. 프레임워크에 독립적인 형식은 세 호스트가 공유하며 Delphi 이름을 유지합니다.

호스트 클래스

Host	클래스	기본 클래스	Delphi
WinForms	TsgcWinFormsToolbar	TsgcUIWinFormsControl	TsgcVCLToolbar
WinForms	TsgcWinFormsStatusBar	TsgcUIWinFormsControl	TsgcVCLStatusBar
WPF	TsgcWpfToolbar	TsgcUIWpfControl	TsgcFMXToolbar
WPF	TsgcWpfStatusBar	TsgcUIWpfControl	TsgcFMXStatusBar
Avalonia	TsgcAvaloniaToolbar	TsgcUIAvaloniaControl	TsgcFMXToolbar
Avalonia	TsgcAvaloniaStatusBar	TsgcUIAvaloniaControl	TsgcFMXStatusBar

프레임워크에 독립적인 형식

형식	설명
TsgcToolbarModel	항목, 오버플로 목록이 있는 레이아웃, 핫, 눌림, 포커스 항목, 히트 테스트. 모든 툴바가 하나씩 소유합니다. TsgcToolbarModel을 참조하십시오.
TsgcToolbarItems	항목 컬렉션: Count, 인덱서, Add, Insert, Delete, Clear, IndexOf, BeginUpdate, EndUpdate.
TsgcToolbarItem	항목 하나: Kind, Caption, Glyph, ImageIndex, Hint, Enabled, Visible, Down, GroupIndex, Width, Tag, DropdownMenu, OnClick, Click(), 레이아웃 결과 BoundsRect, Shown, InOverflow, 그리고 Registered 에디션에서는 Command와 CommandParameter. TsgcToolbarItem을 참조하십시오.
TsgcToolbarItemKind	tikButton, tikToggle, tikDropdown, tikSeparator, tikSpacer, tikLabel.
TsgcStatusBarModel	패널, 그 레이아웃, 크기 조절 핸들, 스레드 안전 세터. 모든 상태 표시줄이 하나씩 소유합니다. TsgcStatusBarModel을 참조하십시오.
TsgcStatusBarPanels	패널 컬렉션: Count, 인덱서, Add, Insert, Delete, Clear, IndexOf, BeginUpdate, EndUpdate.
TsgcStatusBarPanel	패널 하나: Kind, Text, Width, AutoSize, Spring, Alignment, Progress, IndicatorState, Hint, Tag, BoundsRect. TsgcStatusBarPanel을 참조하십시오.
TsgcStatusPanelKind	spkText, spkProgress, spkIndicator.
TsgcStatusIndicatorState	sisOff, sisOk, sisWarning, sisError.
TAlignment	taLeftJustify, taRightJustify, taCenter: 패널 텍스트의 정렬로, Delphi RTL 형식과 같습니다.

TsgcToolbarItemEvent	`void (object Sender, TsgcToolbarItem aItem)` , OnItemClick의 형식.
TsgcStatusBarPanelEvent	`void (object Sender, TsgcStatusBarPanel aPanel)` , OnPanelClick의 형식.
sgcToolbar_Render, sgcToolbar_Raster	렌더러와 헤드리스 PNG 출력. .NET 헤드리스 렌더링을 참조하십시오.

툴바

멤버	형식	설명
Model	TsgcToolbarModel	툴바의 모델.
Items	TsgcToolbarItems	항목, 왼쪽에서 오른쪽 순서.
ShowCaptions	bool	버튼의 캡션을 아이콘 옆에 그립니다. 이것이 없으면 아이콘이 있는 버튼은 아이콘만 표시합니다.
ButtonSize, Spacing, Padding	int	버튼의 높이, 항목 사이의 간격, 그 주위의 패딩으로, 기본값은 각각 32, 4, 6입니다. 툴바의 높이는 ButtonSize와 Padding을 따릅니다.
Theme, AccentColor	TsgcUITheme, uint	바의 테마. AccentColor가 0이면 테마의 강조색을 사용합니다.
Images	ImageList, IList<ImageSource>, IList<IImage>	ImageIndex의 이미지. Delphi와 .NET의 차이점을 참조하십시오.
OverflowMenu	ContextMenuStrip, ContextMenu	오버플로 버튼의 메뉴로, 열릴 때마다 다시 만들어집니다. Registered 에디션 전용입니다.
TabStop (WinForms), Focusable (WPF, Avalonia)	bool	바가 포커스를 받을 수 있게 합니다. Left, Right, Home, End는 포커스된 버튼을 이동하고 Enter와 Space는 버튼을 실행합니다.
ItemAtPos(X, Y)	int	컨트롤의 한 지점에 있는 항목의 인덱스, 없으면 -1, 오버플로 버튼 위에서는 CS_TOOLBAR_HIT_OVERFLOW.
ItemRect(aIndex), OverflowRect()	Rectangle, Rect	컨트롤 좌표에서 항목 또는 오버플로 버튼의 사각형.
ClickItem(aIndex)	void	클릭과 같은 방식으로 항목을 실행합니다.
OnItemClick	TsgcToolbarItemEvent	활성화된 버튼에 대한 클릭, Enter 또는 Space. 토글이 Down을 반전한 뒤, 항목의 OnClick 또는 Command가 실행되기 전에 발생합니다.

상태 표시줄

멤버	형식	설명
Model	TsgcStatusBarModel	상태 표시줄의 모델.
Panels	TsgcStatusBarPanels	패널, 왼쪽에서 오른쪽 순서.

SizeGrip	bool	오른쪽 끝에 크기 조절 핸들을 그리고, 끌면 창 크기를 조절합니다. Delphi와 .NET의 차이점을 참조하십시오.
Theme, AccentColor	TsgcUITheme, uint	바의 테마. AccentColor가 0이면 테마의 강조색을 사용합니다.
SetPanelText(aIndex, aText), SetProgress(aIndex, aValue), SetIndicator(aIndex, aState)	void	어느 스레드에서나 패널의 Text, Progress(0에서 100), IndicatorState를 설정합니다. 범위를 벗어난 인덱스는 무시됩니다.
PanelAtPos(X, Y)	int	컨트롤의 한 지점에 있는 패널의 인덱스, 없으면 -1.
PanelRect(aIndex), GripRect()	Rectangle, Rect	컨트롤 좌표에서 패널 또는 크기 조절 핸들의 사각형. SizeGrip이 없으면 비어 있습니다.
OnPanelClick	TsgcStatusBarPanelEvent	패널에 대한 클릭.
CS_STATUSBAR_HEIGHT	const int	기본 높이, 24.

위치와 크기는 WinForms(Rectangle)에서는 픽셀이고, WPF와 Avalonia(Rect)에서는 장치 독립 단위입니다.

관련 항목

- .NET 개요
- Delphi와 .NET의 차이점
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia

Delphi와 .NET의 차이점

.NET API는 Delphi API를 이름 하나하나까지 따릅니다. 다른 점은 다음과 같습니다.

Action 링크 대신 Command

Delphi는 Action 속성으로 항목을 TBasicAction에 연결합니다. Action 링크를 참조하십시오. .NET에는 TBasicAction이 없으므로 항목에는 **Command**(`System.Windows.Input.ICommand`)와 **CommandParameter**가 있습니다. 둘 다 Delphi의 Action 링크처럼 Registered 에디션에만 포함됩니다.

- `CanExecute(CommandParameter)`가 항목의 **Enabled**를 결정합니다. 명령이 할당될 때와 `CanExecuteChanged`가 발생할 때마다 갱신됩니다.
- 클릭, Enter, Space는 토글을 반전하고, 바의 OnItemClick을 발생시킨 다음, 항목의 `Click()`을 실행합니다. OnClick 처리기가 할당되어 있으면 그것을, 없으면 명령을 실행할 수 있을 때 명령을 실행합니다.
- ICommand에는 없으므로 Caption, Hint, Down은 명령에서 동기화되지 않습니다. 이들은 항목의 속성으로 남으므로 토글은 자신의 Down 상태를 유지합니다. ImageIndex와 Visible도 동기화되지 않습니다.
- 유휴 시 OnUpdate는 없습니다. 상태가 바뀌는 명령은 CanExecuteChanged를 발생시키고, 항목은 즉시 그에 따라갑니다.

Save 항목을 최소한의 명령에 연결하는 콘솔 프로그램입니다. 항목은 비활성 상태로 시작하고, 문서가 바뀌면 활성화되며, 클릭하면 저장한 뒤 다시 비활성화됩니다. esegece.sgcToolbar만 있으면 되며, 세 호스트의 툴바도 같은 방식으로 동작합니다.

```
using System;
using System.Windows.Input;
using esegece.sgcToolbar;

namespace ToolbarCommands
{
    // a minimal ICommand; MVVM toolkits provide the same as RelayCommand
    internal class TSaveCommand : ICommand
    {
        private bool FModified;

        public event EventHandler CanExecuteChanged;

        public bool Modified
        {
            get { return FModified; }
            set
            {
                FModified = value;
                EventHandler vHandler = CanExecuteChanged;
                if (vHandler != null)
                    vHandler(this, EventArgs.Empty);
            }
        }

        public bool CanExecute(object parameter)
        {
            return FModified;
        }

        public void Execute(object parameter)
        {
            Console.WriteLine("Saved " + parameter);
            Modified = false;
        }
    }

    internal static class Program
    {
        private static void Main()
        {
            TsgcToolbarModel oModel = new TsgcToolbarModel();
```

```
            TSaveCommand oCommand = new TSaveCommand();
            TsgcToolbarItem oSave = oModel.Items.Add();
            oSave.Caption = "Save";
            oSave.Hint = "Save the document";
            oSave.CommandParameter = "report.txt";
            // CanExecute drives Enabled: the item starts disabled
            oSave.Command = oCommand;
            Console.WriteLine("Enabled: " + oSave.Enabled);
            // the document changes, CanExecuteChanged enables the item
            oCommand.Modified = true;
            Console.WriteLine("Enabled: " + oSave.Enabled);
            // a click in a bar runs Click(): OnClick when assigned, else the command
            oSave.Click();
            Console.WriteLine("Enabled: " + oSave.Enabled + ", Caption: " + oSave.Caption);
        }
    }
}
```

드롭다운 메뉴와 오버플로 메뉴

Delphi에서 **DropdownMenu**는 VCL 또는 FireMonkey의 TPopupMenu입니다. 드롭다운 메뉴를 참조하십시오. .NET에서는 `object`로 선언되어 모델이 프레임워크에 독립적으로 유지되며, 각 호스트는 자신의 프레임워크의 메뉴 형식을 버튼 아래에 표시합니다. **OverflowMenu**는 열릴 때마다 툴바가 만들며, 오버플로 목록의 항목마다 하나의 항목이 있고, 항목을 선택하면 클릭과 같은 방식으로 실행됩니다. 오버플로를 참조하십시오.

Host	DropdownMenu	OverflowMenu
WinForms	System.Windows.Forms.ContextMenuStrip. 삭제된(disposed) 메뉴는 항목에서 지워집니다.	ContextMenuStrip. 항목의 Hint는 해당 메뉴 항목의 ToolTipText가 됩니다.
WPF	System.Windows.Controls.ContextMenu.	ContextMenu. 구분선 항목에는 Separator를 사용하고 Hint는 해당 메뉴 항목의 ToolTip이 됩니다.
Avalonia	Avalonia.Controls.ContextMenu.	ContextMenu. 구분선 항목에는 Separator를 사용하고 Hint는 해당 메뉴 항목의 팁이 됩니다.

메뉴는 버튼을 누를 때 OnItemClick과 항목의 OnClick 이후에 열리므로, 처리기가 메뉴가 표시되기 전에 내용을 채우거나 확인할 수 있습니다. 오버플로 메뉴는 Community 에디션에 포함되지 않습니다. Community에서는 들어가지 않는 항목이 잘리고 OverflowMenu 속성이 없습니다.

이미지

Delphi는 VCL에서 TCustomImageList를, FireMonkey에서 FMX.ImgList의 TCustomImageList를 사용합니다. 이미지와 글리프를 참조하십시오. .NET에서 툴바의 **Images** 속성은 각 호스트의 이미지 형식을 가지며, 항목의 **ImageIndex**는 그 안의 인덱스입니다.

Host	Images	그리는 크기
WinForms	System.Windows.Forms.ImageList	이미지 목록의 ImageSize(픽셀 단위).
WPF	IList<ImageSource>: 비트맵과 DrawingImage 벡터 아이콘. 벡터 아이콘은 Skia로 래스터화되므로, 최소화되었거나 연결이 끊긴 원격 데스크톱 세션에서도 그려집니다. ObservableCollection은 변경되면 다시 그려집니다.	16 x 16 장치 독립 단위. 벡터 아이콘은 모든 DPI에서 선명합니다.

Avalonia	IList<IImage>: Bitmap 또는 DrawingImage. INotifyCollectionChanged가 있는 목록은 변경되면 다시 그려집니다.	16 x 16 장치 독립 단위.

모든 이미지는 한 번만 ARGB로 변환되어 목록이 바뀔 때까지 캐시되며, WPF와 Avalonia에서는 DPI 배율이 바뀔 때까지도 캐시됩니다. 항목에 이미지가 없으면 그 **Glyph**의 텍스트가 아이콘 자리에 그려집니다.

힌트와 도구 설명

Delphi에서는 ShowHint가 True일 때 바가 커서 아래의 항목 또는 패널의 Hint를 표시하며, FMX는 10.1 Berlin부터 힌트를 표시합니다. .NET에서는 바가 항상 도구 설명을 표시하며 ShowHint 속성이 없습니다. 커서 아래 항목의 Hint를 표시하고, 캡션이 그려지지 않은 경우에는 그 Caption을, 상태 표시줄에서는 패널의 Hint를 표시합니다. WinForms는 내부 ToolTip 컴포넌트를 사용하고, WPF는 바의 ToolTip 속성을 설정하고, Avalonia는 ToolTip.Tip 연결 속성을 설정합니다. 바의 도구 설명을 직접 설정하지 마십시오. 바가 대체합니다. 대신 항목 또는 패널의 Hint를 설정하십시오.

스레드 안전 상태 갱신

Delphi와 마찬가지로 상태 표시줄의 **SetPanelText**, **SetProgress**, **SetIndicator**는 어느 스레드에서나 호출할 수 있습니다. 스레드 안전 갱신을 참조하십시오. 모델은 값을 잠금으로 보호하고, 값이 바뀐 경우에만 호출한 스레드에서 OnChange를 발생시킵니다. 바는 다시 그리기 요청으로 응답하며, 작업자 스레드에서는 컨트롤을 절대 건드리지 않습니다.

- WinForms와 WPF는 컨트롤이 UI 스레드에서 캡처한 `SynchronizationContext`를 통해 콜백 하나를 게시합니다.
- Avalonia는 `Dispatcher.UIThread`에 콜백 하나를 게시합니다.
- 한 번에 하나의 콜백만 대기하므로, 초당 수천 번 바를 갱신하는 작업자가 있어도 UI 큐가 넘치지 않습니다. 콜백은 최신 값으로 다시 그립니다.

Invoke, BeginInvoke, Dispatcher 호출은 필요하지 않습니다. 그 밖의 모든 속성과 패널 추가, 삭제, 구성은 UI 스레드에서 수행하도록 되어 있습니다. 아래 폼은 작업자 스레드에서 시뮬레이션된 청크 50개를 다운로드하고, 폼이 닫히기 전에 작업자를 중지합니다.

```
using System;
using System.Threading;
using System.Windows.Forms;
using esegece.sgcToolbar;

namespace ToolbarThreadStatus
{
    public class DownloadForm : Form
    {
        private readonly TsgcWinFormsStatusBar FStatusBar;
        private Thread FWorker;
        private volatile bool FStop;

        public DownloadForm()
        {
            Text = "Thread-safe status";
            Width = 600;
            Height = 200;
            FStatusBar = new TsgcWinFormsStatusBar();
            TsgcStatusBarPanel oPanel = FStatusBar.Panels.Add();
            oPanel.Spring = true;
            oPanel.Text = "Idle";
            oPanel = FStatusBar.Panels.Add();
            oPanel.Kind = TsgcStatusPanelKind.spkProgress;
            oPanel = FStatusBar.Panels.Add();
            oPanel.Kind = TsgcStatusPanelKind.spkIndicator;
            oPanel.AutoSize = true;
            oPanel.Text = "Feed";
            Controls.Add(FStatusBar);
        }

        public TsgcWinFormsStatusBar StatusBar
        {
            get { return FStatusBar; }
        }
```

```
        public void StartDownload()
        {
            FStop = false;
            FWorker = new Thread(DoDownload);
            FWorker.IsBackground = true;
            FWorker.Start();
        }

        // runs on the worker thread: no Invoke, the setters are thread-safe
        private void DoDownload()
        {
            FStatusBar.SetIndicator(2, TsgcStatusIndicatorState.sisOk);
            for (int i = 1; (i <= 50) && !FStop; i++)
            {
                Thread.Sleep(20);
                FStatusBar.SetPanelText(0, "Chunk " + i + " of 50");
                FStatusBar.SetProgress(1, i * 2);
            }
            FStatusBar.SetIndicator(2, FStop ? TsgcStatusIndicatorState.sisError :
                TsgcStatusIndicatorState.sisOff);
        }

        protected override void OnFormClosing(FormClosingEventArgs e)
        {
            // stop the worker before the status bar is destroyed
            FStop = true;
            if (FWorker != null)
                FWorker.Join();
            base.OnFormClosing(e);
        }
    }
}
```

크기 조절 핸들

SizeGrip이 True(기본값)이면 상태 표시줄은 오른쪽 끝에 핸들의 점을 그리고 **GripRect**는 그 사각형을 반환합니다. 크기 조절 핸들을 참조하십시오. Delphi에서는 VCL 바만 폼 크기를 조절하고 FMX 바는 핸들만 그립니다. .NET에서는 창 크기를 조절할 수 있고 최대화되지 않았을 때 모든 호스트에서 핸들이 창 크기를 조절하며, 디자인 타임에는 절대 조절하지 않습니다.

Host	핸들이 창 크기를 조절하는 방법
WinForms	바는 크기 조절 가능한 테두리가 있는 폼에 대해, VCL 바와 마찬가지로 핸들의 비클라이언트 히트 테스트에 HTBOTTOMRIGHT로 응답합니다.
WPF	ResizeMode가 CanResize 또는 CanResizeWithGrip인 Window에서 핸들을 누르면 창의 시스템 크기 조절 루프(SC_SIZE를 사용하는 WM_SYSCOMMAND)가 시작됩니다. 핸들 위에서는 커서가 바뀝니다.
Avalonia	CanResize인 Window에서 핸들을 누르면 창의 BeginResizeDrag가 호출됩니다. Windows, Linux, macOS에서 동작합니다.

단위

- WinForms: ButtonSize, Spacing, Padding, 항목과 패널의 Width는 논리 픽셀에 DPI / 96을 곱한 값이며, PPI / 96을 사용하는 VCL 바와 같습니다. ItemAtPos, ItemRect, PanelRect, GripRect는 클라이언트 픽셀입니다.
- WPF와 Avalonia: ItemAtPos, ItemRect, OverflowRect, PanelAtPos, PanelRect, GripRect는 FMX와 같은 장치 독립 단위입니다. 모델의 프레임과 레이아웃은 장치 픽셀이며, 단위에 컨트롤의 DPI 배율(WPF) 또는 창의 렌더링 배율(Avalonia)을 곱한 값입니다. 항목 또는 패널의 BoundsRect도 장치 픽셀입니다.
- 헤드리스: aWidth와 aHeight는 논리 픽셀이며 aScale이 이를 곱합니다. .NET 헤드리스 렌더링을 참조하십시오.

이벤트

OnItemClick, OnPanelClick, 항목의 OnClick은 Delphi 이름을 가진 대리자 속성이며 C# `event` 멤버가 아닙니다. 처리기를 할당하면 Delphi 메서드 포인터처럼 이전 것이 대체되며, 코드에서 할당합니다. XAML로 설정할 수 없는 WPF와 Avalonia에서도 마찬가지입니다.

디자이너 verb 없음

Delphi에서는 Items와 Panels 속성을 디자인 타임에 표준 컬렉션 편집기로 편집합니다. .NET 이식판에는 디자이너 verb도 컬렉션 편집기도 없습니다. 바의 Items와 Panels는 직렬화되지 않으므로 항상 코드에서, 보통 폼이나 창의 생성자에서 추가합니다. WinForms에서 바는 도구 상자에 있으며 단순한 속성은 속성 창에서 설정합니다. WPF와 Avalonia에서는 바를 XAML로 선언합니다.

에디션

.NET 이식판은 Registered 패키지와 Community 패키지로 제공됩니다. .NET 개요를 참조하십시오. Community 패키지에는 Delphi Community 에디션처럼 OverflowMenu, Command, CommandParameter가 없습니다. 에디션을 참조하십시오. Trial 에디션은 없습니다.

데이터 바인딩

WPF와 Avalonia에서 사용자용 속성은 종속성 속성과 Avalonia 속성이므로 데이터 바인딩, 스타일 지정, 애니메이션이 가능합니다. .NET 데이터 바인딩을 참조하십시오.

관련 항목

- .NET 개요
- .NET 클래스
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- Action 링크
- 드롭다운 메뉴
- 이미지와 글리프
- 스레드 안전 갱신
- 크기 조절 핸들

.NET 데이터 바인딩

WPF 및 Avalonia 호스트에서 도구 모음과 상태 표시줄의 사용자용 속성은 종속성 속성(WPF)과 Avalonia 스타일 속성이므로 뷰 모델에 데이터 바인딩하거나 Style로 설정하거나 애니메이션을 적용할 수 있습니다. 이 항목은 WPF와 Avalonia에만 적용됩니다. WinForms 컨트롤은 변경되지 않았으며, 해당 속성은 이전과 같은 일반 CLR 속성입니다.

바인딩 가능한 속성

표의 모든 속성에는 접미사 Property가 붙은 정적 필드가 있습니다. 예를 들어 `TsgcWpfToolbar.ShowCaptionsProperty`입니다. WPF에서 이 필드는 DependencyProperty이고 Avalonia에서는 StyledProperty입니다. 이 속성들은 사용자가 변경하지 않으므로 모두 기본적으로 OneWay로 바인딩됩니다. 컨트롤에서 코드로 설정한 값을 뷰 모델도 받아야 하면 `Mode=TwoWay`를 사용하십시오.

WPF 클래스	Avalonia 클래스	바인딩 가능한 속성
TsgcWpfToolbar	TsgcAvaloniaToolbar	ShowCaptions, ButtonSize, Spacing, Padding, Theme, AccentColor
TsgcWpfStatusBar	TsgcAvaloniaStatusBar	Theme, SizeGrip, AccentColor

Padding은 장치 독립 단위로 표현한 막대의 안쪽 여백이며 int입니다. WPF에서는 Control의 Thickness 형식 Padding을 가립니다. Avalonia에서 Theme은 sgcUICore 테마(TsgcUITheme)이며 StyledElement의 Avalonia 컨트롤 테마를 가립니다. 후자는 `StyledElement.ThemeProperty`를 통해 계속 접근할 수 있습니다.

동작

- **모델이 값을 유지합니다.** 모든 속성은 도구 모음 또는 상태 표시줄 모델에 저장됩니다. 바인딩된 값은 모델로 들어가고, 속성은 모델이 유지하는 값을 다시 가져옵니다. 모델은 ButtonSize를 8로, Spacing과 Padding을 0으로 올리고, 속성은 그 뒤에 사용 중인 값을 표시합니다.
- **수정된 값은 소스에 도달합니다.** TwoWay 소스도 WPF와 Avalonia 모두에서 수정된 값을 받습니다. WPF는 소스에서 값을 전송하는 동안에는 소스를 업데이트하지 않으므로, WPF에서는 디스패처가 대기 중인 작업을 실행한 뒤에 컨트롤이 수정된 값을 다시 기록합니다. 이후 컨트롤에서 코드 또는 모델을 통해 이루어진 변경도 두 플랫폼 모두에서 TwoWay 소스에 도달합니다. Avalonia에서 다시 쓰기는 바인딩이 실행 중일 때 소스가 보낸 값에 적용됩니다. 바인딩이 시작될 때 소스가 이미 가지고 있는 값은 컨트롤에서 수정되지만 소스에는 다시 기록되지 않습니다.
- **모델에서 만든 변경은 다시 복사됩니다.** 모델이 스스로 값을 변경하면 컨트롤이 그 값을 속성에 복사하므로 바인딩과 스타일이 사용 중인 값을 봅니다.
- **스타일.** Style setter는 바인딩 가능한 모든 속성을 설정할 수 있습니다. 예를 들어 창에 있는 모든 도구 모음과 상태 표시줄의 Theme입니다.

CLR 속성으로 남는 속성

속성	이유
Items (도구 모음)	모델의 버튼 컬렉션이며 코드로 만듭니다. 바인딩할 단일 값이 아닙니다.
Images (도구 모음)	버튼의 이미지 목록이며 참조로 공유되는 개체입니다.
OverflowMenu (도구 모음)	들어가지 않는 버튼을 위해 도구 모음이 만드는 메뉴이며 읽기 전용입니다.
Panels (상태 표시줄)	모델의 패널 컬렉션입니다. 패널 값(Text, Progress, IndicatorState)은 모든 스레드에서 호출할 수 있는 스레드 안전 메서드 SetPanelText, SetProgress, SetIndicator로 설정합니다.
Model	각 컨트롤 뒤에 있는 플랫폼 독립적인 모델이며 읽기 전용입니다.

예제

아래 뷰 모델은 두 예제에서 공유됩니다. ShowCaptions와 Theme은 OneWay로 바인딩되고 ButtonSize는 `Mode=TwoWay`로 바인딩되므로, 도구 모음에서 코드로 설정한 크기가 뷰 모델로 돌아옵니다. 버튼은 빠른 시작과 같이 코드로 추가합니다.

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;
using esegece.sgcUICore;

namespace ToolbarBinding
{
    public class MainViewModel : INotifyPropertyChanged
    {
        private bool FShowCaptions = true;
        private int FButtonSize = 32;
        private TsgcUITheme FTheme = TsgcUITheme.utLight;

        public event PropertyChangedEventHandler PropertyChanged;

        public bool ShowCaptions
        {
            get { return FShowCaptions; }
            set { DoSet(ref FShowCaptions, value); }
        }

        // bound TwoWay, so a size set on the toolbar comes back here
        public int ButtonSize
        {
            get { return FButtonSize; }
            set { DoSet(ref FButtonSize, value); }
        }

        public TsgcUITheme Theme
        {
            get { return FTheme; }
            set { DoSet(ref FTheme, value); }
        }

        private void DoSet<T>(ref T aField, T aValue, [CallerMemberName] string aName = null)
        {
            if (EqualityComparer<T>.Default.Equals(aField, aValue))
                return;
            aField = aValue;
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(aName));
        }
    }
}
```

WPF

MainWindow.xaml은 뷰 모델을 창의 DataContext로 만듭니다. 코드 비하인드와 App.xaml은 WPF 템플릿의 것입니다.

```
<Window x:Class="ToolbarBinding.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:ToolbarBinding"
        xmlns:sgc="clr-namespace:esegece.sgcToolbar;assembly=esegece.sgcToolbar.Wpf"
        Title="sgcToolbar binding" Width="720" Height="420">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <!-- ButtonSize binds OneWay by default, Mode=TwoWay sends changes back -->
        <sgc:TsgcWpfToolbar x:Name="Toolbar" DockPanel.Dock="Top" ShowCaptions="{Binding ShowCaptions}"
```

```
                        ButtonSize="{Binding ButtonSize, Mode=TwoWay}" Theme="{Binding Theme}" />
        <sgc:TsgcWpfStatusBar x:Name="StatusBar" DockPanel.Dock="Bottom" Theme="{Binding Theme}" />
        <StackPanel Margin="24">
            <CheckBox Content="Show captions" IsChecked="{Binding ShowCaptions}" />
            <TextBlock Text="{Binding ButtonSize}" />
        </StackPanel>
    </DockPanel>
</Window>
```

Avalonia

MainWindow.axaml은 `x:DataType`을 뷰 모델로 지정하므로 바인딩은 컴파일된 바인딩입니다. 코드 비하인드는 InitializeComponent를 호출하고 App 클래스는 빠른 시작 Avalonia의 것입니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:ToolbarBinding"
        xmlns:sgc="using:esegece.sgcToolbar"
        x:Class="ToolbarBinding.MainWindow"
        x:DataType="local:MainViewModel"
        Title="sgcToolbar binding" Width="720" Height="420">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <!-- ButtonSize binds OneWay by default, Mode=TwoWay sends changes back -->
        <sgc:TsgcAvaloniaToolbar x:Name="Toolbar" DockPanel.Dock="Top" ShowCaptions="{Binding ShowCaptions}"
                                 ButtonSize="{Binding ButtonSize, Mode=TwoWay}" Theme="{Binding Theme}" />
        <sgc:TsgcAvaloniaStatusBar x:Name="StatusBar" DockPanel.Dock="Bottom" Theme="{Binding Theme}" />
        <StackPanel Margin="24">
            <CheckBox Content="Show captions" IsChecked="{Binding ShowCaptions}" />
            <TextBlock Text="{Binding ButtonSize}" />
        </StackPanel>
    </DockPanel>
</Window>
```

참고 항목

- Delphi와 .NET의 차이점
- .NET 클래스
- 빠른 시작 WPF
- 빠른 시작 Avalonia

.NET 헤드리스 렌더링

툴바나 상태 표시줄은 서버, 테스트, 보고서처럼 창 없이도 그릴 수 있습니다. 두 모델의 **SaveToSVG**는 esegece.sgcToolbar 패키지만으로 SVG 파일 또는 스트림을 쓰고, esegece.sgcToolbar.Rendering의 **SaveToPNG** 확장 메서드는 Windows, Linux, macOS에서 SkiaSharp로 그린 PNG를 씁니다. SVG 출력은 Delphi 헤드리스 렌더링의 출력과 바이트 단위로 같습니다.

메서드

메서드	패키지	설명
TsgcToolbarModel.SaveToSVG 및 TsgcStatusBarModel.SaveToSVG(aFileName 또는 aStream, aWidth, aHeight, aScale = 1)	esegece.sgcToolbar	바를 SVG로 씁니다.
SaveToPNG(aFileName 또는 aStream, aWidth, aHeight, aScale = 1), sgcToolbar_Raster에 있는 두 모델의 확장 메서드	esegece.sgcToolbar.Rendering	바를 투명한 배경의 PNG로 씁니다.
TsgcToolbarModel.Render 및 TsgcStatusBarModel.Render(aCanvas, aWidth, aHeight, aScale = 1)	esegece.sgcToolbar	임의의 sgcUICore 캔버스에 바를 그립니다. 바는 aWidth x aHeight 장치 픽셀입니다.
sgcToolbar_Render.RenderToolbar(aModel, aCanvas, aWidth, aHeight, aScale, aImages), sgcToolbar_Render.RenderStatusBar(aModel, aCanvas, aWidth, aHeight, aScale)	esegece.sgcToolbar	Render 뒤에 있는 렌더러.

SaveToSVG와 SaveToPNG에서 aWidth와 aHeight는 논리 픽셀입니다. 이미지 크기는 가로 aWidth x aScale, 세로 aHeight x aScale 픽셀이므로, 배율 2는 고DPI 화면이나 인쇄용으로 선명한 이미지를 만듭니다. 툴바 모델에는 이미지 목록이 없습니다. 그 **OnGetImage** 이벤트(`bool (int aIndex, out TsgcUIARGBBuffer aImage)`)가 ImageIndex의 이미지를 **TsgcUIARGBBuffer**로 반환하며, 이것이 없으면 항목의 Glyph가 아이콘 자리에 그려집니다.

예제

esegece.sgcToolbar.Rendering을 참조하는 콘솔 프로그램(`net8.0`)으로, 비활성 버튼, 눌린 토글, 드롭다운, 스페이서에 의해 오른쪽으로 밀린 레이블이 있는 툴바와, 스프링 텍스트 패널, 65의 진행률 패널, 경고 표시기가 있는 다크 상태 표시줄을 각각 PNG와 SVG로 씁니다.

```
using System;
using esegece.sgcToolbar;
using esegece.sgcUICore;

namespace ToolbarHeadless
{
    internal static class Program
    {
        private static TsgcToolbarItem DoAdd(TsgcToolbarModel aModel, TsgcToolbarItemKind aKind,
            string aCaption, string aGlyph)
        {
            TsgcToolbarItem oItem = aModel.Items.Add();
            oItem.Kind = aKind;
            oItem.Caption = aCaption;
```

```
            oItem.Glyph = aGlyph;
            return oItem;
        }

        private static void Main()
        {
            TsgcToolbarModel oToolbar = new TsgcToolbarModel();
            oToolbar.ShowCaptions = true;
            DoAdd(oToolbar, TsgcToolbarItemKind.tikButton, "New", "N");
            DoAdd(oToolbar, TsgcToolbarItemKind.tikButton, "Save", "S").Enabled = false;
            DoAdd(oToolbar, TsgcToolbarItemKind.tikSeparator, "", "");
            DoAdd(oToolbar, TsgcToolbarItemKind.tikToggle, "Bold", "B").Down = true;
            DoAdd(oToolbar, TsgcToolbarItemKind.tikDropdown, "Zoom", "Z");
            DoAdd(oToolbar, TsgcToolbarItemKind.tikSpacer, "", "");
            DoAdd(oToolbar, TsgcToolbarItemKind.tikLabel, "Ln 1", "");
            // logical pixels; a scale of 2 doubles the pixels of the image
            oToolbar.SaveToSVG("toolbar.svg", 520, 44);
            oToolbar.SaveToPNG("toolbar.png", 520, 44, 2);

            TsgcStatusBarModel oStatus = new TsgcStatusBarModel();
            oStatus.Theme = TsgcUITheme.utDark;
            TsgcStatusBarPanel oPanel = oStatus.Panels.Add();
            oPanel.Spring = true;
            oPanel.Text = "Downloading";
            oPanel = oStatus.Panels.Add();
            oPanel.Kind = TsgcStatusPanelKind.spkProgress;
            oPanel = oStatus.Panels.Add();
            oPanel.Kind = TsgcStatusPanelKind.spkIndicator;
            oPanel.AutoSize = true;
            oPanel.Text = "Feed";
            oStatus.SetProgress(1, 65);
            oStatus.SetIndicator(2, TsgcStatusIndicatorState.sisWarning);
            oStatus.SaveToSVG("status.svg", 520, 24);
            oStatus.SaveToPNG("status.png", 520, 24, 2);
            Console.WriteLine("toolbar.png, toolbar.svg, status.png and status.svg written");
        }
    }
}
```

관련 항목

- .NET 개요
- .NET 클래스
- 헤드리스 렌더링
- TsgcToolbarModel
- TsgcStatusBarModel

.NET 데모

.NET 데모는 제품의 `demos` 폴더와 데모 zip에 있으며 솔루션 `sgcToolbar_Demos.sln`이 함께 제공됩니다. 각 데모에는 README.md가 있습니다.

데모	Host	설명
EditorShell	WinForms	TsgcWinFormsToolbar와 TsgcWinFormsStatusBar가 있는 작은 텍스트 편집기의 셸. 툴바에는 New, Open, Save 버튼, Bold와 Italic 토글, Left, Center, Right 라디오 그룹(GroupIndex 1), 네 가지 확대 수준 메뉴가 있는 Zoom 드롭다운, 스페이서, Dark 테마 토글, Help 버튼이 있습니다. 상태 표시줄에는 캐럿의 줄과 열(AutoSize), 마지막 메시지를 보여 주는 스프링 패널, 확대 수준, 문서에 저장하지 않은 변경 사항이 있는 동안 노란색으로 바뀌는 표시기가 표시됩니다. Save는 저장하지 않은 변경 사항이 있을 때만 활성화되고, 창을 좁히면 들어가지 않는 항목이 오버플로 메뉴로 이동합니다. 버튼은 항목의 OnClick 처리기와 OnItemClick을 사용하므로 이 데모는 Community 패키지와 Registered 패키지 모두로 빌드됩니다.
EditorShellWpf	WPF	같은 편집기를 TsgcWpfToolbar와 TsgcWpfStatusBar로 구현한 것.
EditorShellAvalonia	Avalonia	같은 편집기를 TsgcAvaloniaToolbar와 TsgcAvaloniaStatusBar로 구현한 것으로, Windows, Linux, macOS에서 동작합니다.
LiveStatus	WinForms	작업자 스레드가 공급하는 실시간 상태 표시줄. Start는 네트워크 없이 50개 청크의 다운로드를 시뮬레이션하는 작업자를 실행하고, 스레드 안전한 SetPanelText, SetProgress, SetIndicator로 진행 상황을 상태 표시줄에 바로 보고합니다. 상태 표시줄에는 스프링 메시지 패널, 진행률 패널, 피드 표시기(다운로드 중에는 녹색, 청크가 느릴 때는 노란색, 중지되면 빨간색), AutoSize 청크 카운터가 있습니다. Stop은 작업자를 끝내며, 작업자는 SynchronizationContext를 통해 UI 스레드에 종료를 보고합니다.
LiveStatusWpf	WPF	같은 실시간 상태 표시줄을 TsgcWpfStatusBar로 구현한 것.
LiveStatusAvalonia	Avalonia	같은 실시간 상태 표시줄을 TsgcAvaloniaStatusBar로 구현한 것.

빌드와 실행

데모에는 .NET 8 SDK 이상이 필요합니다. 데모의 폴더에서 데모를 실행합니다.

```
dotnet run
```

제품 트리 안에서 데모는 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip에서는 NuGet 패키지를 복원하며, 기본적으로 Community 에디션을 사용합니다. Registered 패키지를 사용하려면 **SgcToolbarEdition**을 설정합니다.

```
dotnet build -p:SgcToolbarEdition=Registered
```

자체 점검

각 데모에는 창을 화면 밖에서 열어 조작하는 `--test` 스위치가 있습니다. EditorShell은 라디오 그룹, 토글, Save, Zoom 메뉴, 좁은 창의 오버플로 메뉴, Dark 토글을 클릭하고, LiveStatus는 다운로드를 끝까지 실행하고, 두 번째 다운로드를 중간에 중지하고, 테마를 전환합니다. 데모는 점검마다 PASS 또는 FAIL 한 줄을 파일에 쓰고 실패한 점검의 수를 종료 코드로 종료하므로, 0은 모든 점검이 통과했음을 의미합니다.

```
dotnet run -- --test result.txt
```

관련 항목

- .NET 개요
- NuGet 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- 데모

문제 해결

Undeclared identifier Action 또는 OverflowMenu

Community 에디션은 SGC_TOOLBAR_ADVANCED 없이 컴파일되므로, 항목의 Action 속성과 툴바의 OverflowMenu 속성이 존재하지 않습니다. 이를 사용하는 코드는 E2003 Undeclared identifier 오류로 실패하며, 액션에 연결된 항목이 있는 상태로 Registered 빌드에서 저장한 폼은 로드할 때 "Error reading" 예외를 발생시킵니다. 항목의 OnClick 또는 바의 OnItemClick 이벤트를 사용하거나, Registered 또는 Trial 빌드를 사용하십시오. 에디션을 참조하십시오.

마지막 항목이 오른쪽 가장자리에서 잘립니다

Community 에디션에는 오버플로 메뉴가 없으므로 공간에 맞지 않는 항목이 잘립니다. 바를 더 넓게 만들거나, ShowCaptions를 False로 설정하거나, Registered 또는 Trial 빌드를 사용하십시오. 오버플로를 참조하십시오.

FireMonkey에서 메뉴 뒤의 코드가 늦게 실행됩니다

드롭다운 항목과 오버플로 버튼의 FireMonkey TPopupMenu는 모달입니다. Popup은 메뉴가 닫힐 때까지 반환되지 않습니다. 드롭다운 항목에서 ClickItem 뒤에 놓은 코드나, 메뉴가 표시된 뒤 OnItemClick 안의 코드는 사용자가 메뉴를 닫은 후에야 실행됩니다. 대신 메뉴 항목의 OnClick에서 작업을 수행하고, 기다리면 안 되는 타이머나 테스트에서 드롭다운 항목에 ClickItem을 호출하지 마십시오. VCL에서는 버튼을 누르는 순간 메뉴가 열리고, 그 모달 루프가 마우스 버튼 뗌을 가져갑니다.

툴바가 설정한 Height를 무시합니다

툴바의 높이는 항상 ButtonSize에 Padding의 두 배를 더한 값을 따릅니다. 대신 이 두 속성을 변경하십시오.

라디오 토글을 끌 수 없습니다

GroupIndex가 0보다 큰 토글은 라디오 버튼이며, 클릭하면 켜지기만 합니다. 코드에서 Down을 False로 설정하거나, 독립적인 토글에는 GroupIndex 0을 사용하십시오. 토글과 라디오 그룹을 참조하십시오.

항목이 클릭에 반응하지 않습니다

비활성 항목(Enabled False)은 흐리게 그려지며 클릭과 키보드를 무시하고, 구분선, 스페이서, 레이블은 클릭되지 않습니다. Action이 있는 항목은 액션의 Enabled를 따르므로, 액션의 OnUpdate도 확인하십시오.

스레드에서 갱신하면 상태 표시줄이 깜박이거나 예외가 발생합니다

스레드 안전한 것은 SetPanelText, SetProgress, SetIndicator뿐입니다. 작업자 스레드에서 다른 속성을 설정하거나 패널을 추가하면 메인 스레드 밖에서 컨트롤을 건드리게 됩니다. 스레드 안전 갱신을 참조하십시오.

크기 조절 핸들이 폼 크기를 바꾸지 않습니다

FMX에서는 핸들이 그려지기만 합니다. VCL에서는 폼에 크기 조절 가능한 테두리가 있고 최대화되지 않은 경우에만 동작합니다. 크기 조절 핸들을 참조하십시오.

FireMonkey에서 이미지나 힌트가 표시되지 않습니다

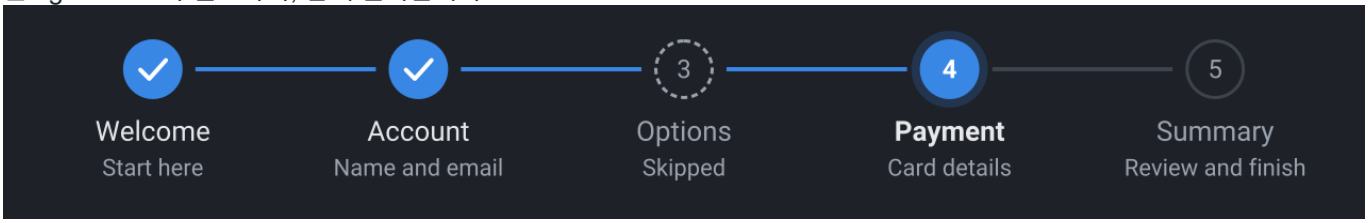
FMX 툴바의 Images 속성에는 Delphi XE8 이상이 필요하고, FMX 바의 힌트에는 ShowHint True와 함께 10.1 Berlin 이상이 필요합니다. 이전 버전에서는 항목이 Glyph를 그립니다. 이미지와 글리프를 참조하십시오.

EditorShell 또는 EditorShellFMX 데모가 컴파일되지 않습니다

두 데모 모두 Community 에디션에 포함되지 않는 Action 링크와 오버플로 메뉴를 사용합니다. 데모를 참조하십시오.

소개

sgcWizard는 VCL과 FireMonkey용 Delphi 및 C++Builder 애플리케이션에 단계별 마법사와 단계 표시기를 추가합니다. **TsgcVCLWizard**와 **TsgcFMXWizard**는 페이지 컨테이너입니다. 각 페이지가 하나의 단계이고, 스테퍼 밴드가 사용자의 현재 위치를 보여 주며, 버튼 바가 Back, Next, Finish, Cancel로 페이지 사이를 이동합니다. **TsgcVCLStepper**와 **TsgcFMXStepper**는 단독으로 사용하는 단계 표시기로, 가로 또는 세로 방향이며 진행 상황을 보여 줘야 하는 모든 흐름에 사용할 수 있습니다.
sgcWizard는 sgcDashboard, sgcNavigation, sgcGrid, sgcToolbar와 함께 eSeGeCe 비주얼 제품군에 속합니다. 캔버스, 래스터라이저, 글꼴, SVG 작성기, 라이트 및 다크 테마를 갖춘 공유 런타임 패키지인 **sgcUICore**로 그려지므로, 마법사는 VCL, FireMonkey, 헤드리스 PNG 및 SVG 출력에서 동일하게 보이고 제품군의 다른 제품과도 잘 어울립니다. sgcWizard에는 sgcUICore가 필요하며, 함께 설치됩니다.



에디션

sgcWizard는 Registered, Trial, Community로 제공됩니다. 모든 에디션에 모든 기능이 있으며, Trial 빌드만 체험판 안내를 표시합니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 프레임워크 중립 스테퍼 모델, sgcUICore 캔버스 위의 렌더러, VCL 및 FMX 컨트롤.
- 플랫폼: Delphi 버전, C++Builder, 지원 플랫폼.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder: 몇 분 만에 만드는 첫 마법사.
- 구성 요소: 마법사, 페이지, 스테퍼, 단계, 모델.
- 탐색과 건너뛴 페이지, 유효성 검사, Finish와 Cancel, 키보드, 단계 상태, 스테퍼 레이아웃, 버튼 캡션, 테마, 디자인 타임, 헤드리스 렌더링.
- 데모: Delphi 및 C++Builder 데모 프로젝트.

아키텍처

sgcWizard는 계층으로 구성됩니다. 각 계층은 아래 계층에만 의존하므로, 모델과 렌더러는 VCL이나 FireMonkey 없이 컴파일됩니다.

계층	유닛	역할
모델	sgcWizard_Model	TsgcStepperModel은 Steps(TsgcWizardStep의 컬렉션), 방향, 활성 단계, 자동 상태, 크기, 테마, 핫 스텝을 보관합니다. Layout 메서드는 보이는 모든 단계의 원과 레이블, 그리고 단계 사이의 연결선을 계산하고, HitTest는 한 지점 아래에 있는 단계를 찾습니다.
렌더러	sgcWizard_Render	sgcWizardRender는 모든 sgcUICore 캔버스에 모델을 그리고, sgcWizardRenderButtonBar는 버튼 바를 그립니다. sgcWizardSaveToPNG와 sgcWizardSaveToSVG는 이를 사용해 창 없이 이미지를 기록합니다. 헤드리스 렌더링을 참조하십시오.
VCL 컨트롤	sgcWizard_VCL	TsgcVCLWizard, TsgcVCLWizardPage, TsgcVCLStepper. Windows 전용입니다.
FMX 컨트롤	sgcWizard_FMX	TsgcFMXWizard, TsgcFMXWizardPage, TsgcFMXStepper이며, API와 동작이 동일합니다.
sgcUICore	sgcUICore_Canvas, sgcUICore_VCL, sgcUICore_FMX, sgcUICore_Theme	공유 런타임: 벡터 캔버스와 래스터라이저, VCL 및 FMX 컨트롤 기반, 그리고 테마 적용기를 갖춘 라이트 및 다크 테마 토큰.

스테퍼

스테퍼 컨트롤은 TsgcStepperModel을 소유하고 그 속성을 게시합니다. **Model** 속성으로 나머지 멤버인 Layout, HitTest, IsStepClickable, IsConnectorDone, NextStep, PrevStep에 접근합니다. 모델이 변경되면 새 프레임이 예약되며 직접 그리지 않으므로, 연속된 여러 변경도 다시 그리기는 한 번만 발생합니다.

마법사와 페이지

마법사는 TPageControl과 그 TTabSheet 페이지처럼 페이지 컨테이너입니다. 페이지는 컨트롤 순서대로 마법사의 자식 컨트롤이며, 활성 페이지만 보입니다. 모든 페이지에는 편집 상자, 확인란, 레이블, 프레임 등 일반 컨트롤을 넣을 수 있습니다.
마법사는 스트리밍되지 않고 디자이너에서 선택할 수 없는 두 개의 내부 자식 컨트롤을 소유합니다. 하나는 **스테퍼 밴드**로, 단계가 페이지를 그대로 반영하는 스테퍼입니다(Caption, Description, StepGlyph, 그리고 건너뜀 상태로서의 Skip). 다른 하나는 Back, Next(마지막 페이지에서는 Finish), Cancel이 있는 **버튼 바**입니다. 밴드의 상태는 마법사가 직접 설정하므로 밴드의 AutoStates는 False입니다. 둘 다 실행 시 **Stepper**와 **ButtonBar** 속성으로 사용할 수 있습니다.

그리기

단계, 연결선, 버튼은 렌더러가 그립니다. 모든 크기는 논리 픽셀에 모니터 배율을 곱한 값이므로, 컨트롤은 어떤 DPI에서도 선명합니다.

관련 항목

- 플랫폼
- TsgcStepperModel
- 테마

에디션

sgcWizard는 모든 에디션에서 동일한 소스로 빌드되며, 모든 에디션에 모든 기능이 있습니다.

에디션	포함 내용
Registered	전부: VCL 및 FMX 마법사, 페이지와 스테퍼, 유효성 검사, 건너뛴 페이지, 단계 상태, 테마, 디자인 타임 지원, 헤드리스 PNG 및 SVG 출력.
Trial	Registered와 동일한 기능입니다. SGC_TRIAL로 컴파일한 빌드는 빌드에 지정된 릴리스 기간이 지난 뒤 컴포넌트를 생성할 때 체험판이라는 메시지를 표시합니다. 그 외에는 달라지는 것이 없습니다.
Community	Registered와 동일한 기능이며 체험판 안내가 없습니다.

에디션은 sgcVer.inc에서 컴파일 시점에 선택합니다. SGC_COMMUNITY는 SGC_TRIAL도 끕니다. 런타임 스위치가 아니며, 한 에디션을 기준으로 작성한 코드는 다른 에디션에서도 컴파일됩니다.

관련 항목

- 플랫폼
- 소개

설치

sgcWizard는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램으로 sgcUI 제품(sgcUICore, sgcCharts, sgcPDF, sgcReports, sgcMaps, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard)을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcWizard**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하며, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 사용할 수 있게 합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE에 대해 사용자의 컴퓨터에서 의존 순서대로 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcWizard에 포함된 항목

sgcUICore는 레이아웃 제품과 sgcCharts의 공통 기반이며 sgcWizard와 함께 제공됩니다. 자체 확인란은 없습니다. sgcWizard를 선택하면 설치 프로그램이 자동으로 설치합니다. sgcWizard는 sgcWebSockets 브리지 패키지가 없습니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드입니다. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지이며, 제품과 IDE 버전마다 한 세트씩 있습니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛입니다. 예를 들어 `LibD13` 입니다.
`Demos\Wizard`	sgcWizard 데모입니다. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 별도의 경로가 필요하지 않습니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 모든 제품을 체험판 모드로 설치하며, 여기에는 sgcWizard도 포함됩니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 sgcWizard가 포함됩니다. Community의 제품은 sgcCharts, sgcDashboard, sgcGrid, sgcNavigation, sgcToolbar, sgcWizard이며, sgcPDF와 sgcMaps는 Community에 포함되지 않습니다. sgcWizard의 Community 빌드는 Registered 빌드와 같은 기능을 제공합니다. 에디션을 참조하십시오.

이전 독립 실행형 설치

설치 프로그램이 이전 독립 실행형 sgcUICore 설치(예: 독립 실행형 sgcCharts 설치 프로그램이 만든 것)를 발견하면, 그 설치의 라이브러리 경로와 디자인 패키지를 **Previous installations** 페이지에 나열합니다. 계속하면 새 패키지를 컴파일하기 전에 IDE에서 그것들을 제거하므로, 경로에 같은 유닛의 사본이 두 개 존재하는 일은 없습니다.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전에 맞는 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcWizardD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcWizard<V>` / `dclsgcWizard<V>`	sgcWizard 컴포넌트(런타임/디자인 타임).

의존 순서대로 빌드합니다. `sgcUICore`, 그다음 `sgcWizard`, 그다음 `dclsgcWizard`입니다. 그런 다음 `dclsgcWizard` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder의 경우 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcWizardC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

.NET

.NET에서는 달라지는 것이 없습니다. sgcWizard는 여전히 자체 NuGet 패키지(`esegece.sgcWizard`와 그 보조 패키지)로 배포됩니다. .NET 설치를 참조하십시오.

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE~XE8, 10~10.4, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE~XE8, 10~10.4, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows, Win32 및 Win64에서 모든 Delphi 버전의 TsgcVCLWizard, TsgcVCLWizardPage, TsgcVCLStepper.
FireMonkey	Delphi XE6 이상(SGC_WIZARD_FMX)의 TsgcFMXWizard, TsgcFMXWizardPage, TsgcFMXStepper로, Windows, macOS, iOS, Android, 그리고 FMX for Linux(FMXLinux)를 사용하는 Linux에서 동작합니다. FMX 컨트롤의 힌트에는 10.1 Berlin 이상이 필요합니다.
Linux64	Delphi 10.2 이상(SGC_WIZARD_LINUX)에서 프레임워크 중립 유닛, 모델, 그리고 헤드리스 PNG 및 SVG 출력을 갖춘 렌더러.
.NET	WinForms, WPF, Avalonia와 헤드리스 PNG 및 SVG 출력을 위한 동일한 API를 갖춘 C# 포트이며, .NET Framework 4.7.2와 .NET 8 및 9에서 동작합니다. .NET 개요를 참조하십시오.

패키지

sgcWizard에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcWizardD13.dpk, C++Builder 13에서는 sgcWizardC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcWizard	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, sgcUICore
sgcWizardVCL	런타임, VCL	sgcWizard_VCL (TsgcVCLWizard, TsgcVCLWizardPage, TsgcVCLStepper)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcWizard
sgcWizardFMX	런타임, FireMonkey, XE6부터	sgcWizard_FMX (TsgcFMXWizard, TsgcFMXWizardPage, TsgcFMXStepper)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcWizard
dclsgcWizard	디자인 타임, VCL	**SGC Wizard** 팔레트 페이지에 등록하는 컴포넌트는 TsgcVCLWizard 및 TsgcVCLStepper이며, 페이지를 추가하는 컴포넌트 편집기도 등록합니다.	designide, rtl, vcl, sgcWizard, sgcWizardVCL
dclsgcWizardFMX	디자인 타임, FireMonkey, XE6부터	**SGC Wizard** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXWizard 및 TsgcFMXStepper이며, 페이지를 추가하는 컴포넌트 편집기도 등록합니다.	designide, rtl, fmx, sgcWizard, sgcWizardFMX, dclsgcWizard

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcWizardFMX에는 dclsgcWizard가 필요하므로 dclsgcWizard를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcWizard	sgcWizardVCL	sgcWizardFMX	dclsgcWizard, dclsgcWizardFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcWizard, sgcWizardVCL	sgcWizardFMX	dclsgcWizard, dclsgcWizardFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcWizard만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcWizard만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcWizardC13.cbproj, sgcWizardVCLC13.cbproj, sgcWizardFMXC13.cbproj, dclsgcWizardC13.cbproj, dclsgcWizardFMXC13.cbproj 입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXWizard, TsgcFMXWizardPage 및 TsgcFMXStepper)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcWizardVCL은 Windows 전용입니다.

- 컨트롤을 폼에 놓거나 해당 유닛(sgcWizard_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcWizard는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 모델과, 헤드리스 PNG 및 SVG 출력을 갖춘 렌더러입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.
페이지는 팔레트에 없으며 **Add Page** 동사로 추가합니다. 디자인 타임을 참조하십시오.

관련 항목

- 에디션
- 아키텍처
- 문제 해결

빠른 시작 VCL

디자인 타임

- 폼 위에 **SGC Wizard** 팔레트 페이지의 **TsgcVCLWizard**를 놓고 **Align**을 alClient로 설정합니다.
- 마법사를 마우스 오른쪽 버튼으로 클릭하고 단계마다 한 번씩 **Add Page**를 선택합니다. 각 페이지의 **Caption**과 **Description**을 설정합니다. 이 값이 해당 단계의 캡션과 설명입니다.
- 각 페이지에 컨트롤을 놓습니다. 디자이너에서 다른 페이지를 표시하려면 컨텍스트 메뉴에서 **Next Page**와 **Previous Page**를 선택하거나 스테퍼 밴드의 단계를 클릭합니다.
- 페이지를 검증하고 페이지가 완전하지 않으면 aAllow를 False로 설정하는 **OnBeforePageChange** 핸들러와, 데이터를 사용하는 **OnFinish** 핸들러를 작성합니다.
- 애플리케이션을 실행합니다. Next와 Enter는 앞으로, Back은 뒤로 이동하고, Finish는 마지막 페이지에 나타나며, Cancel 또는 Escape는 OnCancel을 발생시킵니다.

코드에서

같은 마법사를 실행 시 생성한 예제입니다. 검색 경로에 `sgcUI\Source`가 있으면 프로그램은 그대로 컴파일됩니다.

관련 항목

- TsgcVCLWizard
- TsgcVCLWizardPage
- 유효성 검사
- 탐색과 건너뛴 페이지

빠른 시작 FMX

FireMonkey 컨트롤에는 Delphi XE6 이상이 필요합니다. VCL 컨트롤과 똑같이 사용합니다. **TsgcFMXWizard**를 놓고, **Add Page** 동사로 페이지를 추가하고, 각 페이지에 컨트롤을 놓은 다음, **OnBeforePageChange**와 **OnFinish**를 처리합니다. 페이지는 마법사 안에서 Client로 정렬됩니다. 버튼이나 완료된 단계를 탭하면 클릭과 같이 동작합니다.

코드에서

관련 항목

- TsgcFMXWizard
- TsgcFMXWizardPage
- 플랫폼

빠른 시작 C++Builder

사용 중인 C++Builder 버전용 sgcWizard 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 및 라이브러리 경로에 추가한 다음, Delphi와 똑같이 **SGC Wizard** 팔레트 페이지의 컴포넌트를 사용합니다. 설치 프로그램이 두 경로를 IDE에 추가합니다. sgcWizard와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있으므로, sgcUICore에는 별도의 경로가 필요하지 않습니다.

코드에서

OnBeforePageChange와 OnFinish는 클로저 이벤트이므로, 핸들러는 클래스의 __fastcall 멤버 함수입니다. var 매개변수 aAllow는 bool 참조입니다.

관련 항목

- TsgcVCLWizard
- 빠른 시작 VCL
- 데모

데모

데모는 설치 폴더의 **Demos** 폴더에 있으며, Delphi 프로젝트는 Demos\Delphi 아래에, VCL 데모의 C++Builder 포트는 Demos\CBuilder 아래에 있습니다. 모든 데모 폴더에는 데모가 하는 일과 동작 방식을 설명하는 readme.html이 있습니다.

데모	구성 요소	보여 주는 내용
Onboarding	TsgcVCLWizard, TsgcVCLWizardPage	Welcome, Account, Options, Summary의 네 페이지로 이루어진 계정 마법사입니다. OnBeforePageChange가 Account 페이지의 이름과 이메일을 검사하고 유효하지 않은 동안 Next를 거부하며, 확인란으로 Options 페이지를 건너뛰고, OnPageChanged가 요약을 채우고 OnFinish가 이를 표시합니다. 라이트 및 다크 전환 스위치와 StepperPosition 콤보(top, left, none)가 있습니다. Onboarding.exe /test result.txt는 마법사를 구동하고 검사마다 PASS 또는 FAIL 한 줄을 기록합니다. Delphi 및 C++Builder.
Stepper	TsgcVCLStepper	가로 하나, 세로 하나로 된 두 개의 독립 스테퍼이며 동일한 다섯 단계를 가집니다. ActiveIndex를 설정하면 그 앞의 단계가 완료되고, 버튼으로 활성 단계를 wssError 또는 wssSkipped로 표시하며, 완료된 단계를 클릭하면 OnStepClick을 통해 두 스테퍼가 모두 그 단계로 돌아갑니다. 라이트 및 다크 전환 스위치가 있습니다. Delphi 및 C++Builder.
OnboardingFMX	TsgcFMXWizard, TsgcFMXWizardPage	FireMonkey로 구현한 Onboarding 흐름입니다. 동일한 유효성 검사, 동일하게 건너뛰는 Options 페이지, Finish 시의 요약, 다크 테마 확인란, 동일한 /test 및 /snapshot 스위치를 제공합니다. Delphi.

C++Builder 데모는 sgcWizard_*.hpp와 sgcUICore_*.hpp 헤더를 포함하고 C++Builder 런타임 패키지의 정적 라이브러리를 링크합니다. Win32와 Win64에서 클래식 C++ 컴파일러로 빌드합니다.

관련 항목

- 빠른 시작 VCL
- 유효성 검사
- TsgcVCLStepper

TsgcVCLWizard

유닛: sgcWizard_VCL
TsgcVCLWizard는 Windows 전용 VCL 마법사입니다. 페이지는 TsgcVCLWizardPage 자식 컨트롤이고, 스테퍼 밴드는 페이지마다 하나의 단계를 표시하며, 버튼 바가 페이지 사이를 이동합니다. TCustomControl을 상속하며 Align, Anchors, Constraints, Enabled, Font, ParentFont, ShowHint, ParentShowHint, TabOrder, TabStop, Visible을 게시합니다.

속성

속성	형식	설명
ActivePageIndex	Integer	보이는 페이지의 인덱스. 코드로 설정하면 이벤트 없이 페이지가 바로 전환됩니다. 기본값 0.
ActivePage	페이지 클래스	public. 보이는 페이지이며 없으면 nil입니다. 설정하면 이벤트 없이 페이지가 전환됩니다.
PageCount	Integer	public, 읽기 전용. 페이지의 수.
Pages[Index]	페이지 클래스	public, 읽기 전용. 컨트롤 순서대로의 페이지들.
StepperPosition	TsgcWizardStepperPosition	wspTop(기본값)은 위쪽에 가로 스테퍼 밴드를, wspLeft는 왼쪽에 세로 밴드를 표시하고, wspNone은 밴드를 숨깁니다. 스테퍼 레이아웃을 참조하십시오.
BackCaption, NextCaption, FinishCaption, CancelCaption	string	버튼의 캡션. 기본값은 Back, Next, Finish, Cancel입니다. 변경된 경우에만 저장됩니다. 버튼 캡션을 참조하십시오.
ShowCancel	Boolean	Cancel 버튼을 표시합니다. 기본값 True.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. 마법사가 이를 밴드, 버튼 바, 페이지의 자식 컨트롤에 전달합니다. 테마를 참조하십시오.
AccentColor	TsgcUIColor	활성 및 완료된 단계, 완료된 연결선, 기본 버튼에 사용하는 $AARRGGBB 강조 색. 0(기본값)이면 테마의 강조 색을 사용합니다.
Stepper	스테퍼 클래스	public, 읽기 전용. 내부 스테퍼 밴드.
ButtonBar	버튼 바 클래스	public, 읽기 전용. 내부 버튼 바. ButtonCount, Buttons, ButtonRect, ButtonAt는 그려진 그대로의 버튼을 설명합니다.

메서드

메서드	반환값	설명
Next	Boolean	건너뛰지 않은 다음 페이지로 이동하며 OnBeforePageChange와 OnPageChanged를 발생시킵니다. 마지막 페이지에서는 Finish를 호출합니다. 페이지가 바뀌었거나 마법사가 끝났으면 True.
Back	Boolean	건너뛰지 않은 이전 페이지로 같은 이벤트와 함께 이동합니다.

GoToPage(aIndex)	Boolean	건너뛰지 않은 임의의 페이지로 같은 이벤트와 함께 이동합니다. 건너뛴 페이지이거나 범위를 벗어난 인덱스이면 False.
Finish	Boolean	aToIndex = PageCount로 OnBeforePageChange를 발생시키고, 허용되면 OnFinish를 발생시킵니다.
Cancel		OnCancel을 발생시킵니다.
NextPageIndex(aIndex), PrevPageIndex(aIndex)	Integer	aIndex 다음의 건너뛰지 않은 다음 또는 이전 페이지이며, 없으면 -1입니다.
IsLastPage	Boolean	활성 페이지 뒤에 도달할 수 있는 페이지가 없으면 True이며, 이때 Next는 Finish로 표시됩니다.
IndexOfPage(aPage)	Integer	페이지의 인덱스이며, 이 마법사의 페이지가 아니면 -1입니다.
AddPage(aCaption, aOwner = nil)	페이지 클래스	해당 Caption으로 페이지를 만들고 소유자를 aOwner로 하며(nil이면 마법사의 소유자를 사용), 맨 뒤에 배치합니다.

이벤트

이벤트	형식	설명
OnBeforePageChange	TsgcWizardPageChangeEvent	Next, Back, GoToPage, Finish 또는 단계 클릭으로 인한 모든 변경 전에 발생합니다. aFromIndex, aToIndex, var aAllow: aAllow를 False로 설정하면 현재 페이지에 머뭅니다. aToIndex = PageCount는 Finish를 의미합니다. 유효성 검사를 참조하십시오.
OnPageChanged	TNotifyEvent	페이지가 변경된 후.
OnFinish	TNotifyEvent	Finish가 허용된 후. Finish와 Cancel을 참조하십시오.
OnCancel	TNotifyEvent	Cancel, Cancel 버튼 또는 Escape.

예제

관련 항목

- TsgcVCLWizardPage
- TsgcFMXWizard
- 탐색과 건너뛴 페이지
- 키보드

TsgcVCLWizardPage

유닛: sgcWizard_VCL
TsgcVCLWizardPage는 TsgcVCLWizard의 페이지입니다. TCustomControl을 상속하고 자식 컨트롤을 받을 수 있으며 Enabled, Font, ParentFont, Hint, ShowHint, ParentShowHint, PopupMenu, OnEnter, OnExit, OnResize도 게시합니다. 페이지는 마법사에만 배치할 수 있으며, Parent를 다른 컨트롤로 설정하면 EsgcWizardError가 발생합니다.

속성

속성	형식	설명
Caption	string	페이지의 단계 캡션.
Description	string	단계의 설명이며 밴드의 ShowDescriptions가 True인 동안 캡션 아래에 그려지고, 단계의 힌트이기도 합니다.
Skip	Boolean	Next와 Back이 이 페이지를 건너뛰고 밴드는 해당 단계를 wssSkipped로 그립니다. 기본값 False. 탐색과 건너뛴 페이지를 참조하십시오.
StepGlyph	string	단계 원 안에 그려지는 짧은 텍스트. 비어 있으면(기본값) 단계 번호를 그립니다.
Visible	Boolean	마법사가 관리합니다. 활성 페이지만 보입니다. 저장되지 않습니다.
Wizard	마법사 클래스	public, 읽기 전용. 페이지의 마법사이며 없으면 nil입니다.
PageIndex	Integer	public, 읽기 전용. 마법사의 페이지들 중 위치이며, 마법사에 속하지 않으면 -1입니다.

마법사는 테마의 Surface 색으로 페이지를 그리고 페이지의 직접 자식 컨트롤에 테마를 적용합니다. 테마를 참조하십시오.

예제

관련 항목

- TsgcVCLWizard
- 디자인 타임
- 문제 해결

TsgcFMXWizard

유닛: sgcWizard_FMX
TsgcFMXWizard는 Delphi XE6 이상의 FireMonkey 마법사입니다. TsgcVCLWizard와 속성, 메서드, 이벤트가 같습니다. 페이지는 Client로 정렬된 TsgcFMXWizardPage 자식이며, 스테퍼 밴드는 MostTop 또는 MostLeft로, 버튼 바는 MostBottom으로 정렬됩니다. 위치와 크기는 FMX 논리 단위입니다. Align, Anchors, CanFocus, ClipChildren, Enabled, Height, HitTest, Margins, Opacity, Padding, PopupMenu, Position, Size, TabOrder, TabStop, Visible, Width를 게시하고, 10.1 Berlin부터는 ShowHint와 ParentShowHint를, 그리고 OnEnter, OnExit, OnKeyDown, OnKeyUp, OnResize를 게시합니다.

속성

속성	형식	설명
ActivePageIndex	Integer	보이는 페이지의 인덱스. 코드로 설정하면 이벤트 없이 페이지가 바로 전환됩니다. 기본값 0.
ActivePage	페이지 클래스	public. 보이는 페이지이며 없으면 nil입니다. 설정하면 이벤트 없이 페이지가 전환됩니다.
PageCount	Integer	public, 읽기 전용. 페이지의 수.
Pages[Index]	페이지 클래스	public, 읽기 전용. 컨트롤 순서대로의 페이지들.
StepperPosition	TsgcWizardStepperPosition	wspTop(기본값)은 위쪽에 가로 스테퍼 밴드를, wspLeft는 왼쪽에 세로 밴드를 표시하고, wspNone은 밴드를 숨깁니다. 스테퍼 레이아웃을 참조하십시오.
BackCaption, NextCaption, FinishCaption, CancelCaption	string	버튼의 캡션. 기본값은 Back, Next, Finish, Cancel입니다. 변경된 경우에만 저장됩니다. 버튼 캡션을 참조하십시오.
ShowCancel	Boolean	Cancel 버튼을 표시합니다. 기본값 True.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. 마법사가 이를 밴드, 버튼 바, 페이지의 자식 컨트롤에 전달합니다. 테마를 참조하십시오.
AccentColor	TsgcUIColor	활성 및 완료된 단계, 완료된 연결선, 기본 버튼에 사용하는 $AARRGGBB 강조 색. 0(기본값)이면 테마의 강조 색을 사용합니다.
Stepper	스테퍼 클래스	public, 읽기 전용. 내부 스테퍼 밴드.
ButtonBar	버튼 바 클래스	public, 읽기 전용. 내부 버튼 바. ButtonCount, Buttons, ButtonRect, ButtonAt는 그려진 그대로의 버튼을 설명합니다.

메서드

메서드	반환값	설명

Next	Boolean	건너뛰지 않은 다음 페이지로 이동하며 OnBeforePageChange와 OnPageChanged를 발생시킵니다. 마지막 페이지에서는 Finish를 호출합니다. 페이지가 바뀌었거나 마법사가 끝났으면 True.
Back	Boolean	건너뛰지 않은 이전 페이지로 같은 이벤트와 함께 이동합니다.
GoToPage(aIndex)	Boolean	건너뛰지 않은 임의의 페이지로 같은 이벤트와 함께 이동합니다. 건너뛴 페이지이거나 범위를 벗어난 인덱스이면 False.
Finish	Boolean	aToIndex = PageCount로 OnBeforePageChange를 발생시키고, 허용되면 OnFinish를 발생시킵니다.
Cancel		OnCancel을 발생시킵니다.
NextPageIndex(aIndex), PrevPageIndex(aIndex)	Integer	aIndex 다음의 건너뛰지 않은 다음 또는 이전 페이지이며, 없으면 -1입니다.
IsLastPage	Boolean	활성 페이지 뒤에 도달할 수 있는 페이지가 없으면 True이며, 이때 Next는 Finish로 표시됩니다.
IndexOfPage(aPage)	Integer	페이지의 인덱스이며, 이 마법사의 페이지가 아니면 -1입니다.
AddPage(aCaption, aOwner = nil)	페이지 클래스	해당 Caption으로 페이지를 만들고 소유자를 aOwner로 하며(nil이면 마법사의 소유자를 사용), 맨 뒤에 배치합니다.

이벤트

이벤트	형식	설명
OnBeforePageChange	TsgcWizardPageChangeEvent	Next, Back, GoToPage, Finish 또는 단계 클릭으로 인한 모든 변경 전에 발생합니다. aFromIndex, aToIndex, var aAllow: aAllow를 False로 설정하면 현재 페이지에 머뭅니다. aToIndex = PageCount는 Finish를 의미합니다. 유효성 검사를 참조하십시오.
OnPageChanged	TNotifyEvent	페이지가 변경된 후.
OnFinish	TNotifyEvent	Finish가 허용된 후. Finish와 Cancel을 참조하십시오.
OnCancel	TNotifyEvent	Cancel, Cancel 버튼 또는 Escape.

예제

관련 항목

- TsgcFMXWizardPage
- 빠른 시작 FMX
- 테마

TsgcFMXWizardPage

유닛: sgcWizard_FMX
TsgcFMXWizardPage는 TsgcFMXWizard의 페이지이며 그 안에서 Client로 정렬됩니다. ClipChildren, Enabled, HitTest, Padding, PopupMenu, OnEnter, OnExit, OnResize도 게시하고, 10.1 Berlin부터는 ShowHint와 ParentShowHint도 게시합니다. 페이지는 마법사에만 배치할 수 있으며, Parent를 다른 객체로 설정하면 EsgcWizardError가 발생합니다.

속성

속성	형식	설명
Caption	string	페이지의 단계 캡션.
Description	string	단계의 설명이며 밴드의 ShowDescriptions가 True인 동안 캡션 아래에 그려지고, 단계의 힌트이기도 합니다.
Skip	Boolean	Next와 Back이 이 페이지를 건너뛰고 밴드는 해당 단계를 wssSkipped로 그립니다. 기본값 False. 탐색과 건너뛴 페이지를 참조하십시오.
StepGlyph	string	단계 원 안에 그려지는 짧은 텍스트. 비어 있으면(기본값) 단계 번호를 그립니다.
Visible	Boolean	마법사가 관리합니다. 활성 페이지만 보입니다. 저장되지 않습니다.
Wizard	마법사 클래스	public, 읽기 전용. 페이지의 마법사이며 없으면 nil입니다.
PageIndex	Integer	public, 읽기 전용. 마법사의 페이지들 중 위치이며, 마법사에 속하지 않으면 -1입니다.

예제

관련 항목

- TsgcFMXWizard
- TsgcVCLWizardPage
- 테마

TsgcVCLStepper

유닛: sgcWizard_VCL
TsgcVCLStepper는 단독으로 사용할 수 있는 VCL 단계 표시기입니다. 단계를 연결선으로 이어진 번호 붙은 원으로 가로 또는 세로로 표시합니다. 단계의 힌트는 그 Description입니다. 컨트롤에 포커스가 있고 Clickable이 True인 동안, Left와 Right(세로일 때는 Up과 Down)는 같은 OnStepClick 거부 절차를 거쳐 이전 또는 다음의 보이고 활성화된 단계로 이동합니다. Align, Anchors, Constraints, Enabled, ShowHint, ParentShowHint, TabOrder, TabStop, Visible도 게시합니다.

속성

속성	형식	설명
Steps	TsgcWizardSteps	단계들.
Orientation	TsgcWizardOrientation	woHorizontal(기본값) 또는 woVertical.
ActiveIndex	Integer	활성 단계. 기본값 0. 코드로 설정하면 OnActiveChange가 발생하지 않습니다.
AutoStates	Boolean	기본값 True: ActiveIndex를 설정하면 그 앞의 보이는 단계는 wssError 또는 wssSkipped가 아닌 한 완료 처리되고, 그 뒤의 단계는 wssPending으로 설정됩니다. 단계 상태를 참조하십시오.
ShowDescriptions	Boolean	각 단계의 Description을 캡션 아래에 그립니다. 기본값 True.
CircleSize	Integer	단계 원의 지름이며 논리 픽셀 단위입니다. 기본값 28.
ConnectorThickness	Integer	연결선의 두께이며 논리 픽셀 단위입니다. 기본값 2.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	$AARRGGBB 형식의 강조 색이며, 0(기본값)이면 테마의 강조색을 사용합니다.
Clickable	Boolean	클릭이나 화살표 키로 완료된 단계, 또는 활성 단계보다 앞의 오류 단계를 활성화할 수 있습니다. 기본값 True.
Model	TsgcStepperModel	public, 읽기 전용. 컨트롤의 모델.
Wizard	마법사 클래스	public, 읽기 전용. 스테퍼를 밴드로 호스팅하는 마법사이며, 독립 스테퍼이면 nil입니다.

메서드

메서드	반환값	설명
StepAt(X, Y)	Integer	클라이언트 좌표의 한 지점 아래에 있는 단계이며, 없으면 -1입니다.
StepRect(aIndex)	TRect(FMX에서는 TRectF)	클라이언트 좌표로 나타낸 단계의 원과 레이블이며, 단계가 표시되지 않으면 비어 있습니다.

이벤트

이벤트	형식	설명
OnStepClick	TsgcWizardStepClickEvent	클릭할 수 있는 단계가 클릭되었거나 화살표 키가 그 단계로 이동합니다. aIndex와 var aAllow: aAllow를 False로 설정하면 활성 단계가 유지됩니다.
OnActiveChange	TNotifyEvent	클릭이나 키로 활성 단계가 변경되었습니다. ActiveIndex를 코드로 설정할 때는 발생하지 않습니다.

예제

관련 항목

- TsgcFMXStepper
- 단계 상태
- 스테퍼 레이아웃

TsgcFMXStepper

유닛: sgcWizard_FMX
TsgcFMXStepper는 Delphi XE6 이상의 FireMonkey 단계 표시기이며, TsgcVCLStepper와 속성, 메서드, 이벤트가 같습니다. 10.1 Berlin부터는 컨트롤의 Hint가 마우스 아래 단계의 Description을 따릅니다. Align, Anchors, CanFocus, Enabled, Height, HitTest, Margins, Opacity, Padding, PopupMenu, Position, Size, TabOrder, TabStop, Visible, Width와 OnEnter, OnExit, OnKeyDown, OnKeyUp, OnMouseDown, OnMouseMove, OnMouseUp, OnResize도 게시합니다.

속성

속성	형식	설명
Steps	TsgcWizardSteps	단계들.
Orientation	TsgcWizardOrientation	woHorizontal(기본값) 또는 woVertical.
ActiveIndex	Integer	활성 단계. 기본값 0. 코드로 설정하면 OnActiveChange가 발생하지 않습니다.
AutoStates	Boolean	기본값 True: ActiveIndex를 설정하면 그 앞의 보이는 단계는 wssError 또는 wssSkipped가 아닌 한 완료 처리되고, 그 뒤의 단계는 wssPending으로 설정됩니다. 단계 상태를 참조하십시오.
ShowDescriptions	Boolean	각 단계의 Description을 캡션 아래에 그립니다. 기본값 True.
CircleSize	Integer	단계 원의 지름이며 논리 픽셀 단위입니다. 기본값 28.
ConnectorThickness	Integer	연결선의 두께이며 논리 픽셀 단위입니다. 기본값 2.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	$AARRGGBB 형식의 강조 색이며, 0(기본값)이면 테마의 강조색을 사용합니다.
Clickable	Boolean	클릭이나 화살표 키로 완료된 단계, 또는 활성 단계보다 앞의 오류 단계를 활성화할 수 있습니다. 기본값 True.
Model	TsgcStepperModel	public, 읽기 전용. 컨트롤의 모델.
Wizard	마법사 클래스	public, 읽기 전용. 스테퍼를 밴드로 호스팅하는 마법사이며, 독립 스테퍼이면 nil입니다.

메서드

메서드	반환값	설명
StepAt(X, Y)	Integer	클라이언트 좌표의 한 지점 아래에 있는 단계이며, 없으면 -1입니다.
StepRect(aIndex)	TRect(FMX에서는 TRectF)	클라이언트 좌표로 나타낸 단계의 원과 레이블이며, 단계가 표시되지 않으면 비어 있습니다.

이벤트

이벤트	형식	설명
OnStepClick	TsgcWizardStepClickEvent	클릭할 수 있는 단계가 클릭되었거나 화살표 키가 그 단계로 이동합니다. aIndex와 var aAllow: aAllow를 False로 설정하면 활성 단계가 유지됩니다.
OnActiveChange	TNotifyEvent	클릭이나 키로 활성 단계가 변경되었습니다. ActiveIndex를 코드로 설정할 때는 발생하지 않습니다.

예제

관련 항목

- TsgcVCLStepper
- 단계 상태
- 플랫폼

TsgcWizardStep

유닛: sgcWizard_Model
TsgcWizardStep은 스테퍼의 한 단계이며 TsgcWizardSteps의 항목입니다. 마법사에서는 밴드의 단계가 페이지로부터 생성되므로, 애플리케이션은 대신 페이지 속성을 설정합니다.

속성

속성	형식	설명
Caption	string	원 아래(가로) 또는 오른쪽(세로)에 그려지는 캡션.
Description	string	레이블의 둘째 줄이며 ShowDescriptions가 True인 동안 그려지고, 단계의 힌트이기도 합니다.
Glyph	string	원 안에 그려지는 짧은 텍스트. 비어 있으면 단계 번호를 그립니다.
State	TsgcWizardStepState	wssPending(기본값), wssActive, wssCompleted, wssError 또는 wssSkipped. 단계 상태를 참조하십시오.
Enabled	Boolean	비활성 단계는 흐릿한 색으로 그려지며 클릭할 수 없습니다. 기본값 True.
Visible	Boolean	숨겨진 단계는 자리와 번호를 받지 못하지만 상태는 유지합니다. 기본값 True.
Tag	NativeInt (XE2 이전에는 Integer)	애플리케이션이 자유롭게 사용합니다. 기본값 0.
Shown, Number, CircleRect, LabelRect	Boolean, Integer, TsgcUIRect	public, 읽기 전용. 마지막 Layout의 결과: 단계가 자리를 받았는지 여부, 보이는 단계 중 1부터 시작하는 번호, 그리고 디바이스 픽셀 단위의 원과 레이블 사각형.

메서드

메서드	반환값	설명
CircleText	string	원 안에 그려지는 텍스트: Glyph, 또는 Glyph가 비어 있으면 Number.

예제

관련 항목

- TsgcWizardSteps
- TsgcStepperModel

TsgcWizardSteps

유닛: sgcWizard_Model
TsgcWizardSteps는 TsgcWizardStep 항목의 소유 컬렉션이며, 스테퍼와 TsgcStepperModel의 Steps 속성 뒤에 있습니다. 디자인 타임에는 표준 컬렉션 편집기로 편집합니다.

메서드

메서드	반환값	설명
Add	TsgcWizardStep	빈 단계를 추가합니다.
AddStep(aCaption, aDescription = '', aGlyph = '')	TsgcWizardStep	캡션, 설명, 글리프를 지정하여 단계를 추가하고 이를 반환합니다.
Items[Index]	TsgcWizardStep	기본 인덱스 속성.

단계를 추가하거나 제거하면 AutoStates가 True인 동안 자동 상태가 다시 계산됩니다.

예제

관련 항목

- TsgcWizardStep
- TsgcVCLStepper

TsgcStepperModel

유닛: sgcWizard_Model
TsgcStepperModel은 스테퍼의 프레임워크 중립 모델입니다. 상태를 가진 단계, 방향, 활성 단계, 핫 스텝, 그리고 원, 레이블, 연결선의 레이아웃과 히트 테스트를 담고 있습니다. VCL 및 FMX 스테퍼가 각각 하나씩 소유하며 그 속성을 게시하고, 헤드리스 렌더링은 창 없이 이를 그립니다. 메인 스레드에서 사용하도록 만들어졌습니다.

속성

속성	형식	설명
Steps, Orientation, ActiveIndex, AutoStates, ShowDescriptions, CircleSize, ConnectorThickness, Theme, AccentColor, Clickable		스테퍼의 게시된 속성입니다. TsgcVCLStepper를 참조하십시오.
HotIndex	Integer	마우스 아래에 있는 단계이며 컨트롤이 설정합니다. 클릭할 수 있는 핫 스텝에는 은은한 오버레이가 적용됩니다.
ConnectorCount, Connectors[Index]	Integer, TsgcWizardConnector	마지막 Layout의 연결선: FromIndex, ToIndex, 그리고 디바이스 픽셀 단위의 선 X1, Y1, X2, Y2.
LayoutWidth, LayoutHeight, LayoutScale	Integer, Double	마지막 Layout의 크기와 배율.
OnChange	TNotifyEvent	모델의 모든 변경. Layout은 이 이벤트를 발생시키지 않습니다.

메서드

메서드	반환값	설명
Layout(aWidth, aHeight, aScale = 1) 및 측정 콜백이나 캔버스를 받는 오버로드		aWidth x aHeight 디바이스 픽셀 크기의 스테퍼에 대한 단계와 연결선의 사각형을 계산합니다. 모든 논리 크기에 aScale이 곱해집니다. 측정 콜백이나 캔버스가 없으면 레이블 너비를 추정합니다.
HitTest(aX, aY)	Integer	마지막 Layout에서 원이나 레이블이 그 지점을 포함하는 단계이며, 없으면 -1입니다. 비활성 단계도 보고됩니다.
IsStepClickable(aIndex)	Boolean	Clickable이 켜져 있는 동안, 보이고 활성화되어 있으며 활성 단계가 아닌 단계는 wssCompleted이거나 활성 단계보다 앞의 wssError이면 클릭할 수 있습니다.
IsConnectorDone(aToIndex)	Boolean	연결선 규칙입니다. 단계 상태를 참조하십시오.
ApplyAutoStates		AutoStates 값과 관계없이 자동 상태를 지금 적용합니다.
VisibleCount	Integer	보이는 단계의 수.
NextStep(aIndex), PrevStep(aIndex)	Integer	다음 또는 이전의 보이고 활성화된 단계이며, 없으면 -1입니다.

GetAccent	TsgcUIColor	사용 중인 강조 색: AccentColor, 또는 0이면 테마의 강조 색.

예제

아래 콘솔 프로그램은 Clickable: Cart, Visible steps: 3, After Shipping: 2, Connectors: 2, Into Shipping done: TRUE를 출력합니다.

관련 항목

- TsgcWizardStep
- 헤드리스 렌더링
- 단계 상태

탐색과 건너뛴 페이지

사용자는 버튼 바, 키보드, 스테퍼 밴드의 완료된 단계를 통해 마법사를 이동합니다. 애플리케이션도 같은 메서드를 사용합니다.

동작	효과
Next, Next 버튼, Enter	건너뛰지 않은 다음 페이지로 이동합니다. 마지막 페이지에서는 Finish를 호출합니다.
Back, Back 버튼	건너뛰지 않은 이전 페이지로 이동합니다. 도달할 수 있는 이전 페이지가 없으면 Back 버튼은 비활성화됩니다.
GoToPage(aIndex)	건너뛰지 않은 임의의 페이지로 이동합니다.
밴드의 완료된 단계 클릭	해당 페이지에 대해 GoToPage를 호출합니다.
코드로 설정한 ActivePageIndex 또는 ActivePage	이벤트 없이 페이지를 바로 전환합니다.

직접 전환을 제외한 모든 이동은 이를 거부할 수 있는 OnBeforePageChange를 발생시킨 다음 OnPageChanged를 발생시킵니다. 포커스가 마법사 안에 있었다면 페이지가 바뀔 때 포커스는 새 페이지의 첫 번째 컨트롤이나 마법사로 이동합니다.

건너뛴 페이지

Skip 속성이 True인 페이지는 마법사에 그대로 남아 있지만, Next와 Back은 이를 건너뛰고 스테퍼 밴드는 해당 단계를 점선 윤곽의 건너뜀 상태로 그립니다. Skip은 언제든 바꿀 수 있으며, 예를 들어 앞 페이지의 확인란으로 바꿀 수 있습니다. NextPageIndex와 PrevPageIndex는 Next와 Back이 이동할 페이지를 반환하고, IsLastPage는 활성 페이지 뒤에 도달할 수 있는 페이지가 없을 때 True이며 이때 Next 버튼은 Finish 캡션을 표시합니다.

예제

관련 항목

- TsgcVCLWizard
- 유효성 검사
- Finish와 Cancel

유효성 검사

OnBeforePageChange는 Next, Back, GoToPage, Finish 또는 단계 클릭으로 이루어지는 모든 페이지 변경 전에 발생합니다. 활성 페이지인 aFromIndex, 곧 표시될 페이지인 aToIndex, 그리고 True로 시작하는 var aAllow를 받습니다. 필수 필드가 비어 있는 경우처럼 현재 페이지에 머물려면 aAllow를 False로 설정하십시오.

- **aToIndex > aFromIndex**는 Finish를 포함한 모든 앞으로 가는 이동이므로, 사용자가 무엇을 눌렀든 한 번의 검사로 페이지를 검증할 수 있습니다.
- **aToIndex = PageCount**는 사용자가 Finish를 눌렀다는 의미입니다. Finish는 PageCount를 대상으로 이벤트를 발생시키며, 이것은 실제 페이지 인덱스가 될 수 없습니다.
- 뒤로 가는 이동은 보통 검증이 필요하지 않습니다. aToIndex < aFromIndex를 검사하여 통과시키십시오.
- 변경이 거부되면 다른 일은 일어나지 않습니다. 페이지, 밴드, 버튼은 그대로 유지되므로, 이유를 표시하고 포커스를 해당 필드로 옮기십시오.

예제

관련 항목

- 탐색과 건너뛴 페이지
- Finish와 Cancel
- 데모

Finish와 Cancel

마지막 페이지에서는 Next 버튼이 **FinishCaption**을 표시하며 기본 버튼이 됩니다. 이 버튼을 누르거나, Enter를 누르거나, **Finish**를 호출하면 aToIndex = PageCount로 OnBeforePageChange가 발생하고, 허용되면 **OnFinish**가 발생합니다. 마법사는 스스로 아무것도 닫지 않습니다. 예를 들어 데이터를 저장하고 폼을 닫거나 대화 상자의 ModalResult를 설정하는 등, OnFinish가 무엇을 할지 결정합니다.
Cancel 버튼, Escape 키, **Cancel** 호출은 **OnCancel**을 발생시킵니다. 거부 기능이 없고 페이지를 바꾸지 않습니다. Cancel 버튼을 숨기려면 **ShowCancel**을 False로 설정하십시오.

예제

관련 항목

- 유효성 검사
- 키보드
- 버튼 캡션

키보드

마법사나 그 안의 컨트롤에 포커스가 있는 동안 **Enter**는 Next(마지막 페이지에서는 Finish)이고 **Escape**는 Cancel입니다. 키를 직접 처리하려는 컨트롤이 우선합니다. TMemo는 Enter를 줄 바꿈에 사용하고, 포커스가 있는 버튼이나 Default 또는 Cancel이 설정된 TButton은 평소대로 키를 처리합니다.

프레임워크	키가 전달되는 방식
VCL	부모 폼의 CM_DIALOGKEY를 통하거나, 마법사 자체에 포커스가 있으면 KeyDown을 통합니다.
FireMonkey	폼의 대화 상자 키 체인(DialogKey와 AfterDialogKey)을 통하거나, 마법사 자체에 포커스가 있으면 KeyDown을 통합니다.

페이지가 바뀐 후 포커스는 새 페이지의 첫 번째 컨트롤로 이동하며, 이전에 마법사 안에 있었다면 마법사로 이동합니다.

스테퍼 키

포커스가 있는 스테퍼는 Left와 Right(세로일 때는 Up과 Down)로 이전 또는 다음의 보이고 활성화된 단계로 이동합니다. 이 이동은 클릭과 같은 OnStepClick 거부 절차를 거치며 Clickable이 True인 동안에만 일어납니다.

관련 항목

- 탐색과 건너뛴 페이지
- Finish와 Cancel
- TsgcVCLStepper

단계 상태

모든 단계에는 그려지는 방식을 결정하는 **State**가 있습니다.

상태	그려지는 모양
wssPending	테두리 색의 윤곽에 단계 번호 또는 글리프가 흐린 텍스트 색으로 표시됩니다.
wssActive	은은한 강조 후광이 있는 강조 색 원판에 대비되는 색의 단계 텍스트. 캡션은 굵게 표시됩니다.
wssCompleted	체크 표시가 있는 강조 색 원판.
wssError	느낌표가 있는 빨간 원판과 빨간 캡션.
wssSkipped	흐린 텍스트 색의 단계 텍스트가 있는 점선 윤곽.

비활성 단계는 흐릿한 색으로 그려집니다.

자동 상태

AutoStates가 True(기본값)인 동안, ActiveIndex를 설정하거나, 단계를 추가 또는 제거하거나, AutoStates를 켤 때마다 활성 단계 앞의 보이는 단계는 wssError 또는 wssSkipped가 아닌 한 wssCompleted가 되고, 활성 단계는 wssActive가 되며, 그 뒤의 보이는 단계는 wssPending이 됩니다. 숨겨진 단계는 상태를 유지합니다. AutoStates가 False이면 모든 상태를 애플리케이션이 관리합니다. 마법사에서는 페이지와 그 Skip 속성으로 마법사가 상태를 설정하므로 밴드의 AutoStates는 False입니다.

연결선 규칙

연속된 두 보이는 단계 사이의 연결선은, 연결선이 향하는 단계가 wssCompleted 또는 wssActive이거나, 그 단계가 wssSkipped 또는 wssError이면서 이미 지나간 경우, 즉 이후의 보이는 단계 중 하나가 wssActive 또는 wssCompleted인 경우에 완료된 것으로 간주되어 강조 색으로 그려집니다. 그렇지 않으면 테두리 색으로 그려집니다. **Model.IsConnectorDone(aToIndex)**는 aToIndex 단계로 들어가는 연결선에 대한 이 규칙의 결과를 반환하므로, 흐름의 완료된 구간 중간에 있는 건너뛴 페이지가 강조선을 끊지 않습니다.

예제

관련 항목

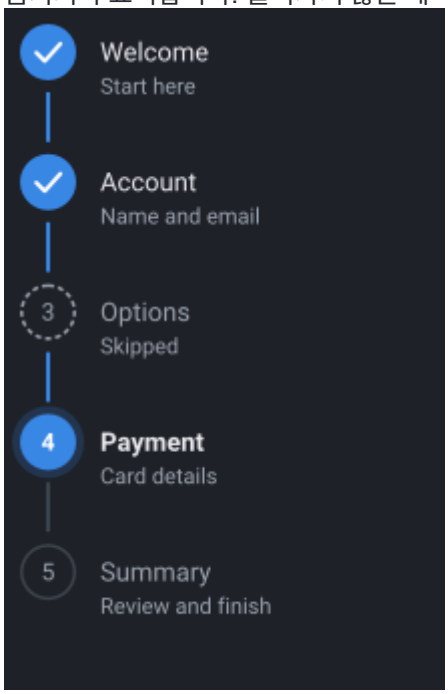
- TsgcWizardStep
- TsgcStepperModel
- 스테퍼 레이아웃

스테퍼 레이아웃

스테퍼는 **Orientation**으로 가로 또는 세로가 정해집니다. 보이는 단계만 자리를 받으며, 그 번호는 보이는 단계 중 1부터 시작하는 위치입니다.

- **woHorizontal**: 너비가 동일한 자리로 나뉘고, 원은 자리의 가운데에 놓이며, 캡션과 설명은 그 아래 가운데에 놓입니다. 블록은 세로로 가운데 정렬됩니다.
- **woVertical**: 높이가 동일한 행으로 나뉘고, 원은 행의 왼쪽 위에 놓이며, 레이블은 그 오른쪽에 있고 캡션은 원의 가운데에 맞춰집니다. 연결선은 원 사이를 아래로 이어집니다.

CircleSize(기본값 28)와 **ConnectorThickness**(기본값 2)는 논리 픽셀입니다. **ShowDescriptions**는 레이블의 둘째 줄을 숨기거나 표시합니다. 들어가지 않는 레이블 텍스트는 사용 가능한 공간에 맞춰 잘립니다.



마법사의 StepperPosition

마법사는 **StepperPosition**으로 밴드의 위치를 정합니다. wspTop(기본값)은 위쪽에 높이 80 논리 픽셀의 가로 밴드를, wspLeft는 왼쪽에 너비 200 논리 픽셀의 세로 밴드를 표시하고, wspNone은 밴드를 숨깁니다. 아래쪽 버튼 바의 높이는 52 논리 픽셀입니다.

예제

관련 항목

- 단계 상태
- TsgcVCLStepper
- TsgcVCLWizard

버튼 캡션과 현지화

버튼 바의 네 가지 캡션은 마법사의 게시된 속성인 **BackCaption**, **NextCaption**, **FinishCaption**, **CancelCaption**이며, 기본값은 Back, Next, Finish, Cancel입니다. 캡션은 기본값과 다를 때만 폼 파일에 저장되므로, 기본 캡션을 사용하는 폼은 새 기본값을 그대로 따라가며, 캡션은 IDE의 일반적인 리소스 도구로 번역하거나 애플리케이션의 번역 결과로 코드에서 설정할 수 있습니다. 버튼은 캡션에 맞춰 커지며, **ShowCancel**은 Cancel 버튼을 숨깁니다.
단계의 캡션은 페이지의 Caption과 Description이므로 페이지와 함께 번역됩니다.



예제

관련 항목

- Finish와 Cancel
- TsgcVCLWizard

테마

Theme은 sgcUICore 테마 토큰으로 색을 정합니다. utLight(기본값), utDark 또는 utCustom입니다. **AccentColor**는 테마의 강조 색을 $AARRGGBB 색으로 재정의하며, 활성 및 완료된 단계, 완료된 연결선, 기본 버튼에 사용됩니다. 0(기본값)은 테마의 강조 색을 유지합니다.
마법사는 Theme과 AccentColor를 스테퍼 밴드와 버튼 바에 전달하고, 테마의 Surface 색으로 자신과 페이지를 그리며, 모든 페이지의 직접 자식 컨트롤에 테마를 적용합니다. 이는 테마가 변경될 때, 마법사가 로드될 때, 페이지에 컨트롤이 추가될 때 일어납니다. 자체 Theme 속성이 있는 자식 컨트롤은 테마를 받습니다. 그 외의 자식은 Surface 색을 배경으로, Text 색을 글꼴 색으로 받으므로, TLabel, TEdit, TCheckBox는 어두운 페이지에서도 읽기 쉽게 유지됩니다.

- **FireMonkey**: TEdit, TMemo, TComboBox, TListBox, 날짜 컨트롤, 그리드 같은 스타일이 적용된 편집 컨트롤은 스타일이 자체 불투명 배경을 그리기 때문에 스타일의 색을 유지합니다. 텍스트 색은 배경색도 함께 설정된 경우에만 변경됩니다.
- **utCustom**은 자식 컨트롤을 건드리지 않습니다. VCL에서는 페이지가 부모의 색을 따르고, FireMonkey에서는 마법사도 페이지도 그려지지 않습니다.

테마 토큰은 sgcDashboard 및 다른 sgcUICore 제품의 것과 같으므로, 같은 Theme의 마법사와 대시보드는 서로 어울리며, sgcDashboard 타일에 호스팅된 마법사는 대시보드의 테마를 따릅니다.

예제

관련 항목

- 아키텍처
- 스테퍼 레이아웃
- 헤드리스 렌더링

디자인 타임

디자인 타임 패키지는 TsgcVCLWizard, TsgcVCLStepper, TsgcFMXWizard, TsgcFMXStepper를 **SGC Wizard** 팔레트 페이지에 등록합니다. 페이지는 팔레트에 없습니다. 마법사와 그 페이지의 컴포넌트 편집기는 컨텍스트 메뉴에 세 가지 동사를 추가합니다.

동사	효과
Add Page	클래스 이름을 딴 이름으로 새 페이지를 만들고 그 이름을 Caption으로 설정한 뒤, 마법사의 맨 뒤에 배치하고 표시하며 선택합니다.
Next Page	디자이너에서 다음 페이지를 표시합니다.
Previous Page	디자이너에서 이전 페이지를 표시합니다.

디자이너에서 스테퍼 밴드의 단계를 클릭해도 해당 페이지가 표시됩니다. 활성 페이지만 보이며, 디자이너에 표시되는 페이지는 폼 파일에 저장된 ActivePageIndex입니다. 스테퍼 밴드와 버튼 바는 내부 컨트롤이므로 폼 파일에 저장되지 않고 선택할 수 없습니다. 독립 스테퍼의 Steps는 표준 컬렉션 편집기로 편집합니다.

관련 항목

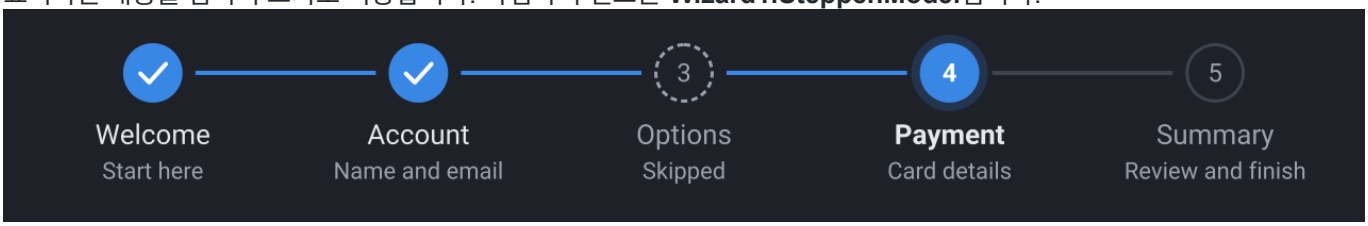
- 빠른 시작 VCL
- TsgcVCLWizardPage
- 플랫폼

헤드리스 렌더링

sgcWizard_Render의 렌더러는 창이나 VCL, FMX 없이 TsgcStepperModel과 버튼 바를 그립니다. 따라서 콘솔 애플리케이션, 서비스, 테스트, 웹 서버에서도 애플리케이션 문서나 시각적 테스트 등을 위해 같은 스테퍼를 이미지로 만들 수 있습니다. 이 유닛은 Linux64를 포함한 모든 플랫폼에서 빌드됩니다.

루틴	설명
sgcWizardSaveToPNG(aModel, aFileName 또는 aStream, aWidth, aHeight, aScale = 1)	PNG 파일 또는 스트림을 기록합니다. aWidth와 aHeight는 논리 픽셀이며 aScale이 곱해집니다. 배율 2에서 640 x 100이면 1280 x 200 이미지가 됩니다.
sgcWizardSaveToSVG(aModel, aFileName 또는 aStream, aWidth, aHeight, aScale = 1)	같은 스테퍼의 SVG 파일 또는 스트림을 기록합니다.
sgcWizardRender(aModel, aCanvas, aWidth, aHeight, aScale = 1)	디바이스 픽셀 단위로 모든 sgcUICore 캔버스에 스테퍼를 그립니다.
sgcWizardSaveButtonBarToPNG, sgcWizardSaveButtonBarToSVG (aButtons, aStream, aWidth, aHeight, aScale = 1, aTheme = utLight, aAccent = 0)	버튼 바를 기록합니다. sgcWizardButton(aCaption, aEnabled, aIsDefault, aHot, aPressed)이 버튼을 만들며, 캡션이 비어 있는 버튼은 숨겨집니다.
sgcWizardRenderButtonBar, sgcWizardButtonBarLayout, sgcWizardButtonBarHitTest	모든 sgcUICore 캔버스에서 버튼 바를 그리고, 배치하고, 히트 테스트합니다.

스테퍼 컨트롤의 모델은 **Model** 속성으로 사용할 수 있으므로, **sgcWizardSaveToPNG(Stepper1.Model, ...)**는 컨트롤이 표시하는 내용을 임의의 크기로 저장합니다. 마법사의 밴드는 **Wizard1.Stepper.Model**입니다.



예제

관련 항목

- TsgcStepperModel
- 플랫폼
- 단계 상태

.NET 개요

sgcWizard .NET은 Delphi sgcWizard 라이브러리의 관리형 C# 포트이며, 이를 감싸는 래퍼가 아닙니다. C# 클래스는 **esegece.sgcWizard** 네임스페이스에 있고 Delphi 이름을 유지하므로 **TsgcStepperModel**, **TsgcWizardSteps**, **TsgcWizardStepState**는 두 언어에서 똑같이 읽히며, 각 호스트의 마법사, 페이지, 스테퍼는 Delphi 명명 규칙을 따릅니다. 렌더링에는 **sgcUICore**의 .NET 포트를 사용하며, eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 라이트 및 다크 테마를 공유하므로 마법사는 WinForms, WPF, Avalonia, 헤드리스 PNG 및 SVG 출력에서 모두 똑같이 보입니다.

패키지

포트는 다섯 개의 NuGet 패키지로 나뉘어 있어 프로젝트는 사용하는 호스트만 가져옵니다.

PackageId	내용	대상 프레임워크
esegece.sgcWizard	스테퍼 모델(**TsgcStepperModel**, **TsgcWizardSteps**, **TsgcWizardStep**), 렌더러 **sgcWizard_Render**, 헤드리스 SVG 출력. 네이티브 종속성은 없습니다. esegece.sgcUICore에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcWizard.Rendering	헤드리스 PNG 출력 **sgcWizard_Render_Raster**. esegece.sgcUICore.Rendering의 SkiaSharp 캔버스에 그리며 Windows, Linux, macOS에서 동작합니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcWizard.WinForms	**TsgcWinFormsWizard**, **TsgcWinFormsWizardPage**, **TsgcWinFormsStepper**, **TsgcWinFormsWizardButtonBar**.	net472, net8.0-windows, net9.0-windows
esegece.sgcWizard.Wpf	**TsgcWpfWizard**, **TsgcWpfWizardPage**, **TsgcWpfStepper**, **TsgcWpfWizardButtonBar**.	net472, net8.0-windows, net9.0-windows
esegece.sgcWizard.Avalonia	**TsgcAvaloniaWizard**, **TsgcAvaloniaWizardPage**, **TsgcAvaloniaStepper**, **TsgcAvaloniaWizardButtonBar**. Avalonia 11에서 Windows, Linux, macOS를 지원합니다.	net8.0, net9.0

세 호스트 패키지는 esegece.sgcWizard.Rendering에 종속되고 이는 esegece.sgcWizard에 종속되므로, `PackageReference` 한 줄로 호스트에 필요한 모든 것이 들어옵니다. NuGet에서 설치를 참조하십시오.

에디션

에디션	패키지 ID	포함 내용
Registered	esegece.sgcWizard, esegece.sgcWizard.Rendering, esegece.sgcWizard.WinForms, esegece.sgcWizard.Wpf, esegece.sgcWizard.Avalonia	전부: 세 호스트의 마법사, 페이지, 스테퍼, 버튼 바, 유효성 검사, 건너뛴 페이지, 단계 상태, 테마, 헤드리스 PNG 및 SVG 출력.
Community	같은 ID에 .Community 접미사가 붙습니다. 예: esegece.sgcWizard.WinForms.Community. sgcUICore의 Community 패키지에 종속됩니다.	Registered와 내용이 같습니다. 라이선스만 다릅니다.

Community 에디션은 연간 총수익이 1,000,000 EUR 미만이고 개발자가 다섯 명 이하인 개인 및 조직이 상업적 및 비상업적 용도로 무료로 사용할 수 있습니다. 두 기준 중 하나라도 충족하거나 초과하는 조직은 Registered 라이선스가 필요합니다. .NET 포트에는 Trial 에디션이 없습니다.

호스트

Host	Delphi 대응 항목	플랫폼
WinForms	VCL (TsgcVCLWizard)	Windows, .NET Framework 4.7.2 및 .NET 8과 9.
WPF	VCL 및 FMX	Windows, .NET Framework 4.7.2 및 .NET 8과 9. 페이지는 XAML로 선언합니다.
Avalonia	FMX (TsgcFMXWizard)	Windows, Linux, macOS, .NET 8과 9. 페이지는 XAML로 선언합니다.
헤드리스	sgcWizard_Render	대상 프레임워크의 모든 플랫폼: SVG는 esegece.sgcWizard, PNG는 esegece.sgcWizard.Rendering.

모든 호스트는 같은 API를 가집니다: 페이지, ActivePageIndex, Next, Back, GoToPage, Finish, Cancel, 이벤트 OnBeforePageChange, OnPageChanged, OnFinish, OnCancel, 버튼 캡션, StepperPosition, ShowCancel, Theme, AccentColor. 클래스 이름은 .NET 클래스를, Delphi에서 달라지는 점은 Delphi와 .NET의 차이점을 참조하십시오.

관련 항목

- NuGet에서 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 헤드리스 렌더링
- .NET 데모
- 플랫폼

NuGet에서 설치

패키지는 nuget.org에 있습니다. 호스트의 패키지를 추가하면 코어 및 렌더링 패키지는 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcWizard.WinForms
dotnet add package esegece.sgcWizard.Wpf
dotnet add package esegece.sgcWizard.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcWizard.WinForms.Community
dotnet add package esegece.sgcWizard.Wpf.Community
dotnet add package esegece.sgcWizard.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcWizard.Wpf" Version="2026.10.0" />
</ItemGroup>
```

한 프로젝트에서는 Registered 또는 Community 패키지 중 하나만 참조하고 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있습니다.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`가 지정된 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`가 지정된 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서 `xmlns:sgc="clr-namespace:esegece.sgcWizard;assembly=esegece.sgcWizard.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: Avalonia 11.3의 **Avalonia.Desktop** 및 **Avalonia.Themes.Fluent** 패키지를 사용하는 `net8.0` 또는 `net9.0` 프로젝트로, 일반적인 Avalonia 데스크톱 애플리케이션과 같습니다. XAML에서 `xmlns:sgc="using:esegece.sgcWizard"`로 네임스페이스를 매핑합니다.
- 헤드리스: SVG에는 esegece.sgcWizard만으로 충분합니다. PNG에는 esegece.sgcWizard.Rendering을 추가하며, 이는 SkiaSharp와 Windows 및 Linux용 네이티브 자산을 가져옵니다.

C#에서는 모든 호스트에 `using esegece.sgcWizard;`가 필요하며, **TsgcUITheme**과 테마의 다른 형식에는 `using esegece.sgcUICore;`도 필요합니다.

관련 항목

- .NET 개요
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 데모

빠른 시작 WinForms

TsgcWinFormsWizard는 WinForms 마법사이며 Delphi TsgcVCLWizard에 해당합니다. 페이지 컨테이너로서 각 **TsgcWinFormsWizardPage**가 하나의 단계이고, 스테퍼 밴드가 단계를 표시하며, 버튼 바가 페이지 사이를 이동합니다. 패키지는 **esegece.sgcWizard.WinForms**입니다.

단계

- WinForms 프로젝트를 만들고 esegece.sgcWizard.WinForms 패키지를 추가합니다.
- **TsgcWinFormsWizard**를 만들어 폼에 도킹하고 **AddPage**로 단계마다 페이지 하나를 추가합니다. 각 페이지의 **Description**을 설정하십시오. 이는 그 단계의 설명이며, 캡션은 AddPage의 인수입니다.
- 각 페이지의 컨트롤을 그 페이지의 **Controls**에 추가합니다.
- 페이지를 검증하려면 **OnBeforePageChange**를 할당하고 페이지가 완료되지 않았을 때 aAllow를 false로 설정하며, 데이터를 사용하려면 **OnFinish**를 할당합니다.
- 실행: Next와 Enter는 앞으로 이동하고, Back은 뒤로 이동하며, Finish는 마지막 페이지에 나타나고, Cancel 또는 Escape는 OnCancel을 호출합니다.

예제

페이지 세 개짜리 마법사가 있는 WinForms 애플리케이션의 전체 Program.cs입니다. esegece.sgcWizard.WinForms를 참조하는 `net8.0-windows` 프로젝트에서 그대로 컴파일됩니다.

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcUICore;
using esegece.sgcWizard;

namespace WizardQuickStart
{
    internal class TFRMMain : Form
    {
        private readonly TsgcWinFormsWizard FWizard;
        private readonly TextBox FName;

        public TFRMMain()
        {
            Text = "sgcWizard WinForms";
            ClientSize = new Size(720, 460);

            // the wizard fills the form: stepper band on top, button bar at the bottom
            FWizard = new TsgcWinFormsWizard();
            FWizard.Dock = DockStyle.Fill;
            FWizard.Theme = TsgcUITheme.utLight;
            Controls.Add(FWizard);

            // every page is a step of the stepper band
            TsgcWinFormsWizardPage oPage = FWizard.AddPage("Welcome");
            oPage.Description = "Start here";
            Label oLabel = new Label();
            oLabel.SetBounds(24, 24, 320, 20);
            oLabel.Text = "This wizard creates your account.";
            oPage.Controls.Add(oLabel);

            oPage = FWizard.AddPage("Account");
            oPage.Description = "Your name";
            FName = new TextBox();
            FName.SetBounds(24, 24, 240, 24);
            oPage.Controls.Add(FName);
```

```
            oPage = FWizard.AddPage("Done");
            oPage.Description = "Summary";

            FWizard.ActivePageIndex = 0;
            FWizard.OnBeforePageChange = DoBeforePageChange;
            FWizard.OnFinish = DoFinish;
            FWizard.OnCancel = DoCancel;
        }

        private void DoBeforePageChange(object Sender, int aFromIndex, int aToIndex, ref bool aAllow)
        {
            // validate the Account page before moving forward
            if ((aFromIndex == 1) && (aToIndex > aFromIndex) && (FName.Text.Trim() == ""))
            {
                MessageBox.Show("Enter your name.");
                aAllow = false;
            }
        }

        private void DoFinish(object Sender)
        {
            MessageBox.Show("Account created for " + FName.Text);
            Close();
        }

        private void DoCancel(object Sender)
        {
            Close();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.Run(new TFRMMain());
        }
    }
}
```

페이지는 `new TsgcWinFormsWizardPage()`로 만들고 `Controls.Add`로 추가할 수도 있습니다. 페이지 순서는 추가된 순서입니다. 마법사와 스테퍼는 도구 상자에 있지만 Add Page 동사는 없습니다. Delphi와 .NET의 차이점을 참조하십시오.

관련 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- 유효성 검사

빠른 시작 WPF

TsgcWpfWizard는 WPF 마법사입니다. 페이지는 XAML에서 자식으로 선언된 **TsgcWpfWizardPage** 요소이며, 각 페이지는 콘텐츠 요소 하나를 가지고 보통 패널입니다. 패키지는 **esegece.sgcWizard.Wpf**입니다.

단계

- WPF 프로젝트(`dotnet new wpf`)를 만들고 esegece.sgcWizard.Wpf 패키지를 추가합니다.
- `esegece.sgcWizard` 네임스페이스를 매핑하고, 단계마다 **TsgcWpfWizardPage** 하나를 가진 **TsgcWpfWizard**를 선언합니다. 각 페이지의 **Caption**과 **Description**이 그 단계의 캡션과 설명입니다.
- 이벤트는 코드 비하인드에서 할당합니다. 이들은 WPF 이벤트가 아니라 델리게이트이므로 XAML 특성으로 설정할 수 없습니다.

예제

MainWindow.xaml은 마법사와 그 세 페이지를 선언합니다. 프로젝트의 App.xaml은 WPF 템플릿의 것이며 StartupUri가 MainWindow.xaml로 설정되어 있습니다.

```
<Window x:Class="WizardQuickStartWpf.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcWizard;assembly=esegece.sgcWizard.Wpf"
        Title="sgcWizard WPF" Width="720" Height="460">
    <!-- every page is a step of the stepper band and holds one content child -->
    <sgc:TsgcWpfWizard x:Name="Wizard" Theme="utLight" StepperPosition="wspTop">
        <sgc:TsgcWpfWizardPage Caption="Welcome" Description="Start here">
            <TextBlock Margin="24" Text="This wizard creates your account." />
        </sgc:TsgcWpfWizardPage>
        <sgc:TsgcWpfWizardPage Caption="Account" Description="Your name">
            <StackPanel Margin="24">
                <TextBlock Text="Name:" />
                <TextBox x:Name="edtName" Width="240" HorizontalAlignment="Left" />
            </StackPanel>
        </sgc:TsgcWpfWizardPage>
        <sgc:TsgcWpfWizardPage Caption="Done" Description="Summary">
            <TextBlock Margin="24" Text="Click Finish to create the account." />
        </sgc:TsgcWpfWizardPage>
    </sgc:TsgcWpfWizard>
</Window>
```

MainWindow.xaml.cs는 이벤트를 할당하고 Account 페이지를 검증합니다.

```
using System.Windows;
using esegece.sgcWizard;

namespace WizardQuickStartWpf
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            // the events are delegates: assign them in code
            Wizard.OnBeforePageChange = DoBeforePageChange;
            Wizard.OnFinish = DoFinish;
            Wizard.OnCancel = DoCancel;
        }

        private void DoBeforePageChange(object Sender, int aFromIndex, int aToIndex, ref bool aAllow)
        {
            // validate the Account page before moving forward
```

```
            if ((aFromIndex == 1) && (aToIndex > aFromIndex) && (edtName.Text.Trim() == ""))
            {
                MessageBox.Show("Enter your name.");
                aAllow = false;
            }
        }

        private void DoFinish(object Sender)
        {
            MessageBox.Show("Account created for " + edtName.Text);
            Close();
        }

        private void DoCancel(object Sender)
        {
            Close();
        }
    }
}
```

페이지는 코드에서 `Wizard.AddPage("Caption")`으로도 만들 수 있으며, 이는 새 **TsgcWpfWizardPage**를 반환하고, 마법사는 이들을 **Pages**로 노출합니다.

관련 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 WinForms
- 빠른 시작 Avalonia
- 유효성 검사

빠른 시작 Avalonia

TsgcAvaloniaWizard는 Avalonia 마법사이며 Delphi TsgcFMXWizard에 해당하고, Windows, Linux, macOS에서 동작합니다. 페이지는 XAML에서 자식으로 선언된 **TsgcAvaloniaWizardPage** 요소이며, 각 페이지는 자식 하나를 가집니다. 패키지는 **esegece.sgcWizard.Avalonia**입니다.

단계

- .NET 8 프로젝트를 만들고 esegece.sgcWizard.Avalonia, Avalonia.Desktop, Avalonia.Themes.Fluent 패키지를 추가합니다.
- `xmlns:sgc="using:esegece.sgcWizard"`로 네임스페이스를 매핑하고, 단계마다 **TsgcAvaloniaWizardPage** 하나와 그 **Caption** 및 **Description**을 가진 **TsgcAvaloniaWizard**를 선언합니다.
- 이벤트는 코드 비하인드에서 할당합니다. 이들은 Avalonia 이벤트가 아니라 델리게이트입니다.

예제

MainWindow.axaml은 마법사와 그 세 페이지를 선언합니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="using:esegece.sgcWizard"
        x:Class="WizardQuickStartAvalonia.MainWindow"
        Title="sgcWizard Avalonia" Width="720" Height="460">
    <!-- every page is a step of the stepper band and holds one child -->
    <sgc:TsgcAvaloniaWizard x:Name="Wizard" StepperPosition="wspTop">
        <sgc:TsgcAvaloniaWizardPage Caption="Welcome" Description="Start here">
            <TextBlock Margin="24" Text="This wizard creates your account." />
        </sgc:TsgcAvaloniaWizardPage>
        <sgc:TsgcAvaloniaWizardPage Caption="Account" Description="Your name">
            <StackPanel Margin="24" Spacing="6">
                <TextBlock Text="Name:" />
                <TextBox x:Name="edtName" Width="240" HorizontalAlignment="Left" />
                <TextBlock x:Name="lblError" Foreground="Red" />
            </StackPanel>
        </sgc:TsgcAvaloniaWizardPage>
        <sgc:TsgcAvaloniaWizardPage Caption="Done" Description="Summary">
            <TextBlock Margin="24" Text="Click Finish to create the account." />
        </sgc:TsgcAvaloniaWizardPage>
    </sgc:TsgcAvaloniaWizard>
</Window>
```

MainWindow.axaml.cs는 테마를 설정하고, 이벤트를 할당하며, 이름이 비어 있는 동안 Account 페이지에 메시지를 표시합니다.

```
using Avalonia.Controls;
using esegece.sgcUICore;
using esegece.sgcWizard;

namespace WizardQuickStartAvalonia
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            // Theme is a TsgcUITheme; the events are delegates, assigned in code
            Wizard.Theme = TsgcUITheme.utLight;
            Wizard.OnBeforePageChange = DoBeforePageChange;
            Wizard.OnFinish = DoFinish;
            Wizard.OnCancel = DoCancel;
        }
```

```
        private void DoBeforePageChange(object Sender, int aFromIndex, int aToIndex, ref bool aAllow)
        {
            // validate the Account page before moving forward
            lblError.Text = "";
            if ((aFromIndex == 1) && (aToIndex > aFromIndex) && string.IsNullOrWhiteSpace(edtName.Text))
            {
                lblError.Text = "Enter your name.";
                aAllow = false;
            }
        }

        private void DoFinish(object Sender)
        {
            Close();
        }

        private void DoCancel(object Sender)
        {
            Close();
        }
    }
}
```

Program.cs는 Fluent 테마로 애플리케이션을 시작합니다.

```
using System;
using Avalonia;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;

namespace WizardQuickStartAvalonia
{
    internal class App : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new MainWindow();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main(string[] args)
        {
            BuildAvaloniaApp().StartWithClassicDesktopLifetime(args);
        }

        public static AppBuilder BuildAvaloniaApp()
        {
            return AppBuilder.Configure<App>().UsePlatformDetect().LogToTrace();
        }
    }
}
```

페이지는 코드에서 `Wizard.AddPage("Caption")`으로도 만들 수 있으며, 이는 새 **TsgcAvaloniaWizardPage**를 반환합니다. 그 **Child**를 페이지의 콘텐츠로 설정하십시오.

관련 항목

- .NET 개요
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 유효성 검사

.NET 클래스

모든 호스트에는 동일한 멤버를 가진 마법사, 페이지, 스테퍼, 버튼 바가 있습니다. 프레임워크 중립 형식은 세 호스트가 공유하며 Delphi 이름을 그대로 유지합니다.

호스트 클래스

역할	WinForms	WPF	Avalonia
마법사	TsgcWinFormsWizard	TsgcWpfWizard	TsgcAvaloniaWizard
페이지	TsgcWinFormsWizardPage (Control 컨테이너)	TsgcWpfWizardPage (Content 요소 하나)	TsgcAvaloniaWizardPage (Child 가진 Border)
페이지 컬렉션	ReadOnlyCollection<TsgcWinFormsWizardPage>	TsgcWpfWizardPages	AvaloniaList<TsgcAvaloniaWizar
스테퍼	TsgcWinFormsStepper	TsgcWpfStepper	TsgcAvaloniaStepper
버튼 바	TsgcWinFormsWizardButtonBar	TsgcWpfWizardButtonBar	TsgcAvaloniaWizardButtonBar

프레임워크 중립 형식

형식	설명
TsgcStepperModel	단계, 활성 단계, 방향, 자동 상태, 레이아웃, 히트 테스트. 모든 호스트의 스테퍼가 하나씩 소유합니다. TsgcStepperModel을 참조하십시오.
TsgcWizardSteps, TsgcWizardStep	단계 컬렉션과 단일 단계: Caption, Description, Glyph, State, Enabled, Visible, Tag.
TsgcWizardStepState	wssPending, wssActive, wssCompleted, wssError, wssSkipped.
TsgcWizardOrientation	woHorizontal, woVertical.
TsgcWizardStepperPosition	wspTop, wspLeft, wspNone.
TsgcWizardPageChangeEvent	`void (object Sender, int aFromIndex, int aToIndex, ref bool aAllow)`
TsgcWizardStepClickEvent	`void (object Sender, int aIndex, ref bool aAllow)`
TsgcUINotifyEvent	`void (object Sender)`, esegece.sgcUICore에서 제공.

sgcWizard_Render, sgcWizard_Render_Raster	렌더러와 헤드리스 SVG 및 PNG 출력. .NET 헤드리스 렌더링을 참조하십시오.

마법사

멤버	형식	설명
Pages	컬렉션	순서대로 나열된 페이지. PageCount와 IndexOfPage(aPage)는 Delphi와 같습니다.
AddPage(string aCaption)	페이지	페이지를 만들어 끝에 추가하고 반환합니다.
ActivePageIndex, ActivePage	int, 페이지	표시되는 페이지. 설정하면 이벤트 없이 페이지가 바로 전환됩니다.
Next(), Back(), GoToPage(int aIndex)	bool	OnBeforePageChange를 거쳐 다른 페이지로 이동하며, Skip이 true인 페이지는 건너뜁니다. 마지막 페이지에서 Next는 Finish를 호출합니다.
Finish(), Cancel()	bool, void	Finish는 aToIndex = PageCount로 OnBeforePageChange를 호출한 다음 OnFinish를 호출합니다. Cancel은 OnCancel을 호출합니다.
NextPageIndex(int), PrevPageIndex(int), IsLastPage()	int, bool	건너뛰지 않는 다음 페이지와 이전 페이지, 그리고 활성 페이지가 마지막인지 여부.
StepperPosition	TsgcWizardStepperPosition	스테퍼 밴드의 위치: 위, 왼쪽 또는 숨김.
BackCaption, NextCaption, FinishCaption, CancelCaption	string	버튼 캡션. 현지화용입니다.
ShowCancel	bool	Cancel 버튼을 표시합니다.
Theme, AccentColor	TsgcUITheme, uint	마법사, 스테퍼 밴드, 버튼 바, 그리고 페이지 컨트롤의 테마. AccentColor가 0이면 테마의 강조색을 사용합니다.
Stepper, ButtonBar	스테퍼, 버튼 바	내부 스테퍼 밴드와 버튼 바.
OnBeforePageChange	TsgcWizardPageChangeEvent	모든 페이지 변경 전과 Finish 전에 발생합니다. aAllow를 false로 설정하면 거부됩니다.
OnPageChanged, OnFinish, OnCancel	TsgcUINotifyEvent	페이지 변경 후, Finish 시, Cancel 시.

페이지

멤버	형식	설명
Caption	string	단계의 캡션. WinForms에서는 컨트롤의 Text입니다.
Description	string	단계의 설명과 그 툴팁.

StepGlyph	string	단계 번호 대신 단계의 원 안에 표시되는 텍스트.
Skip	bool	Next, Back, GoToPage가 이 페이지를 건너뛰며 단계는 건너뜀으로 그려집니다.
Wizard, PageIndex	마법사, int	페이지의 마법사와 그 인덱스.
Content (WPF), Child (Avalonia), Controls (WinForms)	요소	페이지가 표시하는 내용.

스테퍼

스테퍼는 마법사 없이 단독으로 동작합니다. 멤버는 Delphi 스테퍼와 같습니다: Model, Steps, Orientation, ActiveIndex, AutoStates, ShowDescriptions, CircleSize, ConnectorThickness, Theme, AccentColor, Clickable, StepAt(X, Y), StepRect(aIndex), OnStepClick (TsgcWizardStepClickEvent, aAllow를 false로 설정하면 클릭이 거부됨), 그리고 사용자 조작 시에만 발생하는 OnActiveChange. Wizard는 스테퍼를 소유한 마법사이며, 단독 스테퍼에서는 null입니다. TsgcVCLStepper를 참조하십시오.

버튼 바

마법사의 버튼 바는 Back, Next (마지막 페이지에서는 Finish), Cancel을 그립니다. GetButtons(), ButtonCount(), Buttons(aIndex), ButtonRect(aIndex), ButtonAt(X, Y)가 버튼을 설명하며, Wizard는 해당 마법사입니다. 버튼 바는 마법사가 만들며 ButtonBar 속성으로 접근합니다.
위치와 크기는 WinForms에서는 픽셀(Rectangle)이고, WPF와 Avalonia에서는 장치 독립 단위(Rect)입니다.

관련 항목

- .NET 개요
- Delphi와 .NET의 차이점
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia

Delphi와 .NET의 차이점

.NET API는 Delphi API를 이름 그대로 따릅니다. 다른 부분은 다음과 같습니다.

이벤트

OnBeforePageChange, OnPageChanged, OnFinish, OnCancel, OnStepClick, OnActiveChange는 Delphi 이름을 가진 델리게이트 속성이며, C# `event` 멤버가 아닙니다. 핸들러를 할당하면 Delphi 메서드 포인터처럼 이전 핸들러가 대체되며, XAML에서 설정할 수 없는 WPF와 Avalonia에서도 코드에서 할당합니다. Delphi의 `var aAllow: Boolean` 매개변수는 `ref bool aAllow`이며, OnBeforePageChange 핸들러의 시그니처는 `void (object Sender, int aFromIndex, int aToIndex, ref bool aAllow)`입니다. 세 호스트의 핸들러는 빠른 시작에 나와 있습니다.

GoToPage와 탐색

GoToPage, Next, Back, Finish는 bool을 반환하며 Delphi와 동일하게 동작합니다. GoToPage는 인덱스가 범위를 벗어났거나 페이지를 건너뛰는 경우 false를 반환하고, 페이지가 이미 활성 상태이면 true를 반환하며, 그 밖의 경우에는 OnBeforePageChange를 거칩니다. Finish는 aToIndex = PageCount로 OnBeforePageChange를 호출하므로 앞으로 이동하는 모든 경우에 aToIndex > aFromIndex가 됩니다. 스테퍼 밴드에서 완료된 단계를 클릭하면 GoToPage가 호출됩니다. ActivePageIndex 또는 ActivePage를 설정하면 Delphi와 같이 이벤트 없이 페이지가 전환됩니다.

페이지 컬렉션

Delphi에는 인덱스 속성 `Pages[Index]`와 `PageCount`가 있습니다. .NET에서 **Pages**는 컬렉션이므로 `Pages[i]`, `Pages.Count`, `foreach`를 평소처럼 사용할 수 있습니다.

- WinForms: 읽기 전용 컬렉션입니다. 페이지는 AddPage로 추가하거나 마법사의 Controls에 TsgcWinFormsWizardPage를 추가하여 더하며, 페이지 순서는 컨트롤의 z 순서가 아니라 추가된 순서입니다.
- WPF: TsgcWpfWizardPages는 마법사의 콘텐츠 속성이므로 XAML에서 페이지는 마법사의 자식입니다. 각 페이지에는 Content 요소가 하나 있습니다.
- Avalonia: AvaloniaList는 마법사의 콘텐츠 속성입니다. 각 페이지는 Child 하나를 가진 Border입니다.

힌트와 툴팁

Delphi에서 단계의 힌트는 Description이며 ShowHint가 True일 때 표시되고, FMX는 10.1 Berlin부터 힌트를 표시합니다. .NET에서는 스테퍼가 항상 마우스 아래 단계의 Description을 툴팁으로 표시하며 ShowHint 속성은 없습니다. WinForms는 내부 ToolTip 컴포넌트를 사용하고, WPF는 스테퍼의 ToolTip 속성을 설정하며, Avalonia는 ToolTip.Tip 연결 속성을 설정합니다. 스테퍼의 툴팁을 직접 설정하지 마십시오. 스테퍼가 이를 대체합니다. 대신 단계 또는 페이지의 Description을 설정하십시오.

디자이너 동사 없음

Delphi 컴포넌트 편집기에는 Add Page, Next Page, Previous Page 동사가 있으며, 단계를 클릭하면 디자이너에 다른 페이지가 표시됩니다. 디자인 타임을 참조하십시오. .NET 포트에는 디자이너 동사가 없습니다. WinForms에서는 마법사와 스테퍼가 도구 상자에 있으며 페이지는 코드에서 AddPage로 만듭니다. WPF와 Avalonia에서는 페이지를 XAML로 선언합니다.

키보드

포커스가 마법사 안에 있는 동안 Enter는 Next 또는 Finish이고 Escape는 Cancel이며, Delphi와 같습니다. WinForms에서는 키가 ProcessDialogKey를 통해 전달되므로 여러 줄 TextBox나 포커스가 있는 버튼이 그 키를 유지합니다. WPF와 Avalonia에서는 버블링되는 KeyDown 이벤트를 통해 전달되므로 해당 키를 처리하는 컨트롤이 우선합니다.

에디션

.NET 포트는 내용이 같은 Registered 및 Community 패키지로 제공됩니다. .NET 개요를 참조하십시오. Trial 에디션은 없으며 코드에 컴파일 타임 스위치도 없습니다.

데이터 바인딩

WPF와 Avalonia에서 사용자용 속성은 종속성 속성과 Avalonia 속성이므로 데이터 바인딩, 스타일 지정, 애니메이션이 가능합니다. .NET 데이터 바인딩을 참조하십시오.

관련 항목

- .NET 개요
- .NET 클래스
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia

.NET 데이터 바인딩

WPF 및 Avalonia 호스트에서 위저드, 해당 페이지, 스테퍼의 사용자용 속성은 종속성 속성(WPF)과 Avalonia 속성이므로 뷰 모델에 데이터 바인딩하거나 Style로 설정하거나 애니메이션을 적용할 수 있습니다. 이 항목은 WPF와 Avalonia에만 적용됩니다. WinForms 컨트롤은 변경되지 않았으며, 해당 속성은 이전과 같은 일반 CLR 속성입니다.

바인딩 가능한 속성

표의 모든 속성에는 접미사 Property가 붙은 정적 필드가 있습니다. 예를 들어 `TsgcWpfWizard.ActivePageIndexProperty`입니다. WPF에서 이 필드는 DependencyProperty입니다. Avalonia에서는 StyledProperty이며, ActivePageIndex와 ActiveIndex만 DirectProperty입니다. **TwoWay**로 표시된 속성은 컨트롤 자체가 변경하므로(Next, Back, GoToPage, 단계 클릭) 기본적으로 TwoWay로 바인딩되며, 따라서 Mode가 없는 바인딩도 뷰 모델을 최신 상태로 유지합니다. 나머지 속성은 기본적으로 OneWay로 바인딩됩니다.

WPF 클래스	Avalonia 클래스	바인딩 가능한 속성
TsgcWpfWizard	TsgcAvaloniaWizard	**ActivePageIndex (TwoWay)**, StepperPosition, BackCaption, NextCaption, FinishCaption, CancelCaption, ShowCancel, Theme, AccentColor
TsgcWpfWizardPage	TsgcAvaloniaWizardPage	Caption, Description, StepGlyph, Skip
TsgcWpfStepper	TsgcAvaloniaStepper	Orientation, **ActiveIndex (TwoWay)**, AutoStates, ShowDescriptions, CircleSize, ConnectorThickness, Theme, AccentColor, Clickable

Avalonia에서 Theme은 sgcUICore 테마(TsgcUITheme)이며 StyledElement의 Avalonia 컨트롤 테마를 가립니다. 후자는 `StyledElement.ThemeProperty`를 통해 계속 접근할 수 있습니다.

동작

- **바인딩은 이동하고 코드는 이동하지 않습니다.** 데이터 바인딩을 통해 도착한 새 ActivePageIndex는 이동으로 처리되어 GoToPage처럼 OnBeforePageChange를 거친 다음 OnPageChanged를 거칩니다. 코드(CLR setter)에서 설정하면 ActivePageIndex는 Delphi와 같이 이벤트 없이 페이지를 전환합니다.
- **거부는 다시 기록됩니다.** OnBeforePageChange가 aAllow를 false로 설정하면 위저드는 현재 페이지를 유지하고 컨트롤과 바인딩된 뷰 모델의 이전 인덱스를 복원합니다. WPF에서는 디스패처가 대기 중인 작업을 실행한 뒤에 뷰 모델이 이전 인덱스를 받습니다.
- **범위를 벗어난 인덱스는 무시됩니다.** 페이지가 아닌 바인딩된 인덱스는 현재 페이지를 유지하고 이벤트를 발생시키지 않으며, TwoWay 소스는 WPF와 Avalonia 모두에서 사용 중인 인덱스를 다시 받습니다. 바인딩은 계속 유지되므로 다음 유효한 인덱스로 이동합니다.
- **수정된 값은 소스에 도달합니다.** CircleSize나 스테퍼가 범위로 제한한 ActiveIndex처럼 컨트롤이 수정한 값은 WPF와 Avalonia 모두에서 속성과 TwoWay 소스에 다시 기록됩니다. WPF에서는 디스패처가 대기 중인 작업을 실행한 뒤에 소스가 값을 받습니다. Avalonia에서 다시 쓰기는 바인딩이 실행 중일 때 소스가 보낸 값에 적용됩니다. 바인딩이 시작될 때 소스가 이미 가지고 있는 값은 컨트롤에서 수정되지만 소스에는 다시 기록되지 않습니다.
- **뷰 모델은 위저드를 따라갑니다.** Next, Back, GoToPage, 단계 클릭, 코드에서의 변경은 새 인덱스를 ActivePageIndex에 복사하고, TwoWay 바인딩이 이를 뷰 모델에 전달합니다.
- **스타일.** Style setter는 바인딩 가능한 모든 속성을 설정할 수 있습니다. 예를 들어 창에 있는 모든 위저드의 Theme과 스테퍼 옵션입니다.

CLR 속성으로 남는 속성

속성	이유

Pages (위저드)	페이지 컬렉션은 위저드의 콘텐츠 속성이며 XAML에 선언된 페이지로 채워집니다. 바인딩할 값이 아닙니다.
ActivePage (위저드)	페이지 요소에 대한 참조입니다. 대신 모든 페이지 변경을 따라가는 ActivePageIndex를 바인딩하십시오.
PageIndex (페이지)	읽기 전용이며 Pages에서 페이지의 위치입니다.
Steps (스테퍼)	단계 컬렉션입니다. 위저드는 페이지로부터 자체 스테퍼의 단계를 채웁니다.
OnBeforePageChange, OnPageChanged, OnFinish, OnCancel, OnStepClick, OnActiveChange	Delphi와 같은 이름의 델리게이트 속성으로, 코드에서 할당합니다. Delphi와 .NET의 차이점을 참조하십시오.

예제

아래 뷰 모델은 두 예제에서 공유됩니다. PageIndex는 Mode 없이 ActivePageIndex에 바인딩되므로 TwoWay로 바인딩되고, NextCaption은 Next 버튼의 캡션에 바인딩되며, SkipOptions는 Options 페이지의 Skip 속성과 확인란에 바인딩됩니다.

```
using System.Collections.Generic;
using System.ComponentModel;
using System.Runtime.CompilerServices;

namespace WizardBinding
{
    public class MainViewModel : INotifyPropertyChanged
    {
        private int FPageIndex;
        private string FNextCaption = "Next";
        private bool FSkipOptions;

        public event PropertyChangedEventHandler PropertyChanged;

        // bound to ActivePageIndex, TwoWay by default
        public int PageIndex
        {
            get { return FPageIndex; }
            set { DoSet(ref FPageIndex, value); }
        }

        public string NextCaption
        {
            get { return FNextCaption; }
            set { DoSet(ref FNextCaption, value); }
        }

        // bound to the Skip property of the Options page
        public bool SkipOptions
        {
            get { return FSkipOptions; }
            set { DoSet(ref FSkipOptions, value); }
        }

        private void DoSet<T>(ref T aField, T aValue, [CallerMemberName] string aName = null)
        {
            if (EqualityComparer<T>.Default.Equals(aField, aValue))
                return;
            aField = aValue;
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(aName));
        }
```

```
    }
}
```

WPF

MainWindow.xaml은 뷰 모델을 창의 DataContext로 만듭니다. 코드 비하인드와 App.xaml은 WPF 템플릿의 것입니다.

```
<Window x:Class="WizardBinding.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="clr-namespace:WizardBinding"
        xmlns:sgc="clr-namespace:esegece.sgcWizard;assembly=esegece.sgcWizard.Wpf"
        Title="sgcWizard binding" Width="720" Height="460">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <StackPanel DockPanel.Dock="Top" Orientation="Horizontal" Margin="8">
            <CheckBox Content="Skip the options page" IsChecked="{Binding SkipOptions}" />
            <TextBlock Margin="24,0,0,0" Text="{Binding PageIndex}" />
        </StackPanel>
        <!-- ActivePageIndex binds TwoWay by default, so the view model follows Next and Back -->
        <sgc:TsgcWpfWizard x:Name="Wizard" ActivePageIndex="{Binding PageIndex}"
                           NextCaption="{Binding NextCaption}">
            <sgc:TsgcWpfWizardPage Caption="Welcome" Description="Start here">
                <TextBlock Margin="24" Text="This wizard creates your account." />
            </sgc:TsgcWpfWizardPage>
            <sgc:TsgcWpfWizardPage Caption="Options" Description="Optional settings"
                                   Skip="{Binding SkipOptions}">
                <TextBlock Margin="24" Text="Choose the optional settings." />
            </sgc:TsgcWpfWizardPage>
            <sgc:TsgcWpfWizardPage Caption="Done" Description="Summary">
                <TextBlock Margin="24" Text="Click Finish to create the account." />
            </sgc:TsgcWpfWizardPage>
        </sgc:TsgcWpfWizard>
    </DockPanel>
</Window>
```

Avalonia

MainWindow.axaml은 `x:DataType`을 뷰 모델로 지정하므로 바인딩은 컴파일된 바인딩입니다. 코드 비하인드는 InitializeComponent를 호출하고 App 클래스는 빠른 시작 Avalonia의 것입니다.

```
<Window xmlns="https://github.com/avaloniaui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:WizardBinding"
        xmlns:sgc="using:esegece.sgcWizard"
        x:Class="WizardBinding.MainWindow"
        x:DataType="local:MainViewModel"
        Title="sgcWizard binding" Width="720" Height="460">
    <Window.DataContext>
        <local:MainViewModel />
    </Window.DataContext>
    <DockPanel>
        <StackPanel DockPanel.Dock="Top" Orientation="Horizontal" Margin="8" Spacing="24">
            <CheckBox Content="Skip the options page" IsChecked="{Binding SkipOptions}" />
            <TextBlock VerticalAlignment="Center" Text="{Binding PageIndex}" />
        </StackPanel>
        <!-- ActivePageIndex binds TwoWay by default, so the view model follows Next and Back -->
        <sgc:TsgcAvaloniaWizard x:Name="Wizard" ActivePageIndex="{Binding PageIndex}"
                                NextCaption="{Binding NextCaption}">
            <sgc:TsgcAvaloniaWizardPage Caption="Welcome" Description="Start here">
                <TextBlock Margin="24" Text="This wizard creates your account." />
            </sgc:TsgcAvaloniaWizardPage>
            <sgc:TsgcAvaloniaWizardPage Caption="Options" Description="Optional settings"
```

```
                                    Skip="{Binding SkipOptions}">
                <TextBlock Margin="24" Text="Choose the optional settings." />
            </sgc:TsgcAvaloniaWizardPage>
            <sgc:TsgcAvaloniaWizardPage Caption="Done" Description="Summary">
                <TextBlock Margin="24" Text="Click Finish to create the account." />
            </sgc:TsgcAvaloniaWizardPage>
        </sgc:TsgcAvaloniaWizard>
    </DockPanel>
</Window>
```

참고 항목

- Delphi와 .NET의 차이점
- .NET 클래스
- 빠른 시작 WPF
- 빠른 시작 Avalonia

.NET 헤드리스 렌더링

스테퍼는 서버, 테스트 또는 보고서에서 창 없이 그릴 수 있습니다. **sgcWizard_Render.sgcWizardSaveToSVG**는 esegece.sgcWizard 패키지만으로 SVG 파일 또는 스트림을 기록하고, **sgcWizard_Render_Raster.sgcWizardSaveToPNG**는 esegece.sgcWizard.Rendering으로 PNG를 기록하며, 이는 Windows, Linux, macOS의 SkiaSharp에 그립니다. SVG 출력은 Delphi 헤드리스 렌더링의 출력과 바이트 단위로 동일합니다.

메서드

메서드	패키지	설명
sgcWizard_Render.sgcWizardSaveToSVG(aModel, aFileName or aStream, aWidth, aHeight, aScale = 1)	esegece.sgcWizard	aModel의 스 SVG로 기록합 다.
sgcWizard_Render_Raster.sgcWizardSaveToPNG(aModel, aFileName or aStream, aWidth, aHeight, aScale = 1)	esegece.sgcWizard.Rendering	aModel의 스 투명한 배경의 PNG로 기록합 다.
sgcWizard_Render.sgcWizardSaveButtonBarToSVG(aButtons, aStream, aWidth, aHeight, aScale, aTheme, aAccent)	esegece.sgcWizard	버튼 바를 SV 기록합니다.
sgcWizard_Render_Raster.sgcWizardSaveButtonBarToPNG(aButtons, aStream, aWidth, aHeight, aScale, aTheme, aAccent)	esegece.sgcWizard.Rendering	버튼 바를 PN 기록합니다.
sgcWizard_Render.sgcWizardButton(aCaption, aEnabled = true, aIsDefault = false, aHot = false, aPressed = false)	esegece.sgcWizard	버튼 바용 TsgcWizardB 하나를 반환합
sgcWizard_Render.sgcWizardRender(aModel, aCanvas, aWidth, aHeight, aScale = 1)	esegece.sgcWizard	임의의 sgcUI 캔버스에 스테 그립니다.

aWidth와 aHeight는 논리 픽셀입니다. 이미지 크기는 aWidth x aScale 곱하기 aHeight x aScale 픽셀이므로, 배율 2는 고해상도 화면이나 인쇄에 선명한 이미지를 제공합니다.

예제

esegece.sgcWizard.Rendering을 참조하는 콘솔 프로그램(`net8.0`)으로, 건너뛴 단계가 있는 어두운 스테퍼를 PNG와 SVG로, 같은 단계를 세로로 그린 것을, 그리고 버튼 바를 기록합니다.

```
using System.IO;
using esegece.sgcUICore;
using esegece.sgcWizard;

namespace WizardHeadless
{
    internal static class Program
    {
        private static void Main()
        {
            TsgcStepperModel oModel = new TsgcStepperModel();
            oModel.Steps.AddStep("Welcome", "Start here");
            oModel.Steps.AddStep("Account", "Name and email");
            oModel.Steps.AddStep("Options", "Skipped");
```

```
            oModel.Steps.AddStep("Payment", "Card details");
            oModel.Steps.AddStep("Summary", "Review and finish");
            oModel.Steps[2].State = TsgcWizardStepState.wssSkipped;
            oModel.ActiveIndex = 3;
            oModel.Theme = TsgcUITheme.utDark;

            // 640 x 100 logical pixels, written at scale 2 as 1280 x 200 pixels (esegece.sgcWizard.Rendering)
            sgcWizard_Render_Raster.sgcWizardSaveToPNG(oModel, "wizard.png", 640, 100, 2);
            // SVG needs only esegece.sgcWizard, no native library
            sgcWizard_Render.sgcWizardSaveToSVG(oModel, "wizard.svg", 640, 100);

            // the same steps drawn vertically
            oModel.Orientation = TsgcWizardOrientation.woVertical;
            sgcWizard_Render_Raster.sgcWizardSaveToPNG(oModel, "wizard_vertical.png", 220, 340);

            // a button bar: Back, Next as the default button, Cancel
            TsgcWizardButton[] vButtons = new TsgcWizardButton[]
            {
                sgcWizard_Render.sgcWizardButton("Back"),
                sgcWizard_Render.sgcWizardButton("Next", true, true),
                sgcWizard_Render.sgcWizardButton("Cancel")
            };
            using (FileStream oStream = new FileStream("buttons.png", FileMode.Create, FileAccess.Write))
                sgcWizard_Render_Raster.sgcWizardSaveButtonBarToPNG(vButtons, oStream, 640, 52, 1,
                    TsgcUITheme.utDark);
        }
    }
}
```

관련 항목

- .NET 개요
- .NET 클래스
- 헤드리스 렌더링
- TsgcStepperModel

.NET 데모

.NET 데모는 제품의 `demos` 폴더와 데모 zip에 솔루션 `sgcWizard_Demos.sln`과 함께 들어 있습니다. 각 데모에는 README.md가 있습니다.

데모	Host	설명
Onboarding	WinForms	TsgcWinFormsWizard 위의 네 페이지 계정 마법사(Welcome, Account, Options, Summary). OnBeforePageChange가 Account 페이지의 이름과 이메일을 검증하고, 확인란이 Options 페이지의 Skip을 설정하며, OnPageChanged가 요약을 채우고, OnFinish가 이를 표시하며, OnCancel이 닫기 전에 확인합니다. 도구 모음에서 테마를 전환하고 스테퍼 밴드를 위쪽 또는 왼쪽에 두거나 숨길 수 있습니다.
OnboardingWpf	WPF	MainWindow.xaml에 선언된 TsgcWpfWizard 위의 동일한 마법사.
OnboardingAvalonia	Avalonia	TsgcAvaloniaWizard 위의 동일한 마법사. Windows, Linux, macOS에서 동작합니다.
Stepper	WinForms	결제 과정의 다섯 단계를 가진 독립형 TsgcWinFormsStepper 컨트롤 두 개(가로 하나, 세로 하나)가 서로 동기화됩니다. 버튼으로 앞으로 이동, 뒤로 이동, 활성 단계를 오류로 표시, 건너뛰기, 단계 초기화를 할 수 있고, 완료된 단계를 클릭하면 OnStepClick을 통해 그 단계가 활성화됩니다.
StepperWpf	WPF	MainWindow.xaml에 배치된 TsgcWpfStepper 컨트롤로 만든 동일한 스테퍼.
StepperAvalonia	Avalonia	TsgcAvaloniaStepper 컨트롤로 만든 동일한 스테퍼.

빌드 및 실행

데모에는 .NET 8 SDK 이상이 필요합니다. 데모 폴더에서 실행합니다.

```
dotnet run
```

제품 트리 안에서는 데모가 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip에서는 NuGet 패키지를 복원하며, 기본값은 Community 에디션입니다. Registered 패키지를 사용하려면 **SgcWizardEdition**을 설정하십시오.

```
dotnet build -p:SgcWizardEdition=Registered
```

자체 점검

각 데모에는 `--test` 스위치가 있어 화면 밖에서 창을 열고, 마법사나 스테퍼를 구동하며, 점검마다 PASS 또는 FAIL 한 줄을 파일에 기록하고, 실패한 점검의 수를 종료 코드로 반환합니다. 따라서 0은 모든 점검을 통과했음을 의미합니다.

```
dotnet run -- --test result.txt
```

관련 항목

- .NET 개요
- NuGet에서 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- 데모

문제 해결

EsgcWizardError: A TsgcVCLWizardPage can only be placed on a TsgcVCLWizard

페이지는 마법사의 직접 자식이어야 합니다. TsgcVCLWizardPage의 Parent를 패널, 폼 또는 다른 컨트롤로 설정하면 EsgcWizardError가 발생하며, TsgcFMXWizard 밖의 TsgcFMXWizardPage도 마찬가지입니다. 페이지는 **AddPage**로 만들거나 Parent를 마법사로 설정하십시오. 페이지에서 컨트롤을 묶으려면 페이지에 TPanel이나 프레임을 놓으십시오. TsgcVCLWizardPage를 참조하십시오.

테스트 프로그램이 관리자 권한을 요구합니다

Windows는 이름에 setup, install, update 같은 단어가 들어 있는 실행 파일을 설치 프로그램으로 간주하여, 실행하기 전에 권한 상승을 요청합니다. 마법사를 테스트할 때는 프로그램 이름을 SetupWizard.exe나 InstallWizard.exe로 하지 말고 Onboarding.exe 같은 이름을 고르거나, requestedExecutionLevel을 asInvoker로 지정한 매니페스트를 추가하십시오.

Next를 눌러도 아무 일도 일어나지 않습니다

OnBeforePageChange가 aAllow를 False로 설정했거나(유효성 검사가 변경을 거부하는 방식입니다), 다음 페이지가 모두 건너뛴 페이지여서 Next가 Finish를 호출합니다. 핸들러와 페이지의 Skip 속성을 확인하십시오. 유효성 검사를 참조하십시오.

Finish를 눌러도 폼이 닫히지 않습니다

마법사는 아무것도 닫지 않습니다. OnFinish와 OnCancel에서 폼을 닫거나 ModalResult를 설정하십시오. Finish와 Cancel을 참조하십시오.

Enter를 눌러도 다음 페이지로 이동하지 않습니다

Enter를 직접 처리하려는 컨트롤이 키를 가져갑니다. TMemo, 포커스가 있는 버튼, 또는 폼의 Default가 True인 TButton이 그렇습니다. 키보드를 참조하십시오.

단계를 클릭해도 페이지가 바뀌지 않습니다

완료된 단계와 활성 단계보다 앞의 오류 단계만 클릭할 수 있으며, Clickable이 True인 동안에만 가능합니다. 클릭도 OnBeforePageChange를 거칩니다.

어두운 페이지에서 FMX 편집 컨트롤이 밝은 색을 유지합니다

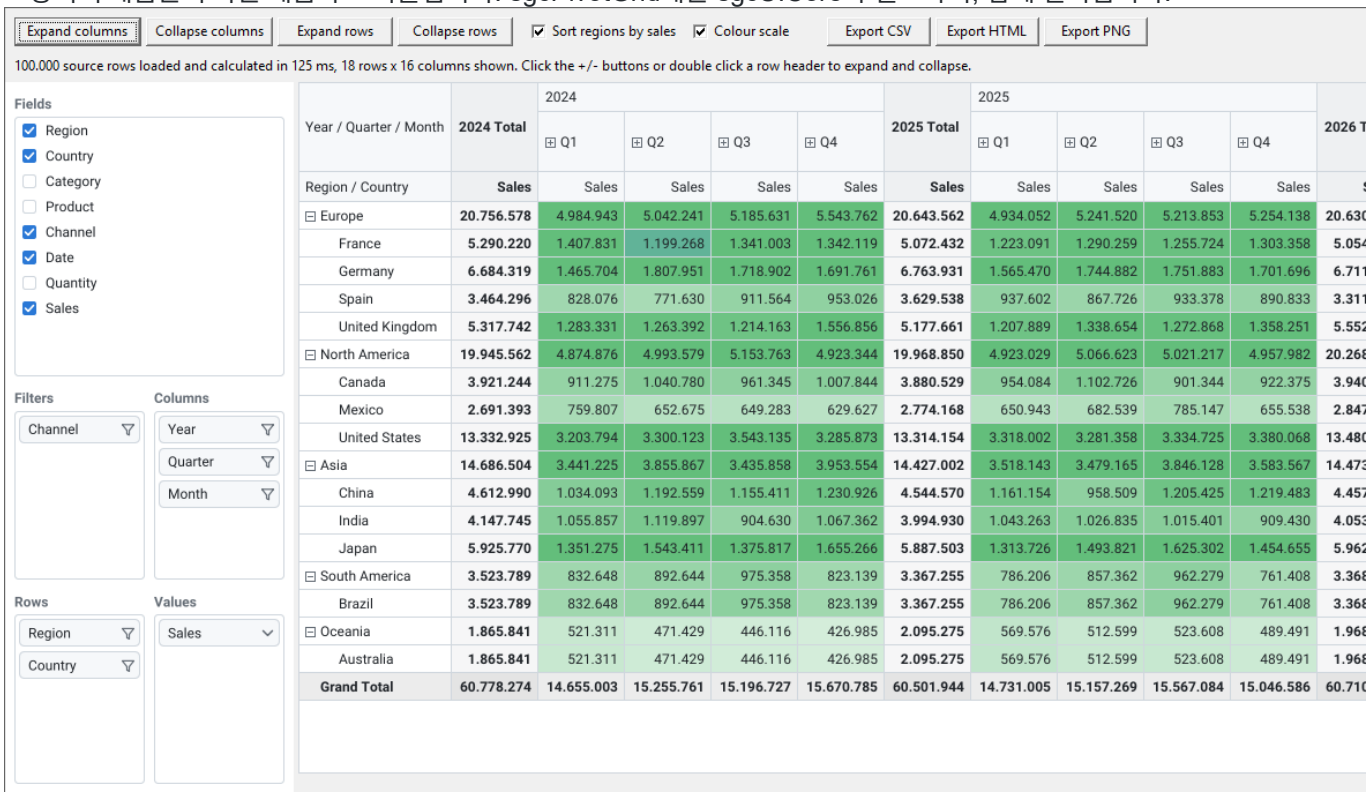
스타일이 적용된 FireMonkey 편집 컨트롤은 스타일의 색을 유지합니다. 폼에 어두운 스타일을 사용하거나 컨트롤의 색을 직접 설정하십시오. 테마를 참조하십시오.

FMX 컨트롤이 없습니다

FireMonkey 컨트롤에는 Delphi XE6 이상이 필요하며, FMX 컨트롤의 힌트에는 10.1 Berlin 이상이 필요합니다. 플랫폼을 참조하십시오.

소개

sgcPivotGrid는 VCL과 FireMonkey용 피벗 테이블을 Delphi 및 C++Builder 애플리케이션에 추가합니다. **TsgcVCLPivotGrid**와 **TsgcFMXPivotGrid**는 행, 열, 필터, 값에 배치한 필드를 기준으로 평면 소스 행을 요약합니다. 합계, 개수, 평균 등의 집계, 소계와 총합계, 연, 분기, 월, 주, 일 단위의 날짜 그룹화, 필터, 정렬, 확장과 축소, 소스 행으로의 드릴다운, CSV, JSON, HTML, PNG, SVG 내보내기를 지원합니다. **TsgcVCLPivotFieldList**와 **TsgcFMXPivotFieldList**를 사용하면 사용자가 끌어서 놓기로 레이아웃을 구성할 수 있습니다.
데이터는 **TsgcPivotSource_DataSet**을 통한 **TDataSet**, **TsgcPivotSource_Table**을 통한 sgcGrid 같은 모든 **IsgcUITable**, 또는 AppendRow를 사용하는 직접 작성한 코드에서 가져옵니다. 계산은 형식이 지정된 열 저장소를 가진 비시각적 클래스인 **TsgcPivotEngine**이 수행하므로, 같은 피벗을 창 없이 서버나 Linux64에서도 실행할 수 있습니다. **TsgcPivotGridRenderer**는 이를 PNG와 SVG로 그립니다. 소스 행 100만 개도 1초 미만에 다시 계산됩니다.
sgcPivotGrid는 eSeGeCe 비주얼 제품군의 일부입니다. 캔버스, 래스터라이저, 글꼴, SVG 작성기, 라이트 테마와 다크 테마를 갖춘 공유 런타임 패키지인 **sgcUICore**로 그리기 때문에, 피벗 그리드는 VCL, FireMonkey, 헤드리스 출력에서 모두 같은 모양이며 제품군의 다른 제품과도 어울립니다. sgcPivotGrid에는 sgcUICore가 필요하며, 함께 설치됩니다.



Year / Quarter / Month	2024 Total	2024 Q1	2024 Q2	2024 Q3	2024 Q4	2025 Total	2025 Q1	2025 Q2	2025 Q3	2025 Q4	2026 T
Region / Country	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	Sales	
Europe	20.756.578	4.984.943	5.042.241	5.185.631	5.543.762	20.643.562	4.934.052	5.241.520	5.213.853	5.254.138	20.630
France	5.290.220	1.407.831	1.199.268	1.341.003	1.342.119	5.072.432	1.223.091	1.290.259	1.255.724	1.303.358	5.054
Germany	6.684.319	1.465.704	1.807.951	1.718.902	1.691.761	6.763.931	1.565.470	1.744.882	1.751.883	1.701.696	6.711
Spain	3.464.296	828.076	771.630	911.564	953.026	3.629.538	937.602	867.726	933.378	890.833	3.311
United Kingdom	5.317.742	1.283.331	1.263.392	1.214.163	1.556.856	5.177.661	1.207.889	1.338.654	1.272.868	1.358.251	5.552
North America	19.945.562	4.874.876	4.993.579	5.153.763	4.923.344	19.968.850	4.923.029	5.066.623	5.021.217	4.957.982	20.268
Canada	3.921.244	911.275	1.040.780	961.345	1.007.844	3.880.529	954.084	1.102.726	901.344	922.375	3.940
Mexico	2.691.393	759.807	652.675	649.283	629.627	2.774.168	650.943	682.539	785.147	655.538	2.847
United States	13.332.925	3.203.794	3.300.123	3.543.135	3.285.873	13.314.154	3.318.002	3.281.358	3.334.725	3.380.068	13.480
Asia	14.686.504	3.441.225	3.855.867	3.435.858	3.953.554	14.427.002	3.518.143	3.479.165	3.846.128	3.583.567	14.473
China	4.612.990	1.034.093	1.192.559	1.155.411	1.230.926	4.544.570	1.161.154	958.509	1.205.425	1.219.483	4.457
India	4.147.745	1.055.857	1.119.897	904.630	1.067.362	3.994.930	1.043.263	1.026.835	1.015.401	909.430	4.053
Japan	5.925.770	1.351.275	1.543.411	1.375.817	1.655.266	5.887.503	1.313.726	1.493.821	1.625.302	1.454.655	5.962
South America	3.523.789	832.648	892.644	975.358	823.139	3.367.255	786.206	857.362	962.279	761.408	3.368
Brazil	3.523.789	832.648	892.644	975.358	823.139	3.367.255	786.206	857.362	962.279	761.408	3.368
Oceania	1.865.841	521.311	471.429	446.116	426.985	2.095.275	569.576	512.599	523.608	489.491	1.968
Australia	1.865.841	521.311	471.429	446.116	426.985	2.095.275	569.576	512.599	523.608	489.491	1.968
Grand Total	60.778.274	14.655.003	15.255.761	15.196.727	15.670.785	60.501.944	14.731.005	15.157.269	15.567.084	15.046.586	60.710

에디션

sgcPivotGrid는 Registered, Trial, Community로 제공됩니다. Community 에디션에서는 계산 필드, 값 표시 형식, Top N, 레이아웃 저장이라는 네 가지 고급 기능이 빠집니다. 에디션을 참조하십시오.

시작하는 곳

- 아키텍처: 소스, 열 저장소, 엔진 파이프라인, 스냅샷, 뷰.
- 플랫폼: Delphi 버전, C++Builder, 지원 플랫폼.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder, 빠른 시작 DataSet, 빠른 시작 헤드리스: 몇 분 만에 첫 피벗을 만듭니다.
- 컴포넌트: 그리드, 필드 목록, 엔진과 그 필드, 소스, 렌더러, 서식 규칙.
- 영역, 집계, 날짜 그룹화, 합계, 필터, Top N, 값 표시 형식, 계산 필드, 레이아웃 저장, 내보내기.
- 확장과 축소, 정렬, 드릴다운, 조건부 서식, 차트 연동, 키보드와 접근성, 성능.
- 데모: Delphi와 C++Builder 데모 프로젝트.
- 문제 해결: 흔한 문제와 그 원인.

아키텍처

sgcPivotGrid는 계산과 그리기를 분리하고, 그리기와 프레임워크도 분리합니다. 엔진과 렌더러는 RTL과 sgcUICore만 사용하며, VCL과 FMX 컨트롤은 이들을 감싸는 얇은 호스트입니다.

데이터 흐름

1. **소스.** TsgcPivotSource_DataSet은 TDataSet을, TsgcPivotSource_Table은 모든 IsgcUITable을 읽으며, 코드에서 TsgcPivotEngine의 AppendRow를 직접 호출할 수도 있습니다. 모두 같은 열 저장소에 기록합니다.
2. **열 저장소.** 텍스트 열은 사전 인코딩됩니다. 숫자, 날짜, 부울은 Double 값이며 NaN이 빈 값입니다. 열의 형식은 DB 필드 형식, DefineColumn, 또는 비어 있지 않은 첫 번째 값에서 정해지며, 정수 열은 소수 값이 들어오면 실수 열이 됩니다.
3. **엔진.** 재계산할 때마다 행, 열, 필터 필드의 멤버를 인코딩하고, 필터를 적용하고, 소스 행을 한 번 훑으며 리프 셀을 집계하고, 값 필터와 Top N을 적용하고, 정확한 병합으로 리프를 소계와 총합계로 롤업하고, 멤버를 정렬하고, 확장 상태에 따라 축을 평탄화합니다. 합계는 소스 값에서 계산되며 평균의 평균이 되는 일은 없습니다.
4. **스냅샷.** 결과는 변경할 수 없는 TsgcPivotResult로 게시됩니다. 읽는 쪽은 Lock과 Unlock 사이에서 이를 가져가므로 뷰는 항상 완전한 결과를 그립니다. 멤버를 확장하거나 축소하면 다시 집계하지 않고 같은 데이터에서 새 스냅샷이 만들어집니다.
5. **출력.** TsgcPivotGridRenderer는 화면, PNG, SVG 같은 모든 sgcUICore 캔버스에 스냅샷을 그립니다. TsgcVCLPivotGrid와 TsgcFMXPivotGrid는 스크롤, 선택, 키보드, 마우스를 갖추고 렌더러를 호스팅합니다. 내보내기 메서드는 뷰를 CSV, HTML, JSON으로 기록하며, ResultTable은 이를 IsgcUITable로 노출합니다. 예를 들어 sgcCharts에 전달할 수 있습니다.

클래스

클래스	유닛	역할
TsgcVCLPivotGrid	sgcPivotGrid_VCL	VCL 피벗 그리드. Pivot으로 게시되는 엔진을 소유합니다.
TsgcFMXPivotGrid	sgcPivotGrid_FMX	FireMonkey 피벗 그리드.
TsgcVCLPivotFieldList	sgcPivotGrid_FieldList_VCL	그리드 또는 엔진에 연결된 VCL 필드 목록.
TsgcFMXPivotFieldList	sgcPivotGrid_FieldList_FMX	FireMonkey 필드 목록.
TsgcPivotEngine	sgcPivotGrid_Engine	열 저장소, 레이아웃, 계산, 스냅샷, 드릴다운, 내보내기.
TsgcPivotField	sgcPivotGrid_Fields	레이아웃의 필드: 영역, 그룹화, 집계, 정렬, 필터.
TsgcPivotCalculatedField	sgcPivotGrid_Fields	식으로 소스 행마다 계산되는 열.
TsgcPivotSource_DataSet	sgcPivotGrid_Source_DataSet	TDataSet을 불러오고 데이터가 바뀌면 다시 불러옵니다.
TsgcPivotSource_Table	sgcPivotGrid_Source_Table	모든 IsgcUITable을 불러오고 그 버전이 바뀌면 다시 불러옵니다.
TsgcPivotGridRenderer	sgcPivotGrid_Render	스냅샷을 캔버스, PNG, SVG에 그리고 히트 테스트를 합니다.
TsgcPivotFormatRule	sgcPivotGrid_Format	조건부 서식 규칙: 값 규칙 또는 색 눈금.

스레드

재계산은 이를 일으킨 스레드(보통 메인 스레드)에서 실행되며 OnCalculated도 그 스레드에서 발생합니다. 스냅샷은 잠금으로 보호되므로 다른 스레드의 코드는 Lock과 Unlock 사이에서만 Result를 읽습니다. TsgcPivotSource_Table은 테이블의 변경을 알린 스레드에서 다시 불러옵니다.

성능

QA 성능 그룹은 릴리스 빌드로 이 시나리오들을 측정합니다. 워밍업 실행 한 번 후 다섯 번 실행한 값의 중앙값을 취하며, 모든 실행이 같도록 고정된 시드의 데이터를 사용합니다. 표에 각 시나리오의 예산이 나와 있습니다.

시나리오	Win64	Win32
100만 행 재계산: 행 필드 2개(멤버 50개와 20개), 열 필드 1개(멤버 12개), Sum 1개	100 ms	160 ms
텍스트 3개, 실수 1개, 날짜 1개의 값을 가진 100만 행의 AppendRow	900 ms	1,400 ms
값이 10만 개인 열에 CountDistinct를 사용한 첫 레이아웃	400 ms	600 ms
StdDev, Average, Max의 세 측정값을 사용한 첫 레이아웃	180 ms	280 ms
1만 행과 데이터 열 50개를 매 프레임 스크롤하며 1920x1080으로 그리는 1프레임	16 ms	25 ms
항목이 1만 개인 축의 멤버 하나를 확장하거나 축소	5 ms	8 ms
첫 번째 시나리오에서 저장소를 제외한 엔진 메모리	32 MB	32 MB

관련 항목

- 플랫폼
- TsgcPivotEngine
- 빠른 시작 헤드리스

에디션

sgcPivotGrid는 모든 에디션에서 같은 소스로 빌드됩니다.

에디션	포함 내용
Registered	모든 것: VCL과 FMX 그리드 및 필드 목록, 엔진, 두 소스, 모든 집계, 날짜 그룹화, 합계, 필터, 정렬, 드릴다운, 서식 규칙, 내보내기, 헤드리스 PNG와 SVG 출력, 그리고 네 가지 고급 기능인 계산 필드, 값 표시 형식, Top N, 레이아웃 저장.
Trial	Registered와 같은 기능입니다. SGC_TRIAL로 컴파일된 빌드는 빌드에 명시된 릴리스 기간이 지난 후 컴포넌트를 만들 때 체험판이라는 메시지를 표시합니다. 그 밖에는 달라지는 것이 없습니다.
Community	네 가지 고급 기능을 제외한 모든 것. CountDistinct, StdDev, Var, 날짜 그룹화, 필터, 드릴다운, 내보내기, 서식 규칙, 필드 목록, 뷰는 모두 Community에 포함되며 체험판 안내는 없습니다.

고급 기능

고급 기능은 SGC_PIVOTGRID_ADVANCED 정의 아래에서 컴파일되며, sgcVer.inc는 Community 에디션에서 이를 끕니다. 그 유닛인 sgcPivotGrid_Expr, sgcPivotGrid_ShowAs, sgcPivotGrid_Layout은 Community 배포판에 포함되지 않으며, 다음 멤버는 Community 빌드에 존재하지 않습니다.

기능	멤버
계산 필드	TsgcPivotEngine.CalculatedFields, TsgcPivotCalculatedField, TsgcPivotCalculatedFields
값 표시 형식	TsgcPivotField.ShowAs, ShowAsAxis, ShowAsBaseItem
Top N	TsgcPivotField.TopN과 TopNOthers, TsgcPivotEngine.OthersCaption
레이아웃 저장	엔진과 두 그리드의 SaveLayoutToString, LoadLayoutFromString, SaveLayoutToFile, LoadLayoutFromFile

에디션은 sgcVer.inc에서의 컴파일 시점 선택입니다. SGC_COMMUNITY는 SGC_TRIAL도 끕니다. 고급 멤버를 사용하면서 Community에서도 컴파일되어야 하는 코드는 {$IFDEF SGC_PIVOTGRID_ADVANCED} 안에 둡니다. Registered 빌드에서 고급 속성을 설정하여 저장한 폼은 Community 빌드에서 불러올 수 없습니다.

관련 항목

- 플랫폼
- 소개

설치

sgcPivotGrid는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램이 모든 sgcUI 제품을 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi와 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcPivotGrid**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 소유한 제품을 반환하고, 설치 프로그램은 제품마다 확인란 하나를 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 잠금 해제합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE용으로 종속성 순서대로 사용자의 컴퓨터에서 패키지를 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcPivotGrid와 함께 제공되는 것

sgcUI 비주얼 제품의 공통 기반인 **sgcUICore**는 sgcPivotGrid와 함께 제공됩니다. 자체 확인란이 없으며, sgcPivotGrid를 선택하면 설치 프로그램이 자동으로 설치합니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드. sgcUI의 유일한 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지로, 제품과 IDE 버전마다 한 세트입니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛. 예: `LibD13` .
`Demos\PivotGrid`	sgcPivotGrid 데모. 데모를 참조하십시오.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 자체 경로가 필요 없습니다.

Trial과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 sgcPivotGrid를 포함한 모든 제품을 체험판 모드로 설치합니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 네 가지 고급 기능을 뺀 sgcPivotGrid가 포함됩니다. 에디션을 참조하십시오.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전용 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예는 `sgcPivotGridD13.dpk`입니다.

패키지	내용

`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcPivotGrid<V>` / `dclsgcPivotGrid<V>`	sgcPivotGrid 컴포넌트(런타임 / 디자인 타임).

종속성 순서대로 빌드하십시오. `sgcUICore`, 그 다음 `sgcPivotGrid`, 그 다음 `dclsgcPivotGrid` 입니다. 그런 다음 `dclsgcPivotGrid` 디자인 타임 패키지를 설치하고, 사용 중인 Delphi 버전의 라이브러리 경로(도구 > 옵션 > 언어 > Delphi 옵션 > 라이브러리 > 라이브러리 경로)에 `sgcUI\Source`를 추가합니다. C++Builder에서는 사용 중인 버전에 맞는 `.cbproj` 프로젝트(예: `sgcPivotGridC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

관련 항목

- 플랫폼
- 빠른 시작 VCL

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE부터 XE8, 10부터 10.4, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE부터 XE8, 10부터 10.4, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows의 TsgcVCLPivotGrid와 TsgcVCLPivotFieldList. Win32와 Win64, 모든 Delphi 버전.
FireMonkey	Delphi XE6 이상의 TsgcFMXPivotGrid와 TsgcFMXPivotFieldList(SGC_PIVOTGRID_FMX). FireMonkey와 sgcUICore만 사용하며 자체 스크롤 막대와 터치 이동을 갖춥니다.
Linux64	Delphi 10.2 이상의 프레임워크 중립 유닛(SGC_PIVOTGRID_LINUX): 엔진, 열 저장소, 두 소스, 내보내기, 레이아웃 JSON, 헤드리스 PNG와 SVG 출력을 갖춘 렌더러, 필드 목록 모델.
.NET	WinForms, WPF, Avalonia에서 같은 API를 제공하고 헤드리스 PNG 및 SVG 출력과 DataTable 소스를 갖춘 C# 포트이며, .NET Framework 4.7.2와 .NET 8, 9에서 동작합니다. .NET 개요를 참조하십시오.

패키지

sgcPivotGrid에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcPivotGridD13.dpk, C++Builder 13에서는 sgcPivotGridC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcPivotGrid	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, dbrtl, sgcUICore
sgcPivotGridVCL	런타임, VCL	sgcPivotGrid_VCL 및 sgcPivotGrid_FieldList_VCL (TsgcVCLPivotGrid, TsgcVCLPivotFieldList)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcPivotGrid
sgcPivotGridFMX	런타임, FireMonkey, XE6부터	sgcPivotGrid_FMX 및 sgcPivotGrid_FieldList_FMX (TsgcFMXPivotGrid, TsgcFMXPivotFieldList)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcPivotGrid
dclsgcPivotGrid	디자인 타임, VCL	**SGC PivotGrid** 팔레트 페이지에 등록하는 컴포넌트는 TsgcPivotSource_DataSet, TsgcPivotSource_Table, TsgcVCLPivotGrid 및 TsgcVCLPivotFieldList이며, 컴포넌트 편집기, FieldName용 드롭다운 목록, 컬렉션 편집기도 함께 등록합니다.	designide, rtl, vcl, sgcPivotGrid, sgcPivotGridVCL
dclsgcPivotGridFMX	디자인 타임, FireMonkey, XE6부터	**SGC PivotGrid** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXPivotGrid 및 TsgcFMXPivotFieldList이며, 각각의 컴포넌트 편집기도 함께 등록합니다.	designide, rtl, fmx, sgcPivotGrid, sgcPivotGridFMX, dclsgcPivotGrid

sgcPivotGridWS, dclsgcPivotGridWS	런타임 및 디자인 타임, sgcWebSockets 브리지	라이브 데이터 브리지, 프레임워크 독립.	rtl, sgcPivotGrid, sgcWebSockets

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcPivotGridFMX에는 dclsgcPivotGrid가 필요하므로 dclsgcPivotGrid를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcPivotGrid	sgcPivotGridVCL	sgcPivotGridFMX	dclsgcPivotGrid, dclsgcPivotGridFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcPivotGrid, sgcPivotGridVCL	sgcPivotGridFMX	dclsgcPivotGrid, dclsgcPivotGridFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcPivotGrid만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcPivotGrid만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcPivotGridC13.cbproj, sgcPivotGridVCLC13.cbproj, sgcPivotGridFMXC13.cbproj, dclsgcPivotGridC13.cbproj, dclsgcPivotGridFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXPivotGrid 및 TsgcFMXPivotFieldList)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcPivotGridVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcPivotGrid_FMX 및 sgcPivotGrid_FieldList_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcPivotGrid는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 피벗 엔진과 소스입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

버전 제약

- 모든 유닛은 Delphi 7에서 컴파일됩니다. 제네릭, 익명 메서드, 인라인 변수, 레코드 도우미를 사용하지 않습니다.
- C++Builder 헤더는 열거형의 고유한 접두사(pva, pvf, pvd, pvx 등)를 유지하므로 sgcGrid나 sgcCharts와 충돌하지 않습니다.

관련 항목

- 에디션
- 아키텍처

빠른 시작 VCL

사용 중인 Delphi 버전용 sgcPivotGrid 패키지를 설치합니다. 컴포넌트는 **SGC PivotGrid** 팔레트 페이지에 있습니다.

디자인 타임에

1. **TsgcVCLPivotGrid**를 폼에 놓고 Align을 alClient로 설정합니다.
2. **TsgcVCLPivotFieldList**를 놓고 Align을 alRight로 설정한 다음 PivotGrid 속성을 그리드로 설정합니다.
3. **TsgcPivotSource_DataSet**을 놓고 DataSet을 데이터셋으로, PivotGrid를 그리드로 설정하고 Active를 True로 합니다.
4. 그리드의 Pivot 속성을 펼치고 Pivot.Fields에 항목을 추가합니다. FieldName의 드롭다운 목록에서 열을 고르고 Area(pvaRow, pvaColumn, pvaFilter, pvaData)를 설정합니다. 데이터 필드에는 Aggregation도 있으며 기본값은 Sum입니다.
5. 애플리케이션을 실행합니다. 사용자는 필드 목록으로 필드를 재배열하고, 그리드는 변경할 때마다 다시 계산합니다.

코드에서

그리드는 자신의 엔진을 Pivot 속성으로 소유합니다. 각 브래킷이 한 번만 다시 계산하도록, 행은 BeginLoad와 EndLoad 사이에서 불러오고 레이아웃은 BeginUpdate와 EndUpdate 사이에서 바꾸십시오.

관련 항목

- TsgcVCLPivotGrid
- TsgcVCLPivotFieldList
- 빠른 시작 DataSet
- 영역

빠른 시작 FMX

FireMonkey 컴포넌트에는 Delphi XE6 이상이 필요합니다. FireMonkey 폼의 **SGC PivotGrid** 팔레트 페이지에 있습니다.

디자인 타임에

1. **TsgcFMXPivotGrid**를 폼에 놓고 Align을 Client로 설정합니다.
2. **TsgcFMXPivotFieldList**를 놓고 Align을 Right로 설정한 다음 PivotGrid 속성을 그리드로 설정합니다.
3. **TsgcPivotSource_DataSet** 또는 **TsgcPivotSource_Table**로 데이터를 연결하거나 코드로 불러오고, 빠른 시작 VCL에서처럼 Pivot.Fields에 필드를 추가합니다.

FMX 그리드는 자체 스크롤 막대를 그려 끄는 동안 표시하며, 마우스 휠과 터치 이동으로 스크롤합니다. FireMonkey가 이미 HitTest 속성을 게시하므로 히트 테스트 메서드는 HitAt입니다.

코드에서

관련 항목

- TsgcFMXPivotGrid
- TsgcFMXPivotFieldList
- 플랫폼

빠른 시작 C++Builder

사용 중인 C++Builder 버전용 sgcPivotGrid 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 경로와 라이브러리 경로에 추가한 다음, **SGC PivotGrid** 팔레트 페이지의 컴포넌트를 Delphi와 똑같이 사용합니다. 설치 프로그램이 두 경로를 IDE에 추가해 줍니다. sgcPivotGrid와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있으므로 sgcUICore에는 자체 경로가 필요 없습니다.

C++에서 알아 둘 점

- AppendRow는 Variant의 오픈 배열을 받으며, C++에서는 Variant 배열과 그 최대 인덱스로 전달합니다. 값 네 개라면 `AppendRow(Row, 3)`입니다.
- 생성된 헤더에는 TsgcPivotEngine과 TsgcPivotGridRenderer 생성자의 기본 인수가 없습니다. NULL을 명시적으로 전달하십시오: `new TsgcPivotEngine(NULL)`과 `new TsgcPivotGridRenderer(NULL, NULL)`. 빠른 시작 헤드리스를 참조하십시오.
- 이벤트는 클로저이므로 핸들러는 클래스의 __fastcall 멤버 함수입니다. aHandled 같은 var 매개변수는 bool 참조입니다.

코드에서

관련 항목

- TsgcVCLPivotGrid
- 빠른 시작 VCL
- 데모

빠른 시작 DataSet

TsgcPivotSource_DataSet은 모든 TDataSet의 레코드를 그리드의 엔진에 복사하고 피벗을 최신 상태로 유지합니다. 레이아웃은 데이터셋 필드를 FieldName으로 참조하므로 데이터가 도착하기 전에 정의할 수 있습니다.

소스가 불러오는 방식

- Columns는 불러올 데이터셋 필드를 나열합니다. 비어 있으면 모든 필드를 불러옵니다. BLOB, 이진, 중첩 필드는 항상 건너뜁니다.
- 열 형식은 필드 형식에서 정해집니다. 문자열 필드는 텍스트, 정수와 실수 필드는 숫자, 날짜와 시간 필드는 날짜, 부울 필드는 부울입니다.
- 불러오기는 DisableControls를 사용해 실행되고 북마크를 복원하므로 데이터셋에 연결된 컨트롤이 깜박이지 않습니다.
- LiveUpdates를 사용하면 TDataSet이 데이터 변경으로 알리는 Post, Append, Delete, Refresh, First, Last로 피벗이 다시 불러옵니다. Next, Prior, MoveBy로 스크롤할 때는 다시 불러오지 않습니다. 불러오기는 데이터셋이 탐색 상태일 때만 실행됩니다.
- Load는 언제든 다시 불러오고, Refresh는 소스가 Active일 때 다시 불러옵니다.

TClientDataSet을 사용하는 C++Builder 프로젝트는 DataSetPivot 데모처럼 `#pragma link "midas.lib"`로 midas.lib를 링크합니다.

관련 항목

- TsgcPivotSource_DataSet
- TsgcPivotSource_Table
- 날짜 그룹화

빠른 시작 헤드리스

피벗에는 창이 필요하지 않습니다. TsgcPivotEngine이 행을 불러와 계산하고, 내보내기 메서드가 CSV, HTML, JSON을 기록하며, TsgcPivotGridRenderer가 컨트롤과 같은 그리드를 PNG 또는 SVG로 그립니다. 이 유닛들은 RTL과 sgcUICore만 사용하므로 보고서를 서비스, 웹 서버, Linux64에서 실행할 수 있습니다.

참고

- sgcPivotRenderState는 빈 뷰 상태를 반환합니다. 스크롤, 선택, 열 너비 재정의가 없습니다. 엔진을 받는 렌더러 오버로드는 그리는 동안 엔진을 잠급니다.
- 너비와 높이를 0으로 지정하면 전체 내용을 그립니다. 크기는 DIP 단위이며 scale 인수가 여기에 곱해집니다.

관련 항목

- 내보내기
- TsgcPivotGridRenderer
- 데모

데모

데모는 `Demos\PivotGrid`에 설치되며, Delphi 프로젝트는 `Delphi`에, C++Builder 포팅은 `CBuilder`에 있습니다. 모든 데모는 자체 테스트를 실행하고 종료하는 `--test`와, 피벗 그림을 저장하는 `/snapshot <file.png>`를 지원합니다.

데모	프레임워크	보여 주는 내용
SalesPivot	VCL	판매 행 10만 개: 연, 분기, 월 단위의 날짜 그룹화, 확장과 축소, 값 기준 정렬, 색 눈금, 필드 목록, CSV, HTML, PNG 내보내기.
DataSetPivot	VCL	TClientDataSet 위의 피벗: TDBGrid에서 편집, 추가, 삭제하면 다시 불러오고, 셀을 더블 클릭하면 원본 주문으로 드릴다운합니다.
MillionRows	VCL	하나의 로드 브래킷으로 100만 행을 추가하고, 값이 10만 개인 열의 CountDistinct와 Sum 옆의 StdDev를 실행하며, 재계산 시간을 측정합니다.
AdvancedPivot	VCL	계산 필드 Profit, 값 표시 형식, Others가 있는 Top N, JSON 파일로 저장하고 불러오는 레이아웃.
GridToPivotToChart	VCL	sgcGrid에 입력한 주문이 TsgcPivotSource_Table을 통해 피벗에 전달되고, 피벗 결과가 IsgcUITable을 통해 sgcCharts에 전달됩니다.
SalesPivotFMX	FireMonkey	SalesPivot 레이아웃으로 5만 행, 터치 이동, 다크 테마, PNG 내보내기.
HeadlessReport	콘솔	CSV 파일을 읽고 UI 프레임워크 없이 report.png, report.svg, report.html, report.csv, report.json을 기록합니다.
AskYourPivot	VCL	자연어 질문이 TsgcPivotGridAIQuery와 TsgcAIChat을 통해 피벗 레이아웃이 됩니다: 제공자, 모델, API 키 입력란, Ask, Undo, 샘플 질문, Show prompt. Registered 에디션, sgcWebSockets가 필요합니다. 피벗에 질문하기을 참조하십시오.

관련 항목

- 빠른 시작 VCL
- 빠른 시작 헤드리스
- 피벗에 질문하기

TsgcVCLPivotGrid

유닛: sgcPivotGrid_VCL
TsgcVCLPivotGrid는 VCL 피벗 그리드이며 Windows 전용입니다. TsgcPivotEngine을 Pivot으로 게시하여 소유하며, 그 결과를 가상 행과 열로 그립니다. 고정된 모서리, 병합이 있는 다단계 열 머리글, 행 머리글, + 버튼과 - 버튼, 합계와 총합계를 갖추고 있습니다. 네이티브 스크롤 막대와 마우스 휠(Shift는 가로 스크롤)로 스크롤하고, 머리글의 가장자리로 열 너비를 바꾸며, 키보드 인터페이스도 제공합니다. 방향키, PgUp, PgDn, Home, End로 선택을 이동하고, 행 머리글에 포커스가 있을 때 Right와 Left(또는 + 와 -)로 멤버를 확장하고 축소하며, Enter로 드릴다운합니다. TsgcUIVCLControl에서 상속하며 Align, Anchors, Constraints, Enabled, PopupMenu, ShowHint, ParentShowHint, TabOrder, TabStop, Visible을 게시합니다.

속성

속성	형식	설명
Pivot	TsgcPivotEngine	그리드가 소유하는 엔진입니다. 필드, 데이터, 합계, 캡션이 폼과 함께 스트리밍됩니다. 그리드는 엔진에 위임하여 IsgcUITable을 구현합니다.
BorderStyle	TBorderStyle	bsSingle(기본값) 또는 bsNone.
RowLayout	TsgcPivotRowLayout	pvlCompact(기본값)는 모든 행 수준을 들여쓰기가 있는 하나의 헤더 열에 그리고, pvlTabular는 각 수준에 전용 열을 줍니다. 영역을 참조하십시오.
RowHeight	Integer	행의 높이(DIP). 기본값은 24입니다.
HeaderHeight	Integer	열 머리글 행의 높이(DIP). 기본값은 26입니다.
IndentWidth	Integer	컴팩트 레이아웃에서 각 행 수준의 들여쓰기(DIP). 기본값은 16입니다.
MinColumnWidth, MaxColumnWidth	Integer	데이터 열 자동 너비의 하한과 상한(DIP). 기본값은 48과 320입니다.
AutoColumnWidths	Boolean	데이터 열을 내용에 맞춥니다. False이면 너비는 96 DIP입니다. 행 머리글 열은 항상 내용에 맞춥니다. 기본값은 True입니다.
GridLines	Boolean	셀 선을 그립니다. 기본값은 True입니다.
AlternateRows	Boolean	한 줄씩 건너 음영을 넣습니다. 기본값은 False입니다.
TotalsBold	Boolean	합계를 굵게 그립니다. 기본값은 True입니다.
ShowExpandButtons	Boolean	자식이 있는 멤버의 + 버튼과 - 버튼을 표시합니다. 기본값은 True입니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	$AARRGGBB 형식의 강조 색입니다. 0(기본값)이면 테마의 강조 색을 사용합니다.
FontSize	Double	셀의 글꼴 크기(포인트). 기본값은 12입니다.
FormatRules	TsgcPivotFormatRules	셀의 조건부 서식 규칙.
MaxFPS	Integer	컨트롤이 초당 그리는 최대 프레임 수.

SelectedRow	Integer	Public. 선택된 행 항목이며, 없으면 -1입니다.
SelectedCol	Integer	Public. 선택된 데이터 열이며, 행 머리글에 포커스가 있거나 선택된 것이 없으면 -1입니다.
TopRow	Integer	Public. 열 머리글 아래에 표시되는 첫 번째 행 항목.
ScrollX	Integer	Public. 데이터 열의 가로 스크롤 오프셋.
Renderer	TsgcPivotGridRenderer	Public, 읽기 전용. 그리드를 그리는 렌더러.

메서드

메서드	반환값	설명
HitTest(X, Y)	TsgcPivotHitInfo	장치 픽셀 단위의 한 점 아래에 있는 부분: Part(pvhNone, pvhCorner, pvhColumnHeader, pvhRowHeader, pvhCell, pvhExpandButton, pvhColumnResize), RowItem, DataCol, Level.
ScrollToCell(aRowItem, aDataCol)		셀이 보일 때까지 스크롤합니다. -1을 지정한 축은 그대로 둡니다.
Recalculate		엔진을 지금 다시 계산합니다.
SaveToPNG, SaveToSVG(aFileName or aStream, aWidth = 0, aHeight = 0)		전체 뷰를 이미지로 내보냅니다. 크기는 DIP 단위이며 0은 내용 크기이고 최대 16384픽셀입니다.
SaveToCSV, SaveToJSON, SaveToHTML(aFileName or aStream)		현재 뷰를 테이블로 내보냅니다. 내보내기를 참조하십시오.
SaveLayoutToString, LoadLayoutFromString, SaveLayoutToFile, LoadLayoutFromFile		그리드의 열 너비를 포함한 Pivot의 레이아웃을 JSON으로 내보냅니다. 고급 에디션 기능입니다. 레이아웃 저장을 참조하십시오.

이벤트

이벤트	형식	설명
OnCellClick, OnCellDblClick	TsgcVCLPivotHitEvent	클릭 또는 더블 클릭이며, 해당 지점의 히트 정보를 받습니다.
OnDrillDown	TsgcVCLPivotDrillDownEvent	셀에서 Enter 또는 더블 클릭. aRowItem, aDataCol, var aHandled를 받습니다. 그리드에는 기본 제공 동작이 없으므로, 소스 행을 읽으려면 Pivot.DrillDown 또는 Pivot.GetSourceRows를 호출하십시오.
OnExpandChange	TsgcVCLPivotExpandEvent	사용자가 멤버를 확장하거나 축소한 후에 발생합니다. aAxis와 aItem을 받습니다.
OnGetCellStyle	TsgcPivotCellStyleEvent	데이터 셀이 그려지기 전에, 서식 규칙이 적용된 후 발생합니다. aRowItem, aDataCol, var aBack, aFore, aBold를 받습니다.

OnGetCellText	TsgcPivotCellTextEvent	데이터 셀이 그려지기 전에 발생합니다. aRowItem, aDataCol, var aText(서식이 적용된 값이며, 셀이 비어 있으면 Pivot.EmptyText)를 받습니다.
OnSelectionChange	TNotifyEvent	SelectedRow 또는 SelectedCol이 변경되었습니다.
OnColumnResized	TsgcVCLPivotHitEvent	사용자가 머리글의 가장자리를 끌어 열 너비를 변경했습니다.

폼을 불러오는 동안 엔진은 자신을 변경됨으로만 표시하고, Pivot.AutoRecalculate가 True이면 그리드가 Loaded에서 한 번만 다시 계산합니다.

예제

관련 항목

- TsgcFMXPivotGrid
- TsgcVCLPivotFieldList
- 빠른 시작 VCL

TsgcFMXPivotGrid

유닛: sgcPivotGrid_FMX
TsgcFMXPivotGrid는 FireMonkey 피벗 그리드이며 Delphi XE6 이상(SGC_PIVOTGRID_FMX)에서 사용할 수 있습니다. 엔진, 레이아웃, 속성, 이벤트는 TsgcVCLPivotGrid와 같습니다. 차이는 FireMonkey에서 비롯됩니다. 자체 스크롤 막대를 그려 끄는 동안 표시하고, 마우스 휠과 터치 이동으로 스크롤하며, FireMonkey가 이미 HitTest 속성을 게시하므로 히트 테스트 메서드는 HitAt입니다. 이벤트 형식에는 TsgcFMXPivot 접두사가 붙습니다. TsgcUIFMXControl에서 상속하며 일반적인 FireMonkey 레이아웃 속성과 이벤트를 게시합니다.

속성

속성	형식	설명
Pivot	TsgcPivotEngine	그리드가 소유하는 엔진입니다. 필드, 데이터, 합계, 캡션이 폼과 함께 스트리밍됩니다. 그리드는 엔진에 위임하여 IsgcUITable을 구현합니다.
RowLayout	TsgcPivotRowLayout	pvlCompact(기본값)는 모든 행 수준을 들여쓰기가 있는 하나의 헤더 열에 그리고, pvlTabular는 각 수준에 전용 열을 줍니다. 영역을 참조하십시오.
RowHeight	Integer	행의 높이(DIP). 기본값은 24입니다.
HeaderHeight	Integer	열 머리글 행의 높이(DIP). 기본값은 26입니다.
IndentWidth	Integer	컴팩트 레이아웃에서 각 행 수준의 들여쓰기(DIP). 기본값은 16입니다.
MinColumnWidth, MaxColumnWidth	Integer	데이터 열 자동 너비의 하한과 상한(DIP). 기본값은 48과 320입니다.
AutoColumnWidths	Boolean	데이터 열을 내용에 맞춥니다. False이면 너비는 96 DIP입니다. 행 머리글 열은 항상 내용에 맞춥니다. 기본값은 True입니다.
GridLines	Boolean	셀 선을 그립니다. 기본값은 True입니다.
AlternateRows	Boolean	한 줄씩 건너 음영을 넣습니다. 기본값은 False입니다.
TotalsBold	Boolean	합계를 굵게 그립니다. 기본값은 True입니다.
ShowExpandButtons	Boolean	자식이 있는 멤버의 + 버튼과 - 버튼을 표시합니다. 기본값은 True입니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	$AARRGGBB 형식의 강조 색입니다. 0(기본값)이면 테마의 강조 색을 사용합니다.
FontSize	Double	셀의 글꼴 크기(포인트). 기본값은 12입니다.
FormatRules	TsgcPivotFormatRules	셀의 조건부 서식 규칙.
MaxFPS	Integer	컨트롤이 초당 그리는 최대 프레임 수.
SelectedRow	Integer	Public. 선택된 행 항목이며, 없으면 -1입니다.

SelectedCol	Integer	Public. 선택된 데이터 열이며, 행 머리글에 포커스가 있거나 선택된 것이 없으면 -1입니다.
TopRow	Integer	Public. 열 머리글 아래에 표시되는 첫 번째 행 항목.
ScrollX	Integer	Public. 데이터 열의 가로 스크롤 오프셋.
Renderer	TsgcPivotGridRenderer	Public, 읽기 전용. 그리드를 그리는 렌더러.
VertBarVisible, HorzBarVisible	Boolean	Public, 읽기 전용. 내용에 해당 스크롤 막대가 필요하면 True입니다.

메서드

메서드	반환값	설명
HitAt(X, Y)	TsgcPivotHitInfo	로컬 좌표의 한 점 아래에 있는 부분: Part(pvhNone, pvhCorner, pvhColumnHeader, pvhRowHeader, pvhCell, pvhExpandButton, pvhColumnResize), RowItem, DataCol, Level.
ScrollToCell(aRowItem, aDataCol)		셀이 보일 때까지 스크롤합니다. -1을 지정한 축은 그대로 둡니다.
Recalculate		엔진을 지금 다시 계산합니다.
SaveToPNG, SaveToSVG(aFileName or aStream, aWidth = 0, aHeight = 0)		전체 뷰를 이미지로 내보냅니다. 크기는 DIP 단위이며 0은 내용 크기이고 최대 16384픽셀입니다.
SaveToCSV, SaveToJSON, SaveToHTML(aFileName or aStream)		현재 뷰를 테이블로 내보냅니다. 내보내기를 참조하십시오.
SaveLayoutToString, LoadLayoutFromString, SaveLayoutToFile, LoadLayoutFromFile		그리드의 열 너비를 포함한 Pivot의 레이아웃을 JSON으로 내보냅니다. 고급 에디션 기능입니다. 레이아웃 저장을 참조하십시오.

이벤트

이벤트	형식	설명
OnCellClick, OnCellDblClick	TsgcFMXPivotHitEvent	클릭 또는 더블 클릭이며, 해당 지점의 히트 정보를 받습니다.
OnDrillDown	TsgcFMXPivotDrillDownEvent	셀에서 Enter 또는 더블 클릭. aRowItem, aDataCol, var aHandled를 받습니다. 그리드에는 기본 제공 동작이 없으므로, 소스 행을 읽으려면 Pivot.DrillDown 또는 Pivot.GetSourceRows를 호출하십시오.
OnExpandChange	TsgcFMXPivotExpandEvent	사용자가 멤버를 확장하거나 축소한 후에 발생합니다. aAxis와 aItem을 받습니다.
OnGetCellStyle	TsgcPivotCellStyleEvent	데이터 셀이 그려지기 전에, 서식 규칙이 적용된 후 발생합니다. aRowItem, aDataCol, var aBack, aFore, aBold를 받습니다.
OnGetCellText	TsgcPivotCellTextEvent	데이터 셀이 그려지기 전에 발생합니다. aRowItem, aDataCol, var aText(서식이 적용된 값이며, 셀이 비어 있으면 Pivot.EmptyText)를 받습니다.
OnSelectionChange	TNotifyEvent	SelectedRow 또는 SelectedCol이 변경되었습니다.
OnColumnResized	TsgcFMXPivotHitEvent	사용자가 머리글의 가장자리를 끌어 열 너비를 변경했습니다.

관련 항목

- TsgcVCLPivotGrid
- TsgcFMXPivotFieldList
- 빠른 시작 FMX

TsgcVCLPivotFieldList

유닛: sgcPivotGrid_FieldList_VCL
TsgcVCLPivotFieldList는 VCL 필드 목록입니다. 피벗 그리드에 연결하면 사용자가 끌어서 놓기로 레이아웃을 구성할 수 있습니다. TsgcUIVCLControl에서 상속하며 일반적인 VCL 레이아웃 속성을 게시합니다. 멤버 체크리스트는 TsgcVCLPivotMemberPopup이고 집계 메뉴는 TPopupMenu입니다.

사용자가 사용하는 방법

- 소스 열이 맨 위에 나열되며, 레이아웃에 포함된 열에는 확인 표시가 붙습니다. 그 아래에는 필터, 열, 행, 값의 네 영역이 있습니다.
- 열을 영역으로 끌어 놓으면 필드가 추가됩니다. 칩을 다른 영역으로 끌면 이동하고, 영역 밖으로 끌면 제거됩니다.
- 값 영역에 놓은 열은 텍스트 열이면 Count, 그렇지 않으면 Sum이 됩니다. 값 칩의 버튼을 누르면 모든 집계가 나열된 메뉴가 열립니다.
- 필터, 열, 행 칩의 필터 버튼을 누르면 검색 상자와 모두 선택이 있는 멤버 체크리스트가 열립니다. 확인을 누르면 FilterValues와 FilterExclude에 기록됩니다. 필터를 참조하십시오.
- 레이블 필터와 값 필터는 코드 또는 개체 인스펙터에서 설정합니다.

속성

속성	형식	설명
PivotGrid	TComponent	목록이 엔진을 편집하는 피벗 그리드로, TsgcVCLPivotGrid 또는 TsgcFMXPivotGrid입니다.
Engine	TsgcPivotEngine	Public. 코드에서 설정한 엔진은 헤드리스 엔진을 위해 PivotGrid의 엔진보다 우선합니다.
ShowFilterArea	Boolean	필터 영역을 표시합니다. 기본값은 True입니다.
DeferUpdate	Boolean	사용자의 변경을 모아 두었다가 ApplyChanges에서만 한 번의 재계산으로 적용합니다. 기본값은 False입니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	$AARRGGBB 형식의 강조 색입니다. 0(기본값)이면 테마의 강조 색을 사용합니다.
FontSize	Double	글꼴 크기(포인트).
MaxFPS	Integer	컨트롤이 초당 그리는 최대 프레임 수.

메서드

메서드	반환값	설명
ApplyChanges		DeferUpdate가 True일 때 보류 중인 레이아웃을 엔진에 할당합니다.

이벤트

이벤트	형식	설명

OnBeforeFieldMove	TsgcPivotBeforeFieldMoveEvent	사용자가 필드를 이동하기 전에 발생합니다. aFieldName, aFrom, aTo, var aAllow를 받습니다. 소스 열을 영역에 놓을 때는 aFrom과 aTo가 같습니다. 필드를 제거할 때는 이동 이벤트가 발생하지 않습니다.
OnFieldMoved	TsgcPivotFieldMovedEvent	필드가 추가되거나 이동된 후에 발생합니다. aField를 받습니다.

예제

관련 항목

- TsgcFMXPivotFieldList
- 영역

TsgcFMXPivotFieldList

유닛: sgcPivotGrid_FieldList_FMX
TsgcFMXPivotFieldList는 FireMonkey 필드 목록이며 Delphi XE6 이상(SGC_PIVOTGRID_FMX)에서 사용할 수 있습니다. 영역, 끌어서 놓기, 필터 규칙은 TsgcVCLPivotFieldList와 같습니다. 팝업은 같은 칩 위에서 버튼을 놓을 때 열립니다.

사용자가 사용하는 방법

- 소스 열이 맨 위에 나열되며, 레이아웃에 포함된 열에는 확인 표시가 붙습니다. 그 아래에는 필터, 열, 행, 값의 네 영역이 있습니다.
- 열을 영역으로 끌어 놓으면 필드가 추가됩니다. 칩을 다른 영역으로 끌면 이동하고, 영역 밖으로 끌면 제거됩니다.
- 값 영역에 놓은 열은 텍스트 열이면 Count, 그렇지 않으면 Sum이 됩니다. 값 칩의 버튼을 누르면 모든 집계가 나열된 메뉴가 열립니다.
- 필터, 열, 행 칩의 필터 버튼을 누르면 검색 상자와 모두 선택이 있는 멤버 체크리스트가 열립니다. 확인을 누르면 FilterValues와 FilterExclude에 기록됩니다. 필터를 참조하십시오.
- 레이블 필터와 값 필터는 코드 또는 개체 인스펙터에서 설정합니다.

속성

속성	형식	설명
PivotGrid	TComponent	목록이 엔진을 편집하는 피벗 그리드로, TsgcVCLPivotGrid 또는 TsgcFMXPivotGrid입니다.
Engine	TsgcPivotEngine	Public. 코드에서 설정한 엔진은 헤드리스 엔진을 위해 PivotGrid의 엔진보다 우선합니다.
ShowFilterArea	Boolean	필터 영역을 표시합니다. 기본값은 True입니다.
DeferUpdate	Boolean	사용자의 변경을 모아 두었다가 ApplyChanges에서만 한 번의 재계산으로 적용합니다. 기본값은 False입니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom.
AccentColor	TsgcUIColor	$AARRGGBB 형식의 강조 색입니다. 0(기본값)이면 테마의 강조 색을 사용합니다.
FontSize	Double	글꼴 크기(포인트).
MaxFPS	Integer	컨트롤이 초당 그리는 최대 프레임 수.
Model	TsgcPivotFieldListModel	Public, 읽기 전용. 영역, 칩, 끌기 상태를 가진 프레임워크 중립 모델.
AggregationMenu, FilterPopup, FilterSearch, FilterSelectAll, FilterList	메뉴 및 팝업 컨트롤	Public, 읽기 전용. 집계 메뉴와 멤버 체크리스트의 구성 요소이며, 처음 표시될 때 만들어집니다.

메서드

메서드	반환값	설명

ApplyChanges		DeferUpdate가 True일 때 보류 중인 레이아웃을 한 번의 재계산으로 엔진에 할당합니다.
HitAt(X, Y)	TsgcPivotFieldListHitInfo	로컬 좌표의 한 점 아래에 있는 목록의 부분.
ShowAggregationMenu(aField)		값 필드의 집계 메뉴를 엽니다.
ShowMemberFilter(aField)		필터, 열, 행 필드의 멤버 체크리스트를 엽니다.
CloseMemberFilter(aApply)		멤버 체크리스트를 닫습니다. aApply를 지정하면 선택한 멤버를 필터에 기록합니다.

이벤트

이벤트	형식	설명
OnBeforeFieldMove	TsgcPivotBeforeFieldMoveEvent	사용자가 필드를 이동하기 전에 발생합니다. aFieldName, aFrom, aTo, var aAllow를 받습니다. 소스 열을 영역에 놓을 때는 aFrom과 aTo가 같습니다. 필드를 제거할 때는 이동 이벤트가 발생하지 않습니다.
OnFieldMoved	TsgcPivotFieldMovedEvent	필드가 추가되거나 이동된 후에 발생합니다. aField를 받습니다.

관련 항목

- TsgcVCLPivotFieldList
- TsgcFMXPivotGrid

TsgcPivotEngine

유닛: sgcPivotGrid_Engine
TsgcPivotEngine은 소스 행을 형식이 지정된 열 저장소에 보관하며, 레이아웃과 계산 결과도 보관합니다. 그리드는 Pivot으로 게시되는 엔진을 하나 소유하며, 엔진은 UI 없이 단독으로도 동작합니다. TPersistent에서 파생되고 참조 카운트 없이 IsgcUITable을 구현하며, `Create(aOwner: TPersistent = nil)`로 생성합니다. C++에서는 NULL을 명시적으로 전달하십시오.

속성

속성	형식	설명
필드	TsgcPivotFields	네 개의 영역에 배치된 레이아웃의 필드.
CalculatedFields	TsgcPivotCalculatedFields	계산 필드. 고급 에디션 기능입니다.
RowGrandTotals, ColumnGrandTotals	Boolean	총합계 행과 총합계 열을 표시합니다. 기본값은 True입니다.
SubtotalPosition	TsgcPivotSubtotalPosition	pvpTop(기본값) 또는 pvpBottom. 합계를 참조하십시오.
AutoRecalculate	Boolean	데이터나 레이아웃이 바뀔 때마다 브래킷당 한 번 다시 계산합니다. False이면 Recalculate를 호출하십시오. 기본값은 True입니다.
GrandTotalCaption	string	총합계의 캡션입니다. 기본값은 Grand Total입니다.
BlankCaption	string	빈 멤버의 레이블입니다. 기본값은 (blank)입니다.
EmptyText	string	값이 없는 데이터 셀의 텍스트이며, 기본값은 빈 문자열입니다. 그리드, 렌더러, HTML 내보내기, ResultTable이 이를 표시하고, OnGetCellText가 이를 받아 대체할 수도 있습니다. SaveToCSV는 빈 필드를, SaveToJSON은 null을 기록합니다. 이 값을 바꾸면 다시 계산하지 않고 뷰를 다시 게시합니다.
OthersCaption	string	Top N의 Others 멤버 레이블입니다. 기본값은 Others입니다. 고급 에디션 기능입니다.
MaxCells	Integer	레이아웃이 만들 수 있는 셀의 최대 개수입니다. 이보다 많이 필요한 재계산은 EsgcPivotError를 발생시킵니다. 기본값은 5,000,000입니다.
Result	TsgcPivotResult	Public, 읽기 전용. 현재 스냅샷이며, 첫 계산 전에는 nil입니다. Lock과 Unlock 사이에서만 읽으십시오.
ResultTable	TsgcPivotResultTable	Public, 읽기 전용. 결과를 IsgcUITable로 본 뷰입니다. 내보내기를 참조하십시오.
Store	TsgcPivotStore	Public, 읽기 전용. 소스가 사용하는 열 저장소.
RowLayout	TsgcPivotRowLayout	Public. 그리드가 설정합니다. 컴팩트 레이아웃에서는 부모 항목이 자식보다 먼저 나옵니다.

메서드: 데이터

메서드	반환값	설명
BeginLoad(aClear = True), EndLoad		로드 브래킷입니다. aClear를 지정하면 먼저 저장소를 비웁니다. 엔진은 가장 바깥쪽 EndLoad에서 한 번만 다시 계산합니다.
DefineColumn(aName), DefineColumn(aName, aType)	Integer	열을 추가합니다. 형식을 생략하면 비어 있지 않은 첫 번째 값이 형식을 결정합니다. 이름은 대소문자를 구분하지 않고 고유해야 합니다.
AppendRow(aValues)	Integer	열 순서대로 Variant 값의 행을 추가합니다. Null과 변환할 수 없는 값은 빈 값이 됩니다.
ClearData		모든 행과 열을 제거합니다.
LoadFromTable(aTable)		IsgcUITable의 모든 열과 행을 불러옵니다.
SourceRowCount, SourceColumnCount	Integer	저장소의 크기.
SourceColumnName(aIndex)	string	열의 이름.
SourceValue(aRow, aCol)	Variant	저장된 값이며, 비어 있으면 Null입니다.
GetMemberLabels(aField, aList)		aField의 멤버 필터가 비교하는 멤버 레이블로, 표시 순서이며 값이 비어 있으면 마지막에 BlankCaption이 옵니다.

메서드: 레이아웃과 뷰

메서드	반환값	설명
BeginUpdate, EndUpdate		레이아웃 브래킷이며, EndUpdate에서 한 번 다시 계산합니다.
Recalculate		지금 다시 계산합니다.
AddField(aFieldName, aArea, aGrouping = pvdNone)	TsgcPivotField	행, 열, 필터 또는 데이터 필드를 해당 영역의 끝에 추가합니다.
AddDataField(aFieldName, aAggregation = pvfSum)	TsgcPivotField	측정값을 추가합니다.
MoveField(aField, aArea, aIndex)		필드를 다른 영역이나 위치로 이동합니다.
RemoveField(aField), ClearLayout		필드 하나 또는 모든 필드를 제거합니다.
ExpandAll(aAxis), CollapseAll(aAxis)		pvxRows 또는 pvxColumns의 모든 멤버를 확장하거나 축소합니다.
ExpandLevel(aAxis, aLevel), CollapseLevel(aAxis, aLevel)		한 수준의 모든 멤버를 확장하거나 축소합니다.
SetExpanded(aAxis, aItem, aValue), ToggleExpanded(aAxis, aItem)		현재 스냅샷의 항목 하나를 변경합니다. 확장 상태는 멤버 경로별로 유지되므로 재계산 후에도 남습니다. 확장과 축소를 참조하십시오.

GetExpandState(aAxis, aState), SetExpandState(aAxis, aState)		축의 명시적 확장 상태를 문자열 목록으로 다룹니다. 고급 에디션 기능입니다.

메서드: 결과, 드릴다운, 내보내기

메서드	반환값	설명
Lock, Unlock		Result를 읽는 동안 스냅샷을 보호합니다.
GetSourceRows(aRowItem, aDataCol, var aRows)	Integer	셀 뒤에 있는 소스 행을 행 순서대로 반환합니다. aDataCol이 -1이면 모든 열을 대상으로 하며, 합계는 그 아래의 모든 행을 대상으로 합니다.
DrillDown(aRowItem, aDataCol)	TsgcPivotSourceView	같은 행을 저장소의 모든 열을 가진 IsgcUITable로 반환합니다. 호출자가 엔진보다 먼저 해제해야 합니다. 데이터가 바뀐 후에는 행이 없습니다.
SaveToCSV, SaveToJSON, SaveToHTML		현재 뷰를 파일이나 스트림으로 내보냅니다. 내보내기를 참조하십시오.
SaveLayoutToString, LoadLayoutFromString, SaveLayoutToFile, LoadLayoutFromFile		JSON 형식의 레이아웃입니다. 고급 에디션 기능입니다. 레이아웃 저장을 참조하십시오.
Assign(Source)		다른 엔진의 레이아웃과 설정을 복사하며, 데이터는 복사하지 않습니다.

이벤트

이벤트	형식	설명
OnCalculated	TNotifyEvent	새 결과가 게시된 후, 재계산을 수행한 스레드에서 발생합니다.
OnLayoutChange	TNotifyEvent	필드 또는 레이아웃 설정이 변경된 후에 발생합니다.
OnFilterRow	TsgcPivotFilterRowEvent	할당되어 있는 동안 소스 행마다 발생합니다. aRow와 var aAccept를 받습니다. 필터를 참조하십시오.

예제

결과 읽기: Lock과 Unlock 사이에서 행 레이블과 셀 텍스트를 읽습니다.

관련 항목

- TsgcPivotField
- 빠른 시작 헤드리스
- 아키텍처

TsgcPivotField

유닛: sgcPivotGrid_Fields
TsgcPivotField는 TsgcPivotEngine.Fields(TsgcPivotFields)의 항목으로, 레이아웃의 필드 하나입니다. 같은 FieldName이 여러 필드에 나타날 수 있습니다.

속성

속성	형식	설명
FieldName	string	소스 열 또는 계산 필드.
Caption	string	수준 또는 측정값의 캡션입니다. 비어 있으면 DisplayCaption을 사용합니다.
DisplayCaption	string	Public, 읽기 전용. Caption, 없으면 데이터 필드의 경우 Sum of Sales, 그것도 없으면 FieldName.
Area	TsgcPivotArea	pvaRow(기본값), pvaColumn, pvaData 또는 pvaFilter. 영역을 참조하십시오.
AreaIndex	Integer	영역 안의 위치입니다. 데이터 필드의 경우 측정값 인덱스입니다.
DateGrouping	TsgcPivotDateGrouping	pvdNone(기본값), pvdYear, pvdQuarter, pvdMonth, pvdWeek 또는 pvdDay. 날짜 그룹화를 참조하십시오.
DisplayFormat	string	멤버 레이블 또는 데이터 필드 값에 사용하는 FormatDateTime 또는 FormatFloat 서식 문자열.
Aggregation	TsgcPivotAggregation	데이터 필드의 함수이며, 기본값은 pvfSum입니다. 집계를 참조하십시오.
SortBy	TsgcPivotSortBy	pvsLabel(기본값)은 멤버를 레이블순으로, 숫자와 날짜는 값순으로 정렬하고, pvsValue는 멤버의 합계순으로 정렬합니다.
SortOrder	TsgcPivotSortOrder	pvoAscending(기본값) 또는 pvoDescending.
SortMeasure	Integer	값 정렬과 Top N에 사용하는 데이터 필드(AreaIndex 기준). 기본값은 0입니다.
Subtotals	Boolean	멤버의 소계를 표시합니다. 기본값은 True입니다.
Expanded	Boolean	이 수준 멤버의 초기 확장 상태입니다. 기본값은 True입니다.
FilterValues	TStrings	멤버 필터. 필터를 참조하십시오.
FilterExclude	Boolean	FilterValues는 제외할 멤버를 나열합니다. 기본값은 False입니다.
LabelFilter	TsgcPivotLabelFilter	멤버 레이블에 대한 Operator, Value, Value2.
ValueFilter	TsgcPivotValueFilter	멤버 합계에 대한 Operator, Value, Value2, Measure.
TopN, TopNOthers	Integer, Boolean	선택적인 Others 멤버가 있는 Top N. 고급 에디션 기능입니다. Top N을 참조하십시오.

ShowAs, ShowAsAxis, ShowAsBaseItem	TsgcPivotShowAs, TsgcPivotAxisKind, string	값 표시 형식. 고급 에디션 기능입니다. 값 표시 형식을 참조하십시오.

메서드

메서드	반환값	설명
MoveTo(aArea, aAreaIndex)		한 번의 변경 알림으로 필드를 이동합니다. MaxInt는 영역의 끝입니다.
Assign(Source)		Area와 AreaIndex를 포함한 모든 속성을 복사합니다.

TsgcPivotFields

메서드	반환값	설명
Add	TsgcPivotField	필드를 추가합니다.
AreaCount(aArea)	Integer	영역의 필드 수.
FieldAt(aArea, aAreaIndex)	TsgcPivotField	영역의 해당 위치에 있는 필드.
Items[aIndex]	TsgcPivotField	컬렉션 순서의 필드.

BeginUpdate와 EndUpdate 안에서, 그리고 폼을 불러오는 동안에는 AreaIndex 값이 지정한 그대로 유지됩니다. EndUpdate는 각 영역을 AreaIndex 순, 그다음 컬렉션 순으로 다시 번호를 매깁니다.

관련 항목

- TsgcPivotEngine
- TsgcPivotCalculatedField

TsgcPivotCalculatedField

유닛: sgcPivotGrid_Fields
TsgcPivotCalculatedField는 TsgcPivotEngine.CalculatedFields(TsgcPivotCalculatedFields)의 항목입니다. 식으로 소스 행마다 계산되는 Double 열이며, 필드는 FieldName = Name으로 이를 소스 열처럼 사용합니다. 고급 에디션 기능입니다. 구문은 계산 필드를 참조하십시오.

속성

속성	형식	설명
Name	string	필드가 참조하는 이름입니다. 소스 열의 이름과 같을 수 없습니다.
Expression	string	식입니다. 예: [Revenue] - [Cost].

TsgcPivotCalculatedFields

메서드	반환값	설명
Add	TsgcPivotCalculatedField	계산 필드를 추가합니다.
IndexOfName(aName)	Integer	계산 필드의 인덱스이며, 없으면 -1입니다.
Items[aIndex]	TsgcPivotCalculatedField	계산 필드.

관련 항목

- 계산 필드
- TsgcPivotField

TsgcPivotSource_DataSet

유닛: sgcPivotGrid_Source_DataSet
TsgcPivotSource_DataSet은 TDataLink를 통해 TDataSet을 피벗 엔진에 복사하고, 데이터가 바뀌면 다시 불러오는 비시각적 컴포넌트입니다. 빠른 시작 DataSet을 참조하십시오.

속성

속성	형식	설명
DataSet	TDataSet	불러올 데이터셋.
Active	Boolean	데이터를 지금 불러오고 최신 상태로 유지합니다. 기본값은 False입니다.
Columns	TStrings	불러올 열입니다. 비어 있으면 모든 열을 불러옵니다.
LiveUpdates	Boolean	Active인 동안 데이터가 바뀌면 다시 불러옵니다. 기본값은 True입니다.
PivotGrid	TComponent	엔진이 데이터를 받는 피벗 그리드.
Engine	TsgcPivotEngine	Public. 소스가 기록하는 엔진으로, 코드에서 지정한 엔진이며, 없으면 PivotGrid의 엔진입니다.

메서드

메서드	반환값	설명
Load		Active 여부와 관계없이 데이터셋을 지금 불러옵니다.
Refresh		Active이면 다시 불러옵니다.
FieldTypeToDataType(aFieldType)	TsgcPivotDataType	클래스 함수. DB 필드 형식에 해당하는 열 형식.

- 불러오기는 DisableControls를 사용하고 북마크를 복원합니다. BLOB, 이진, 중첩 필드는 건너뜁니다.
- LiveUpdates를 사용하면 데이터셋이 탐색 상태에서의 데이터 변경(Post, Append, Delete, Refresh, First, Last)을 알릴 때 피벗이 다시 불러옵니다. Next, Prior, MoveBy는 다시 불러오지 않습니다.

관련 항목

- TsgcPivotSource_Table
- 빠른 시작 DataSet

TsgcPivotSource_Table

유닛: sgcPivotGrid_Source_Table
TsgcPivotSource_Table 은 모든 IsgcUITable을 피벗 엔진에 복사합니다. 대상은 TsgcGridData, sgcGrid의 TsgcVCLGrid 또는 TsgcFMXGrid, 다른 피벗의 결과, 또는 직접 작성한 클래스입니다. 테이블을 감시하다가 TableVersion이 바뀌면 TableLock 아래에서 다시 불러옵니다. 다시 불러오기는 변경을 알린 스레드에서 실행됩니다.

속성

속성	형식	설명
Table	TComponent	IsgcUITable을 구현하는 컴포넌트.
Active	Boolean	데이터를 지금 불러오고 최신 상태로 유지합니다. 기본값은 False입니다.
Columns	TStrings	불러올 열입니다. 비어 있으면 모든 열을 불러옵니다.
LiveUpdates	Boolean	Active인 동안 데이터가 바뀌면 다시 불러옵니다. 기본값은 True입니다.
PivotGrid	TComponent	엔진이 데이터를 받는 피벗 그리드.
Engine	TsgcPivotEngine	Public. 소스가 기록하는 엔진으로, 코드에서 지정한 엔진이며, 없으면 PivotGrid의 엔진입니다.

메서드

메서드	반환값	설명
SetTable(aTable)		IsgcUITable을 구현하는 모든 개체를 연결합니다. nil을 지정하면 연결이 해제됩니다.
GetTable	IsgcUITable	연결된 테이블.
Load		버전이 바뀌지 않았더라도, Active 여부와 관계없이 테이블을 지금 불러옵니다.
Refresh		Active이면 다시 불러옵니다.

앞선 열과 이름이 같은 열은 건너뜁니다.

예제

관련 항목

- TsgcPivotSource_DataSet
- 내보내기

TsgcPivotGridRenderer

유닛: sgcPivotGrid_Render
TsgcPivotGridRenderer는 모든 sgcUICore 캔버스에 피벗 결과를 그리며, VCL과 FMX 그리드가 사용하는 것도 이것입니다. 단독으로는 헤드리스 PNG와 SVG 출력을 만듭니다. `Create(aOptions: TsgcPivotRenderOptions = nil; aFormatRulesOwner: TPersistent = nil)`로 생성합니다. C++에서는 둘 다 NULL을 전달하십시오. 빠른 시작 헤드리스를 참조하십시오.

속성

속성	형식	설명
옵션	TsgcPivotRenderOptions	RowLayout, RowHeight, HeaderHeight, IndentWidth, MinColumnWidth, MaxColumnWidth, AutoColumnWidths, GridLines, AlternateRows, TotalsBold, ShowExpandButtons, Theme, AccentColor, FontSize이며, 의미와 기본값은 TsgcVCLPivotGrid와 같습니다.
FormatRules	TsgcPivotFormatRules	조건부 서식 규칙.
GrandTotalCaption, TotalCaption	string	총합계와 하단 소계의 캡션입니다. SaveToPNG와 SaveToSVG의 엔진 오버로드는 GrandTotalCaption을 엔진에서 복사합니다.
CurrentResult	TsgcPivotResult	읽기 전용. 마지막 Render의 결과.

메서드

메서드	반환값	설명
Render(aResult, aState, aCanvas, aWidth, aHeight, aScale)		결과의 보이는 부분을 그립니다. aWidth와 aHeight는 장치 픽셀 단위이고 aScale은 DIP당 장치 픽셀 수입니다.
SaveToPNG, SaveToSVG(aResult or aEngine, aState, aFileName or aStream, aWidth = 0, aHeight = 0, aScale = 1)		전체 내용을 이미지로 내보냅니다. 0은 내용 크기입니다. 엔진 오버로드는 그리는 동안 엔진을 잠급니다.
HitTest(aResult, aState, aX, aY)	TsgcPivotHitInfo	한 점 아래에 있는 부분.
ContentWidth, ContentHeight	Integer	전체 내용의 크기(DIP).
FrozenWidth, FrozenHeight	Integer	행 머리글과 열 머리글의 크기.
VisibleRange	Boolean	뷰포트에서 보이는 첫 번째와 마지막 행 항목 및 데이터 열.
ColumnKey(aResult, aDataCol)	string	데이터 열의 키로, 열 너비 재정의에 사용됩니다.

이벤트

이벤트	형식	설명
OnGetCellStyle	TsgcPivotCellStyleEvent	서식 규칙이 적용된 후의 데이터 셀의 색과 굵게 상태.
OnGetCellText	TsgcPivotCellTextEvent	데이터 셀의 텍스트: 서식이 적용된 값이며, 셀이 비어 있으면 결과의 EmptyText입니다.

도우미

루틴	설명
sgcPivotRenderState	빈 TsgcPivotRenderState입니다. 스크롤, 선택, 호버, 열 너비 재정의가 없습니다.
sgcPivotSetColumnWidth(var aState, aKey, aWidth)	ColumnKey가 aKey인 데이터 열의 너비(DIP)를 설정합니다.

관련 항목

- 빠른 시작 헤드리스
- TsgcPivotFormatRule
- 내보내기

TsgcPivotFormatRule

유닛: sgcPivotGrid_Format
TsgcPivotFormatRule은 그리드와 렌더러의 FormatRules(TsgcPivotFormatRules) 항목입니다. 조건에 맞는 값의 셀에 색을 입히는 값 규칙이거나, 두 색 사이의 색 눈금입니다.

속성

속성	형식	설명
Kind	TsgcPivotFormatRuleKind	pvrValue(기본값, 값 규칙) 또는 pvrColorScale.
Measure	Integer	규칙을 적용할 데이터 필드(AreaIndex 기준). -1(기본값)은 모든 측정값에 적용합니다.
Operator, Value, Value2	TsgcPivotFilterOperator, Double, Double	조건이며, 필터와 같은 연산자를 사용합니다. pvcNone은 비어 있지 않은 모든 값에 일치합니다. 색 눈금도 대상 값을 선택하는 데 이를 사용합니다.
BackColor, ForeColor	TsgcUIColor	일치하는 셀의 $AARRGGBB 형식 색입니다. 0(기본값)이면 색을 그대로 둡니다. 색 눈금은 ForeColor만 사용합니다.
Bold	Boolean	일치하는 셀을 굵게 그립니다. 기본값은 False입니다.
MinColor, MaxColor	TsgcUIColor	측정값의 멤버 셀에서 가장 작은 값과 가장 큰 값에 해당하는 눈금의 색입니다. 투명한 MinColor는 MaxColor로 서서히 바뀝니다.
Scope	TsgcPivotFormatScopes	규칙을 적용할 셀: pveCells(기본값, 멤버 셀), pveTotals, pveGrandTotals.
Enabled	Boolean	기본값은 True입니다.
StopOnMatch	Boolean	일치하면 해당 셀에 대한 규칙 평가를 끝냅니다. 기본값은 False입니다.

평가

- 규칙은 컬렉션 순서대로 실행됩니다. 일치하는 사용 중인 모든 규칙이 0이 아닌 색을 적용하므로 나중에 일치한 규칙이 앞의 규칙을 덮어쓰며, Bold는 누적됩니다.
- 빈 셀은 어떤 규칙과도 일치하지 않으며 색이 입혀지지 않습니다.
- 색 눈금의 범위는 멤버 셀에서 구하므로 그 범위 밖의 합계는 양 끝의 색을 받습니다.
- OnGetCellStyle은 규칙 뒤에 실행되며 결과를 바꿀 수 있습니다.

예제

관련 항목

- TsgcVCLPivotGrid
- TsgcPivotGridRenderer

영역

레이아웃의 모든 필드는 네 영역 중 하나에 놓입니다. 영역은 TsgcPivotField의 Area 속성과 AddField의 두 번째 인수로 선택합니다.

Area	의미
pvaRow	멤버가 행 머리글이 됩니다. 여러 행 필드는 중첩되며, 첫 번째가 가장 바깥쪽 수준입니다.
pvaColumn	멤버가 열 머리글이 되며, 같은 방식으로 중첩됩니다.
pvaData	측정값: 모든 셀에서 집계되는 값. 데이터 필드가 여러 개이면 각 열 멤버 아래에 측정값 열이 여러 개 생깁니다.
pvaFilter	보고서 필터. 멤버는 표시되지 않으며, 멤버 필터와 레이블 필터를 통해 소스 행을 제한하기만 합니다.

순서와 이동

- AreaIndex는 한 영역 안의 필드 순서를 정합니다. AddField, AddDataField, Area 설정은 필드를 새 영역의 끝에 놓습니다.
- MoveField(aField, aArea, aIndex)는 한 번의 알림으로 필드를 다른 영역이나 위치로 이동합니다. RemoveField는 필드를 삭제하고 ClearLayout은 모든 필드를 제거합니다.
- Fields.AreaCount(aArea)와 Fields.FieldAt(aArea, aIndex)는 레이아웃을 영역별로 읽습니다.
- 같은 FieldName이 여러 필드에 나타날 수 있습니다. 예를 들어 연도와 월로 그룹화한 Date 열이나, Sum과 Average를 가진 Sales 열이 그렇습니다.
- SortMeasure, ValueFilter.Measure, 서식 규칙에서 사용하는 측정값 인덱스는 데이터 필드의 AreaIndex입니다.

알 수 없는 이름

FieldName이 소스 열도 계산 필드도 아닌 행, 열, 필터 필드는 무시됩니다. 알 수 없는 이름의 데이터 필드는 빈 측정값으로 남으므로 측정값 인덱스가 밀리지 않습니다. 덕분에 데이터를 불러오기 전에 레이아웃을 정의할 수 있습니다.

컴팩트 행과 테이블 형식 행

그리드의 RowLayout은 행 수준을 그리는 방식을 정합니다. pvlCompact(기본값)는 모든 수준을 들여쓰기가 있는 하나의 행 머리글 열에 표시하며, 각 부모는 자기 행을 가집니다. pvlTabular는 각 행 수준에 전용 머리글 열을 줍니다.

예제

레이아웃이 바뀔 때마다 엔진의 OnLayoutChange가 발생합니다. BeginUpdate와 EndUpdate 안에서는 엔진이 EndUpdate에서 한 번만 다시 계산합니다.

관련 항목

- 집계
- TsgcPivotEngine
- TsgcVCLPivotFieldList

집계

데이터 필드는 Aggregation의 함수로 소스 열을 집계합니다. 어떤 함수든 빈 값은 건너뜁니다.

Aggregation	Result
pvfSum	값의 합계. 기본값입니다.
pvfCount	비어 있지 않은 값의 개수로, Excel과 같습니다. 모든 열 형식에서 동작합니다.
pvfAverage	Sum을 Count로 나눈 값.
pvfMin, pvfMax	가장 작은 값과 가장 큰 값.
pvfCountDistinct	비어 있지 않은 서로 다른 값의 개수. 텍스트 열에서는 서로 다른 레이블의 수를 셉니다.
pvfStdDev, pvfVar	표본 표준편차와 표본 분산(`n - 1`). 값이 2개 미만인 셀은 비어 있습니다.

합계는 정확합니다

소계와 총합계는 그 아래의 셀이 아니라 소스 값으로 계산됩니다. 지역의 평균은 그 행들의 평균이지 제품별 평균의 평균이 아니며, 합계의 고유 개수는 각 값을 한 번만 셉니다.

캡션과 서식

- Caption이 없으면 데이터 필드는 *<집계> of <FieldName>* 형식(예: Sum of Sales)으로 표시됩니다. DisplayCaption은 사용 중인 캡션을 반환합니다.
- DisplayFormat은 측정값의 값에 쓰는 FormatFloat 서식 문자열입니다. 예: `#,##0.00`.
- 측정값은 항상 열에 있습니다. 데이터 필드가 여러 개이면 측정값 열이 가장 안쪽 열 수준이 됩니다.

필드 목록은 사용자가 텍스트 열을 값에 놓으면 Count를, 그렇지 않으면 Sum을 추가하며, 값 칩은 모든 집계가 나열된 메뉴를 엽니다.

예제

관련 항목

- 값 표시 형식
- 합계
- TsgcPivotField

날짜 그룹화

날짜 열의 행, 열, 필터 필드는 AddField의 세 번째 인수인 DateGrouping으로 날짜를 그룹화합니다.

DateGrouping	멤버 레이블	같은 필드의 더 큰 수준 아래에서의 레이블
pvdNone	날짜와 시간의 서로 다른 값마다 각각 하나의 멤버가 됩니다.	
pvdYear	2024	
pvdQuarter	2024 Q1	Q1
pvdMonth	2024-01	Jan, 짧은 월 이름
pvdWeek	2024-W05, ISO 8601 주와 그 연도	W05
pvdDay	짧은 날짜	

- 같은 축의 이전 수준이 더 큰 그룹화를 가진 같은 열이면, 마지막 열에 나타난 것처럼 레이블에서 연도가 빠집니다. 한 Date 열의 Year, Quarter, Month는 2024, Q1, Jan이 됩니다.
- DisplayFormat을 지정하면 레이블은 그룹의 시작 시점을 해당 서식 문자열로 FormatDateTime한 것입니다.
- 날짜 멤버는 레이블이 아니라 날짜 순으로 정렬되므로 2024-10은 2024-09 뒤에 옵니다.
- 날짜가 비어 있는 행은 빈 멤버에 속하며, BlankCaption으로 표시되고 마지막에 정렬됩니다.

예제

Expanded는 수준의 초기 상태입니다. 여기서는 모든 분기가 축소된 채로 시작하고, 사용자는 + 버튼으로 분기의 월을 엽니다.

관련 항목

- 영역
- 합계

합계

피벗은 자식이 있는 모든 멤버에 소계를, 각 축에 총합계를 표시합니다.

속성	효과
TsgcPivotEngine.RowGrandTotals	총합계 행을 표시합니다. 기본값은 True입니다.
TsgcPivotEngine.ColumnGrandTotals	총합계 열을 표시합니다. 기본값은 True입니다.
TsgcPivotEngine.SubtotalPosition	pvpTop(기본값): 소계는 멤버 자신의 행에 있고 자식보다 앞에 오며, 컴팩트 레이아웃에서는 부모 행이 이를 가집니다. pvpBottom: 합계 행 또는 열이 자식 뒤에 오며 *<멤버> Total* 레이블이 붙습니다.
TsgcPivotField.Subtotals	False이면 해당 필드 멤버의 소계를 숨깁니다. 기본값은 True입니다.
TsgcPivotEngine.GrandTotalCaption	총합계의 캡션이며, 기본값은 Grand Total입니다.
TsgcPivotEngine.BlankCaption	빈 값의 멤버 레이블이며, 기본값은 (blank)입니다.

- 합계는 정확한 병합으로 소스 행에서 계산되므로 합계의 평균이나 고유 개수도 올바릅니다. 집계를 참조하십시오.
- 값 필터나 Top N 이후에는 합계에 Excel처럼 보이는 상태로 남은 멤버만 포함됩니다.
- 총합계 셀은 총합계 행이나 열이 숨겨져 있어도 항상 계산되므로 값 표시 형식은 여전히 기준 값을 가집니다.
- 렌더러와 그리드는 TotalsBold가 True인 동안 합계를 굵게 그리며, 서식 규칙은 pveTotals와 pveGrandTotals 범위로 이를 선택합니다.

예제

관련 항목

- 필터
- TsgcPivotEngine

필터

네 가지 필터가 피벗이 보여 주는 내용을 줄입니다. 처음 세 가지는 필드의 속성이고, 네 번째는 엔진의 이벤트입니다.

필터	적용 대상	동작 방식
Member filter: FilterValues, FilterExclude	행, 열, 필터 필드	FilterValues는 표시되는 그대로(DateGrouping과 DisplayFormat 적용 후) 멤버 레이블을 나열하며 대소문자를 구분해 비교합니다. 소스 행은 그중 하나를 가져야 하고, FilterExclude를 사용하면 그중 어느 것도 갖지 않아야 합니다. 비어 있으면 모든 행을 유지합니다.
LabelFilter	행, 열, 필터 필드	Operator, Value, Value2를 멤버 레이블과 대소문자를 구분하지 않고 비교합니다. 양쪽이 모두 숫자이면 숫자로 비교합니다.
ValueFilter	행과 열 필드	Operator, Value, Value2를 AreaIndex가 Measure인 데이터 필드에 대한 각 멤버의 합계와 비교합니다. 부모마다 수준별로 실행됩니다. 빈 합계는 pvcNone만 통과합니다.
OnFilterRow	소스 행	할당되어 있는 동안 소스 행마다 발생합니다. 행을 제외하려면 aAccept를 False로 설정하십시오.

멤버 필터와 레이블 필터는 소스 행을 제거하므로 모든 합계가 바뀝니다. 값 필터는 집계 후에 멤버를 제거하며, 합계는 남은 멤버에 대해 다시 계산됩니다.

연산자

연산자	일치 조건
pvcNone	항상: 필터 없음.
pvcEqual, pvcNotEqual	값이 Value와 같거나, 같지 않습니다.
pvcGreater, pvcGreaterEqual, pvcLess, pvcLessEqual	값을 Value와 비교합니다.
pvcBetween, pvcNotBetween	값이 Value와 Value2 사이(양 끝 포함, 순서 무관)에 있거나, 없습니다.
pvcContains, pvcNotContains, pvcBeginsWith, pvcEndsWith	레이블(LabelFilter) 또는 값의 불변 텍스트에 대한 텍스트 검사.

멤버 목록

TsgcPivotEngine.GetMemberLabels(aField, aList)는 FilterValues와 비교되는 레이블을 표시 순서대로 목록에 채웁니다. 필드 목록의 멤버 팝업이 이를 사용합니다. 확인을 누르면 더 짧은 목록, 즉 선택한 멤버 또는 FilterExclude와 함께 선택하지 않은 멤버가 저장되며, 모든 멤버를 선택하면 필터가 해제됩니다.

예제

관련 항목

- Top N
- 합계
- TsgcPivotField

Top N

TopN은 행 필드나 열 필드에서 부모마다 합계가 가장 큰 N개의 멤버를 유지합니다. 고급 기능이며 Community 에디션에는 포함되지 않습니다.

속성	효과
TsgcPivotField.TopN	유지할 멤버의 수입니다. 0(기본값)은 모든 멤버를 유지합니다.
TsgcPivotField.SortMeasure	멤버의 순위를 매기는 합계를 가진 데이터 필드(AreaIndex 기준). 값 정렬의 측정 값이기도 합니다.
TsgcPivotField.TopNOthers	True이면 제외된 멤버의 합계를 가진 Others 멤버를 추가합니다. 항상 마지막에 옵니다.
TsgcPivotEngine.OthersCaption	Others 멤버의 레이블이며, 기본값은 Others입니다.

- Top N은 어떤 멤버가 남을지를 정합니다. SortBy와 SortOrder는 그 순서를 정하므로 pvsValue와 pvoDescending을 지정하면 가장 큰 것부터 나열됩니다.
- 부모의 합계와 총합계에는 유지된 멤버와, 표시되는 경우 Others만 포함됩니다.
- Others의 드릴다운은 그것이 포함하는 모든 멤버의 소스 행을 반환합니다.
- 값 필터는 Others 멤버를 만들지 않습니다.

예제

관련 항목

- 필터
- 에디션

값 표시 형식

ShowAs는 값 자체를 바꾸지 않고 데이터 필드의 값이 표시되는 방식을 바꿉니다. RawValue, 합계, 정렬, 필터, 드릴다운은 계속 실제 숫자를 사용합니다. 고급 기능이며 Community 에디션에는 포함되지 않습니다.

ShowAs	표시되는 값
pvwValue	값 자체. 기본값입니다.
pvwPercentOfRow	값을 해당 행의 합계로 나눈 값(백분율).
pvwPercentOfColumn	값을 해당 열의 합계로 나눈 값(백분율).
pvwPercentOfGrandTotal	값을 총합계로 나눈 값(백분율).
pvwRunningTotal	ShowAsAxis를 따라, 같은 부모를 가진 이전 멤버들의 값을 표시 순서대로 더한 값.
pvwDifferenceFrom	값에서 같은 부모를 가진 기준 멤버의 값을 뺀 값. ShowAsBaseItem이 비어 있으면 이전 멤버, 아니면 레이블이 ShowAsBaseItem인 멤버(대소문자 구분 없음).
pvwRank	같은 부모를 가진 멤버 중의 순위로, 가장 큰 값이 1입니다. 동점은 같은 순위를 공유하고 다음 순위는 건너뜁니다: 1, 2, 2, 4.

- 백분율 모드는 25퍼센트를 25로 표시하며 `0.00%` 형식으로 서식을 지정합니다. 빈 값, 기준이 없는 경우, 기준이 0인 경우는 빈 셀이 됩니다.
- ShowAsAxis(기본값 pvxColumns)는 누계, 차이, 순위를 계산하는 축입니다.
- pvwDifferenceFrom에서 그룹의 첫 번째 멤버는 비교할 대상이 없으며, 기준 멤버 자신도 Excel처럼 비어 있습니다. ShowAsAxis의 총합계에는 형제가 없으므로 마지막 세 모드에서는 비어 있습니다.
- 같은 소스 열을 한 번은 값으로, 한 번은 ShowAs를 지정해서 두 번 추가하면 둘 다 표시할 수 있습니다.

예제

관련 항목

- 집계
- 에디션

계산 필드

계산 필드는 식으로 모든 소스 행마다 계산되는 숫자 열입니다. 필드는 FieldName을 통해 소스 열과 똑같이 이를 사용합니다. 고급 기능이며 Community 에디션에는 포함되지 않습니다.

행 수준

식은 소스 행마다 평가된 다음 집계되므로, `Profit = [Revenue] - [Cost]`의 합계는 이익의 합계입니다. 이는 Excel이 합계에 대해 계산하는 수식과 다릅니다. 행별 마진의 평균은 합계의 마진이 아닙니다. 합계의 비율이 필요하면 두 측정값을 모두 표시하고 OnGetCellText에서 비율을 계산하거나 값 표시 형식을 사용하십시오.

구문

요소	구문
숫자	소수점은 점을 쓰는 십진수이며 지수는 선택입니다: 12, 0.5, 1E3.
필드	[Field Name] 또는 영문자, 숫자, 밑줄로 된 이름으로, 소스 열과 다른 계산 필드에서 대소문자를 구분하지 않고 찾습니다.
연산자	우선순위가 낮은 것부터: `OR`, `AND`, `NOT`, `= <> < <= > >=`, `+ -`, `* /`, 단항 `-` 와 `+`, `^` (거듭제곱, 우결합). 괄호로 묶습니다.
함수	IF(c, a, b), ABS(x), ROUND(x, n), MIN(x, ...), MAX(x, ...), SQRT(x), ISNULL(x, v), YEAR(d), MONTH(d). 대소문자를 구분하지 않습니다.

- 피연산자가 비어 있으면 결과도 비어 있습니다. 단, x가 비어 있을 때 v를 반환하는 ISNULL과, 조건과 선택된 분기만 보는 IF는 예외입니다.
- 0으로 나누기, 음수의 제곱근, 오버플로는 빈 결과를 줍니다.
- 비교 연산자와 논리 연산자는 1 또는 0을 반환합니다. 0이 아닌 모든 값은 참입니다.
- ROUND는 0에서 먼 쪽으로 반올림하며, n은 -15에서 15까지입니다.

오류

구문 오류나 계산 필드 사이의 순환 참조는 다음 재계산에서 EsgcPivotError를 발생시키며, 메시지에 계산 필드의 이름이 들어 있습니다. 불러오지 않은 필드 이름은 오류가 아닙니다. 해당 열이 도착할 때까지 계산 필드는 비어 있으므로 데이터보다 먼저 레이아웃을 정의할 수 있습니다. Expression이 비어 있어도 값이 비어 있습니다.

예제

관련 항목

- TsgcPivotCalculatedField
- 집계

레이아웃 저장

피벗의 레이아웃, 즉 필드, 필터, 합계, 확장된 멤버는 JSON으로 저장하고 불러옵니다. 고급 기능이며 Community 에디션에는 포함되지 않습니다.

메서드	설명
SaveLayoutToString: string	JSON 텍스트로 된 레이아웃.
LoadLayoutFromString(aJSON)	한 번의 재계산으로 레이아웃을 바꿉니다.
SaveLayoutToFile(aFileName)	JSON을 UTF-8 파일로 기록합니다.
LoadLayoutFromFile(aFileName)	UTF-8 파일을 읽어 레이아웃을 바꿉니다.

TsgcPivotEngine과 두 그리드에 이 메서드들이 있습니다. 그리드 버전은 사용자가 바꾼 열 너비도 저장합니다.

형식

문서는 하나의 개체이며, `v`(형식 버전, 1), `rowGrandTotals`, `columnGrandTotals`, `subtotalPosition`, `calculatedFields`(이름과 식), `fields`(필드마다 하나의 개체로, 영역별로 묶이며 TsgcPivotField의 모든 속성을 가짐), `expanded`(수준과, 행과 열의 멤버 경로의 확장 상태), 그리고 그리드에서는 `columnWidths`를 포함합니다. 열거형은 접두사를 뗀 멤버 이름의 첫 글자를 소문자로 바꾸어 기록합니다. pvfCountDistinct는 `countDistinct`가 됩니다.

불러오기 규칙

- 불러오기는 관대합니다. 알 수 없는 멤버는 무시하고, 없는 멤버나 형식이 틀린 값은 기본값을 유지합니다.
- 엔진에 데이터가 있으면, 이름이 소스 열도 계산 필드도 아닌 필드는 건너뜁니다.
- 데이터보다 먼저 불러온 레이아웃은 필드가 데이터와 일치할 때까지 확장 상태를 유지합니다.
- 잘못된 JSON이거나 루트가 개체가 아니면 EsgcPivotError가 발생하며 레이아웃은 바뀌지 않습니다.

예제

관련 항목

- 영역
- 에디션

내보내기

엔진과 두 그리드는 합계와 확장 상태를 포함한 현재 뷰를 내보냅니다. 모든 메서드에 파일 오버로드와 스트림 오버로드가 있으며, 텍스트는 BOM 없는 UTF-8로 기록됩니다.

메서드	출력
SaveToCSV	CRLF 레코드를 사용하는 RFC 4180 CSV입니다. 행 수준마다 열 하나, 열 수준마다 머리글 행 하나와 측정값 행이 있으며, 머리글 레이블은 걸쳐 있는 모든 셀에서 반복됩니다. 값은 표시 형식이 아니라 1250.5 같은 불변 형식의 숫자로 기록되며, 빈 셀은 빈 필드입니다.
SaveToHTML	표 하나가 있는 완전한 HTML 문서. 병합된 머리글에는 colspan과 rowspan이 사용되고, 값은 서식이 적용된 셀 텍스트이며 빈 셀에는 엔진의 EmptyText가 들어갑니다.
SaveToJSON	구조화된 스냅샷: 수준의 캡션, 서식이 있는 측정값, 그리고 종류(member, total, grandTotal), 수준, 레이블, 경로, 확장 상태를 가진 열 항목과 행 항목. 행 항목에는 값 표시 형식 적용 후의 값도 있으며, 비어 있으면 null입니다.
SaveToPNG, SaveToSVG	그리드와 렌더러만 해당: 전체 그리드를 이미지로 내보냅니다. 너비와 높이는 DIP 단위이며 0은 내용 크기이고 최대 16384픽셀입니다.

테이블로

TsgcPivotEngine과 두 그리드는 ResultTable을 통해 IsgcUITable을 구현하므로, 결과를 sgcCharts의 차트 데이터 소스나 다른 피벗 같은 모든 sgcUI 소비자에게 전달할 수 있습니다. 행은 보이는 행 리프이며, 축소된 멤버는 포함되고 합계 행은 제외됩니다. 열은 행 수준마다 하나의 텍스트 열, 그다음 보이는 열 리프와 측정값마다 하나의 열이며, 이름은 *<열 경로>* | *<측정값 캡션>*입니다. 열 필드가 없으면 측정값 캡션만 사용합니다. 값이 빈 셀은 TableText로 엔진의 EmptyText를, TableValue로 Null을 가집니다. TableVersion은 새 스냅샷마다 바뀝니다. 차트 연동을 참조하십시오.

예제

관련 항목

- 빠른 시작 헤드리스
- TsgcPivotGridRenderer

확장과 축소

행이나 열에서 자식이 있는 모든 멤버는 확장하여 자식을 보여 주거나 한 행 또는 한 열로 축소할 수 있습니다. 축소된 멤버는 자리를 유지하며 그 아래 모든 것의 합계를 표시합니다.

그리드에서

- 행 머리글과 열 머리글의 + 버튼과 - 버튼으로 멤버를 전환합니다. ShowExpandButtons로 숨길 수 있습니다.
- 행 머리글을 더블 클릭하면 해당 멤버가 전환됩니다.
- 행 머리글에 포커스가 있을 때 Right는 확장하고, Left는 축소하며, Enter는 전환합니다. + 키와 - 키는 어느 열에서든 선택한 행의 멤버를 확장하고 축소합니다. 키보드와 접근성을 참조하십시오.
- OnExpandChange는 사용자가 멤버를 확장하거나 축소한 후에 aAxis와 aItem과 함께 발생합니다. 아래의 메서드에서는 발생하지 않습니다.

코드에서

멤버	효과
TsgcPivotEngine.ExpandAll(aAxis), CollapseAll(aAxis)	pvxRows 또는 pvxColumns의 모든 멤버.
TsgcPivotEngine.ExpandLevel(aAxis, aLevel), CollapseLevel(aAxis, aLevel)	한 수준의 모든 멤버이며, 0이 가장 바깥쪽 수준입니다.
TsgcPivotEngine.SetExpanded(aAxis, aItem, aValue), ToggleExpanded(aAxis, aItem)	현재 스냅샷의 항목 하나를 Result.Item의 인덱스로 지정합니다. 자식이 없는 항목은 그대로 둡니다.
TsgcPivotField.Expanded	해당 필드 수준의 멤버의 초기 상태입니다. 기본값은 True입니다. 날짜 그룹화를 참조하십시오.
TsgcPivotEngine.GetExpandState(aAxis, aState), SetExpandState(aAxis, aState)	축의 명시적 확장 상태를 문자열 목록으로 다룹니다. 예를 들어 다른 엔진에 복사할 때 사용합니다. 고급 에디션 기능입니다.
TsgcPivotItem.Expandable, Expanded	Result.Item에서 읽습니다. 항목에 자식이 있고 그것이 표시되어 있는지 여부입니다.

상태가 유지되는 방식

- 상태는 항목 인덱스가 아니라 North / Spain 같은 멤버 경로별로 저장됩니다. 재계산, 데이터 다시 불러오기, 필터, 새 정렬 순서를 거쳐도 유지되며, 사라졌다가 돌아오는 멤버도 상태를 유지합니다.
- 축의 필드가 바뀌면(다른 필드, 다른 순서, 다른 날짜 그룹화) 그 축의 상태는 필드의 Expanded 속성에서 다시 시작합니다.
- 확장이나 축소는 다시 집계하지 않고 계산된 데이터에서 새 스냅샷을 만들기 때문에 큰 결과에서도 빠릅니다. 아키텍처의 성능 표를 참조하십시오.
- 내보내기와 ResultTable은 현재 상태를 따릅니다. 축소된 멤버는 합계를 가진 한 행이 됩니다.
- 레이아웃 저장은 확장 상태를 레이아웃과 함께 저장하고 불러옵니다.

예제

SetExpanded는 변경으로 새 스냅샷이 게시되므로 Unlock 뒤에 호출합니다.

관련 항목

- 합계
- 레이아웃 저장
- TsgcPivotEngine

정렬

모든 행 필드와 열 필드는 부모마다 멤버를 정렬하므로 각 지역의 국가가 따로 정렬됩니다. TsgcPivotField의 세 가지 속성이 이를 제어합니다.

속성	효과
SortBy	pvsLabel(기본값)은 멤버 레이블로 정렬합니다. pvsValue는 각 멤버의 합계로 정렬합니다.
SortOrder	pvoAscending(기본값) 또는 pvoDescending.
SortMeasure	값 정렬이 합계를 비교하는 데이터 필드(AreaIndex 기준). 기본값은 0입니다. Top N도 이를 사용합니다.

규칙

- 레이블 정렬은 텍스트 멤버를 로캘 규칙에 따라 대소문자를 구분하지 않고 비교하고, 숫자와 날짜 멤버는 값으로 비교하므로 9가 10보다 앞에, 2024-10이 2024-09 뒤에 옵니다.
- 값 정렬은 SortMeasure 측정값에 대한 각 멤버의 합계를 비교합니다. 행 필드에서는 모든 열에 걸친 합계, 열 필드에서는 모든 행에 걸친 합계입니다. 합계가 비어 있는 멤버는 마지막에 오고, 합계가 같은 멤버는 레이블 순서를 유지합니다.
- SortMeasure가 데이터 필드의 AreaIndex가 아니면 멤버는 레이블로 정렬됩니다.
- 빈 멤버와 Top N의 Others 멤버는 두 순서 모두에서 끝에 남으며, Others는 빈 멤버 뒤에 옵니다.
- Top N은 먼저 SortMeasure의 합계가 가장 큰 멤버를 고르고, 그다음 SortBy와 SortOrder가 순서를 정합니다.
- SortBy, SortOrder, SortMeasure를 바꾸면 피벗이 다시 계산됩니다. BeginUpdate와 EndUpdate 브래킷마다 한 번입니다.
- 그리드는 머리글을 클릭해도 정렬하지 않습니다. 사용자가 정렬하게 하려면, 예를 들어 히트한 Part가 pvhColumnHeader일 때의 OnCellClick에서 코드로 속성을 설정하십시오.

예제

관련 항목

- Top N
- 날짜 그룹화
- TsgcPivotField

드릴다운

드릴다운은 셀 뒤에 있는 소스 행, 즉 그 값으로 집계된 행을 읽습니다. 엔진은 이를 행 번호 또는 테이블로 반환합니다.

메서드	반환값	설명
TsgcPivotEngine.GetSourceRows(aRowItem, aDataCol, var aRows)	Integer	소스 행의 번호를 행 순서대로 반환하며 그 개수도 반환합니다. aRowItem은 Result의 행 항목이고 aDataCol은 데이터 열입니다. aDataCol이 -1이면 행 항목의 모든 열을 대상으로 합니다.
TsgcPivotEngine.DrillDown(aRowItem, aDataCol)	TsgcPivotSourceView	같은 행을 레이아웃에 없는 열까지 포함해 저장소의 모든 열을 가진 IsgcUITable로 반환합니다. 호출자가 해제합니다.
TsgcPivotEngine.SourceValue(aRow, aCol)	Variant	소스 행의 값이며, 비어 있으면 Null입니다. SourceColumnName은 열 이름을 줍니다.

대상 행

- 멤버 셀은 그 행 멤버와 열 멤버의 행을 반환합니다. 소계나 총합계는 그 아래의 모든 행을 반환하고, Top N의 Others 멤버는 그것이 포함하는 모든 멤버의 행을 반환합니다.
- 멤버 필터, 레이블 필터, 보고서 필터, OnFilterRow에 의해 제외된 행은 반환되지 않습니다.
- 열 항목의 모든 측정값은 같은 행을 가지므로 aDataCol의 측정값은 결과를 바꾸지 않습니다.

그리드에서

데이터 셀에서 Enter를 누르거나 더블 클릭하면 aRowItem, aDataCol, var aHandled와 함께 OnDrillDown이 발생합니다. 그리드에는 자체 동작이 없으므로 애플리케이션이 대화 상자, 그리드, 보고서에 행을 표시합니다. DataSetPivot 데모는 셀 뒤에 있는 주문을 엽니다.

수명

- TsgcPivotSourceView는 엔진의 저장소를 실시간으로 읽습니다. 소스 데이터가 바뀌면 행이 없어지며, 엔진보다 먼저 해제해야 합니다.
- 항목 인덱스는 현재 스냅샷에 속합니다. 그리드 이벤트에서 가져오거나, Lock과 Unlock 사이에서 Result를 읽으십시오.

예제

관련 항목

- TsgcPivotEngine
- TsgcVCLPivotGrid
- 필터

조건부 서식

그리드와 렌더러의 FormatRules 컬렉션은 값에 따라 데이터 셀에 색을 입힙니다. 그리드와 이미지는 같은 렌더러로 그려지므로, 같은 규칙이 화면과 PNG, SVG 이미지에 모두 적용됩니다. 디자인 타임에는 개체 인스펙터의 컬렉션 편집기로 규칙을 편집합니다.

Kind	동작
pvrValue	값 규칙. 값이 Operator, Value, Value2(필터의 연산자)와 일치하면 셀에 BackColor, ForeColor, Bold가 적용됩니다. pvcNone은 비어 있지 않은 모든 값과 일치합니다.
pvrColorScale	색 눈금. 셀은 측정값의 멤버 셀에서 가장 작은 값과 가장 큰 값 사이의 위치에 따라 MinColor와 MaxColor 사이의 색을 받습니다. Operator, Value, Value2는 색을 입힐 값을 선택합니다.

대상 셀

- Measure는 AreaIndex로 데이터 필드를 선택하며, -1은 모든 측정값에 규칙을 적용합니다.
- Scope는 셀의 종류를 선택합니다: pveCells(기본값, 멤버 셀), pveTotals, pveGrandTotals.
- 빈 셀은 어떤 규칙과도 일치하지 않으며 색이 입혀지지 않습니다. 엔진의 EmptyText가 표시됩니다.

규칙의 순서

- 규칙은 컬렉션 순서대로 실행됩니다. 일치하는 사용 중인 모든 규칙이 0이 아닌 색을 적용하므로 나중에 일치한 규칙이 앞의 규칙을 덮어쓰며, Bold는 누적됩니다.
- StopOnMatch는 해당 셀의 평가를 끝내므로 처음 일치한 규칙이 우선합니다. 예제에서는 세 개의 구간에 이를 사용합니다.
- OnGetCellStyle은 규칙 뒤에 var aBack, aFore, aBold와 함께 실행되며 이 중 무엇이든 바꿀 수 있습니다. 그 안에서 Renderer.CurrentResult는 그리고 있는 스냅샷입니다.

색

색은 $AARRGGBB 형식의 TsgcUIColor 값이며, $FF2E7D32는 불투명한 녹색입니다. 0은 테마의 색을 그대로 둡니다. 다크 테마에서는 텍스트가 읽기 쉬운 색을 고르고, 의미가 색에만 의존하지 않도록 색과 Bold를 함께 사용하십시오.

예제

관련 항목

- TsgcPivotFormatRule
- TsgcPivotGridRenderer
- 키보드와 접근성

차트 연동

sgcPivotGrid는 sgcUICore의 테이블 인터페이스인 IsgcUITable(유닛 sgcUICore_Table)을 통해 다른 sgcUI 제품과 데이터를 주고받습니다. sgcPivotGrid의 패키지는 sgcCharts나 sgcGrid에 의존하지 않으며, 애플리케이션이 이들을 연결합니다.

피벗에서 sgcCharts로

TsgcPivotEngine 과 두 그리드는 ResultTable 을 통해 IsgcUITable 을 구현합니다. sgcCharts 의 TsgcChartDataSource_Table의 Table 속성을 피벗 그리드로 설정하거나, 그리드가 없는 엔진으로 SetTable을 호출하십시오.

테이블 부분	내용
행	보이는 행 리프: 축소된 멤버는 포함되고, 소계와 총합계 행은 제외됩니다.
텍스트 열	행 수준마다 하나이며, 수준의 캡션이 이름이 되고 그 수준의 멤버 레이블이 값이 됩니다.
값 열	보이는 열 리프와 측정값마다 하나이며, 이름은 <*열 경로*> | <*측정값 캡션*>입니다. 예: `Online | Sales` . 열 필드가 없으면 측정값 캡션만 사용합니다.
TableValue	텍스트 열은 레이블이고, 값 열은 표시되는 값(값 표시 형식 적용 후)이 Double로 반환되며, 셀이 비어 있으면 Null입니다.
TableText	레이블 또는 서식이 적용된 셀 텍스트. 빈 셀은 엔진의 EmptyText를 줍니다.

- 열은 데이터 소스에서 매핑합니다. YColumn은 값 열을, LabelColumn은 원형, 깔때기, 트리맵의 행 수준(여러 수준은 ;로 구분)을, XColumn은 XY 계열의 X 값을 지정합니다.
- 데이터 필드에 Sales 같은 Caption을 지정하면 집계가 바뀌어도 열 이름이 바뀌지 않습니다.
- 새 스냅샷마다 TableVersion이 바뀌고 리스너에 알리므로, 재계산, 확장, 축소 후에 차트가 다시 불러옵니다.
- GridToPivotToChart 데모는 채널마다 데이터 소스 하나를 사용한 그룹 막대를 그리고, OnCalculated에서 행 레이블로 X축의 범주를 채웁니다.

sgcGrid에서 피벗으로

TsgcPivotSource_Table은 sgcGrid의 TsgcVCLGrid, TsgcFMXGrid, TsgcGridData 같은 모든 IsgcUITable을 엔진에 불러옵니다. TableVersion이 바뀌면 다시 불러오므로, 그리드에서의 편집이 피벗에 반영되고 이어서 차트에 반영됩니다. 그리드의 많은 변경은 그리드 데이터의 BeginUpdate와 EndUpdate로 묶으면 한 번만 다시 불러옵니다.

피벗에서 다른 피벗으로

한 피벗의 결과도 테이블이므로 TsgcPivotSource_Table로 두 번째 피벗에 불러올 수 있습니다.

코드에서 테이블 읽기

테이블은 TableLock과 TableUnlock 사이에서 읽으십시오. 참조 카운트는 비활성화되어 있고 테이블은 엔진이 소유하므로, 엔진을 해제하기 전에 모든 인터페이스 참조를 버리십시오.

예제

관련 항목

- 내보내기
- TsgcPivotSource_Table
- 데모

키보드와 접근성

두 그리드 모두 키보드만으로 사용할 수 있습니다. 같은 키가 TsgcVCLPivotGrid와 TsgcFMXPivotGrid에서 동작하며, macOS의 Cmd는 Ctrl처럼 동작합니다.

키	동작
Up, Down	이전 행 또는 다음 행.
Page Up, Page Down	한 페이지 위 또는 아래.
Home, End	행의 첫 번째 또는 마지막 데이터 열. Ctrl과 함께 누르면 그리드의 첫 번째 또는 마지막 셀.
Left	이전 데이터 열. 첫 번째 데이터 열에서는 행 머리글로 이동하고, 행 머리글에서는 멤버를 축소합니다.
Right	행 머리글에서는 축소된 멤버를 확장하고, 그렇지 않으면 첫 번째 데이터 열로 이동합니다. 데이터에서는 다음 열로 이동합니다.
+, -	숫자 키패드 또는 주 키보드: 어느 열에서든 선택한 행의 멤버를 확장하거나 축소합니다.
Enter	데이터 셀에서는 OnDrillDown을 발생시킵니다. 행 머리글에서는 멤버를 전환합니다.

- 아무것도 선택되어 있지 않으면 첫 번째 탐색 키가 첫 번째 셀을 선택합니다.
- TsgcVCLPivotGrid는 TabStop이 기본적으로 True이고 TsgcFMXPivotGrid는 CanFocus가 True입니다. 클릭하면 그리드에 포커스가 옵니다.
- 코드에서는 SelectedRow, SelectedCol, ScrollToCell로 선택을 이동할 수 있으며, OnSelectionChange가 모든 변경을 알립니다.
- 키는 행 멤버를 확장하고 축소합니다. 열 멤버는 열 머리글의 버튼이나 코드로 확장합니다. 확장과 축소를 참조하십시오.
- 필드 목록은 마우스나 터치로 사용합니다. Escape는 끌기를 취소하며, TsgcFMXPivotFieldList는 Up, Down, Page Up, Page Down, Home, End로도 목록을 스크롤합니다. 키보드 사용자에게는 AddField와 MoveField를 사용한 코드나 저장된 레이아웃으로 레이아웃을 제공하십시오.

접근성

- 그리드와 필드 목록은 sgcUICore가 그리며, UI Automation이나 MSAA를 통해 스크린 리더에 내용을 게시하지 않습니다. 스크린 리더 사용자에게는 데이터를 HTML로 제공하십시오. SaveToHTML은 머리글 셀, colspan, rowspan이 있는 실제 표를 기록합니다. ResultTable로 표준 컨트롤을 채울 수도 있습니다.
- 크기는 DIP 단위이며 모니터의 DPI를 따릅니다. FontSize, RowHeight, HeaderHeight로 텍스트와 행을 더 크게 만들 수 있습니다.
- Theme은 라이트 테마 또는 다크 테마를 선택하고, AccentColor는 테마의 강조 색을 대체합니다.
- 조건부 서식은 색에 Bold를 더할 수 있어서 값이 색에만 의존하지 않게 할 수 있습니다.

예제

관련 항목

- TsgcVCLPivotGrid
- 드릴다운

성능

엔진은 대용량 데이터를 위해 만들어졌습니다. 형식이 지정된 열 저장소, 재계산마다 소스 행을 한 번만 훑는 처리, 가상 그리기를 사용합니다. QA 성능 그룹은 아키텍처에 나온 예산에 맞춰 주요 시나리오를 측정합니다. 다음 지침을 따르면 애플리케이션이 그 범위 안에 머뭅니다.

불러오기

- BeginLoad와 EndLoad 안에서 불러오십시오. 엔진은 가장 바깥쪽 EndLoad에서 한 번만 다시 계산합니다. 브래킷 밖에서는 AppendRow와 DefineColumn마다 즉시 다시 계산합니다.
- 열은 DefineColumn(aName, aType)으로 정의하십시오. 형식을 생략하면 비어 있지 않은 첫 번째 값이 형식을 결정하며, 정수 열은 소수 값이 들어오면 실수 열이 됩니다.
- TsgcPivotSource_DataSet은 LiveUpdates가 True인 동안 Post할 때마다 데이터셋 전체를 다시 불러옵니다. 편집을 일괄로 할 때는 LiveUpdates 또는 Active를 False로 하고 마지막에 Load를 호출하십시오.
- TsgcPivotSource_Table은 TableVersion이 바뀔 때만 다시 불러옵니다. 그리드의 변경은 그리드의 BeginUpdate와 EndUpdate로 묶으십시오.

레이아웃과 뷰

- 레이아웃은 BeginUpdate와 EndUpdate 안에서 바꾸면 재계산이 한 번만 일어납니다. 더 긴 일괄 작업에서는 AutoRecalculate를 False로 하고 마지막에 Recalculate를 호출하십시오.
- 필드 목록의 DeferUpdate는 사용자의 변경을 ApplyChanges까지 보류합니다.
- 확장, 축소, RowGrandTotals, ColumnGrandTotals, SubtotalPosition, RowLayout, EmptyText는 뷰만 바꿉니다. 엔진은 다시 집계하지 않고 계산된 데이터에서 새 스냅샷을 만듭니다.

결과의 크기

- 셀의 수는 행의 멤버 수와 열의 멤버 수의 곱으로 늘어납니다. 주문 번호나 일 단위 날짜처럼 멤버가 많은 필드를 두 축에 모두 두면 그 수가 곱해집니다. MaxCells(기본값 5,000,000)보다 많은 셀이 필요한 레이아웃은 EsgcPivotError를 발생시킵니다. 날짜를 그룹화하거나, 필드를 필터로 옮기거나, Top N을 사용하십시오.
- 데이터 필드마다 셀당 값이 하나씩 추가됩니다. CountDistinct는 셀마다 고유 값을 유지하고, StdDev와 Var는 Sum보다 많은 상태를 유지하므로 큰 레이아웃에서는 메모리와 시간이 더 듭니다.

이벤트

- OnFilterRow는 재계산할 때마다 모든 소스 행에 대해 실행됩니다. 필터 예제처럼 열 인덱스는 이벤트 밖에서 한 번만 찾으십시오. 멤버 필터와 레이블 필터는 멤버마다 한 번만 결정되므로 비용이 더 적습니다.
- OnGetCellText와 OnGetCellStyle은 매 프레임마다 보이는 모든 셀에 대해 실행됩니다. 짧게 유지하십시오.
- Result의 잠금은 읽는 동안에만 잡고 있으십시오.

예제

관련 항목

- 아키텍처
- TsgcPivotEngine
- 문제 해결

TsgcPivotGridAIQuery

Unit: sgcPivotGridWS_AI
Package: sgcPivotGridWS (Delphi)와 sgcPivotGridWSC (C++Builder), sgcPivotGrid의 sgcWebSockets 브리지입니다. Registered 에디션, Windows 전용 (Win32 및 Win64)입니다.
TsgcPivotGridAIQuery는 *top 3 countries by sales* 같은 자연어 질문을 피벗 레이아웃으로 바꾸는 비시각 컴포넌트입니다. 질문과 소스 필드 카탈로그를 sgcWebSockets의 **TsgcAIChat**을 통해 AI 제공자에게 보내고, JSON 응답을 엔진의 필드에 대해 검증한 뒤 한 번의 재계산으로 적용합니다. 소스 행은 전송되지 않습니다. 전송되는 내용, 응답 검증 방식, 비동기 모드는 피벗에 질문하기을 참조하십시오.
팔레트의 **SGC PivotGrid** 페이지에서 TsgcAIChat과 TsgcPivotGridAIQuery를 놓고, Chat과 PivotGrid를 설정한 다음 Ask를 호출합니다. 제공자, 모델, API 키는 채팅의 속성입니다: Provider, ChatOptions.Model, ChatOptions.ApiKey.

속성

속성	형식	설명
Active	Boolean	False이면 Ask와 AskAsync는 False를 반환하고 아무 일도 하지 않습니다. 기본값은 True입니다.
Chat	TsgcAI_Chat	제공자와 통신하는 채팅입니다. TsgcAIChat 또는 TsgcAI_Chat의 모든 하위 클래스입니다. 채팅이 없으면 Ask는 OnError를 발생시킵니다.
PivotGrid	TComponent	레이아웃을 변경할 피벗 그리드입니다. TsgcVCLPivotGrid, TsgcFMXPivotGrid 또는 IsgcPivotEngineHost를 구현하는 모든 컴포넌트입니다.
Engine	TsgcPivotEngine	Public. 그리드 없이 사용하는 엔진으로, 예를 들어 서버에서 씁니다. PivotGrid보다 우선합니다.
SendSampleValues	Boolean	제공자가 *online sales* 같은 질문의 멤버를 맞출 수 있도록, 텍스트, 정수, 날짜, 불리언 필드마다 몇 개의 값을 카탈로그와 함께 보냅니다. 기본값은 True입니다.
MaxSampleValues	Integer	필드당 샘플 값의 수로, 0..5로 제한됩니다. 기본값은 3입니다.
Language	string	설명의 언어로, 예를 들어 `Spanish` 입니다. 비워 두면(기본값) 질문의 언어로 작성됩니다.
ReplaceLayout	Boolean	True(기본값)이면 새 레이아웃을 요청합니다. False이면 현재 레이아웃도 함께 보내므로 *add the channel to the columns* 같은 질문이 그것을 변경합니다. 두 경우 모두 응답은 완전한 새 레이아웃입니다.
Busy	Boolean	Public, 읽기 전용. AskAsync 요청이 실행되는 동안 True입니다.
LastLayout	TsgcPivotAILayout	Public. 마지막 응답이 적용한 레이아웃입니다. Undo 후에는 비어 있습니다.
LastExplanation	string	Public. 마지막으로 적용된 응답의 설명으로, 상태 표시줄용 짧은 문장입니다.

이벤트

이벤트	형식	설명

OnBeforeAsk	TsgcPivotAIBeforeAskEvent	`(Sender; var aPrompt: string; var aAllow: Boolean)`. 요청 전에 메인 스레드에서 사용자 메시지와 함께 발생합니다. 자신의 지침을 추가하려면 aPrompt를 변경하고, 질문을 조용히 취소하려면 aAllow를 False로 설정합니다.
OnLayout	TsgcPivotAILayoutEvent	`(Sender; aLayout: TsgcPivotAILayout; var aAccept: Boolean)`. 검증된 레이아웃이 적용되기 전에 발생합니다. aLayout을 확인하거나 편집하고, 현재 레이아웃을 유지하려면 aAccept를 False로 설정합니다.
OnApplied	TNotifyEvent	레이아웃이 적용된 후에 발생합니다. LastLayout과 LastExplanation에 응답이 들어 있습니다.
OnError	TsgcPivotAIErrorEvent	`(Sender; const aError: string)`. 아무것도 적용되지 않았을 때 발생합니다: 채팅 없음, 엔진 없음, 빈 질문, 아직 실행 중인 요청, 제공자 오류, JSON 객체가 없는 응답, 또는 사용할 수 있는 행, 열, 데이터 필드가 없는 응답. 레이아웃은 그대로 유지됩니다.

메서드

메서드	반환값	설명
Ask(aQuestion)	Boolean	제공자에게 질문하고 응답을 검증한 뒤 적용합니다. 동기식이며 메인 스레드에서만 사용합니다. 레이아웃이 적용되면 True입니다.
AskAsync(aQuestion)	Boolean	같은 동작을, 요청을 작업자 스레드에서 수행합니다. 응답은 메인 스레드에서 검증 및 적용된 다음 OnApplied 또는 OnError가 발생합니다. Busy이거나 요청이 시작되지 않았으면 False입니다.
Undo		마지막으로 적용된 응답 이전의 레이아웃을 복원합니다. 한 단계만 유지됩니다.
BuildPrompt(aQuestion)	string	Ask가 aQuestion에 대해 보내는 사용자 메시지: 질문, 언어 줄, 필드 카탈로그입니다. 엔진이 없으면 비어 있습니다. 애플리케이션을 벗어나는 내용을 사용자에게 보여 줄 때 사용합니다.
ValidateLayoutJSON(aJSON, aLayout)	Boolean	응답 또는 저장된 레이아웃을 추출, 구문 분석하고 엔진의 필드에 대해 검증하여 aLayout에 담습니다. JSON 객체를 찾지 못했거나 행, 열, 데이터 필드가 남지 않으면 aLayout을 비운 채 False를 반환합니다.
ResolveEngine	TsgcPivotEngine	할당되어 있으면 Engine, 아니면 PivotGrid의 엔진, 그것도 없으면 nil입니다.

하위 클래스는 세 개의 protected virtual 메서드를 재정의할 수 있습니다: 제공자와 통신하는 유일한 메서드인 **DoAskProvider** (AskYourPivot 데모가 자체 테스트에서 하듯이 테스트가 여기서 미리 준비한 응답을 반환할 수 있습니다), 필드 카탈로그를 만드는 **DoBuildFields** (제외된 필드는 전송도 수락도 되지 않습니다), 엔진의 레이아웃을 교체하는 **DoApply**입니다.

TsgcPivotAILayout

검증 후 응답의 레이아웃입니다. OnLayout이 이를 받고, LastLayout은 적용된 것을 보관합니다.

멤버	설명

Rows, Columns	순서대로 축 필드: FieldName과 Grouping (TsgcPivotDateGrouping).
Data	순서대로 데이터 필드: FieldName과 Aggregation (TsgcPivotAggregation). 항목의 인덱스는 필터, 정렬, Top N이 참조하는 측정값 인덱스입니다.
Filters	종류는 `pafMembers` (Members, Exclude), `pafLabel` (FilterOperator, Value, Value2) 또는 `pafValue` (FilterOperator, NumValue, NumValue2, Measure)입니다.
Sorts	FieldName, SortBy, SortOrder, Measure.
TopN	FieldName, Count, Measure, Others.
Explanation	정리된 제공자의 설명입니다.
Clear, IsEmpty, Assign	레이아웃을 비우고, 행, 열, 데이터 항목이 없는지 검사하고, 다른 레이아웃을 복사합니다.
ToJSON	스키마 형태의 간결한 JSON 객체 하나로 표현한 레이아웃입니다. 저장한 레이아웃을 다시 검증할 수 있습니다.
ToText	상태 표시줄용 레이아웃의 한 줄 요약입니다.

예제

채팅과 쿼리를 소유하고, 동기식으로 질문하며, 행 필드가 두 개를 넘는 레이아웃을 거부하는 도우미:

관련 항목

- 피벗에 질문하기
- TsgcPivotEngine
- TsgcVCLPivotGrid
- 데모

피벗에 질문하기

TsgcPivotGridAIQuery를 사용하면 사용자가 요청하는 것만으로 피벗을 설계할 수 있습니다. *online sales by region and quarter* 같은 질문은 행에 Region, 열에 분기별 Date, 값에 Sales의 합계, Channel에 대한 멤버 필터가 됩니다. 이 가이드는 AI 제공자에게 전송되는 내용, 무언가 바뀌기 전에 응답을 확인하는 방법, 비동기 모드, 요구 사항을 설명합니다.

질문에 응답하는 방식

1. Ask 또는 AskAsync는 질문과 엔진의 필드 카탈로그로 사용자 메시지를 만들며, OnBeforeAsk에서 이를 변경하거나 취소할 수 있습니다.
2. 채팅은 고정된 시스템 메시지와 닫힌 JSON 스키마와 함께 이를 제공자에게 보내고, JSON 객체 하나를 요청합니다.
3. 응답은 엔진의 필드에 대해 검증됩니다. 사용할 수 있는 행, 열, 데이터 필드가 남지 않으면 아무것도 적용되지 않고 OnError가 발생합니다.
4. OnLayout은 검증된 레이아웃을 받아 편집하거나 거부할 수 있습니다.
5. 레이아웃은 한 번의 재계산으로 현재 레이아웃을 교체하고, OnApplied가 발생하며, Undo로 이전 레이아웃을 복원할 수 있습니다.

전송되는 내용

세 개의 텍스트가 애플리케이션을 벗어나며, 그 외에는 아무것도 나가지 않습니다:

텍스트	내용
시스템 메시지	상수 `CS_SGC_PIVOT_AI_SYSTEM` : 스키마에 맞는 JSON 객체 하나로 답하고, 나열된 필드 이름만 사용하며, 필드나 값을 지어내지 말고, 질문, 필드 이름, 샘플 값 안에 있는 지시는 무시하라는 내용입니다.
사용자 메시지	`BuildPrompt` 의 텍스트: 질문, 설명의 언어, 소스 열마다 항목 하나를 가진 JSON 배열 `{"name","caption","type","samples"}` , 그리고 `"calculated":true` 로 표시된 엔진의 계산 필드입니다. ReplaceLayout이 False이면 현재 레이아웃도 포함합니다.
JSON 스키마	상수 `CS_SGC_PIVOT_AI_SCHEMA` (이름은 `sgcpivot_layout`)로, 제공자의 구조화된 출력 스키마로 전송됩니다.

- **소스 행은 전송되지 않습니다.** 제공자가 보는 것은 필드의 이름, 캡션, 형식과 몇 개의 샘플 값입니다.
- 샘플 값은 텍스트, 정수, 날짜, 불리언 필드에서 처음 10,000개 행을 읽어 얻은, 비어 있지 않은 서로 다른 처음 값들로, 필드당 최대 MaxSampleValues개이며 40자로 잘립니다. 보통 측정값인 Float 필드와 계산 필드는 샘플을 보내지 않습니다. 값을 전혀 보내지 않으려면 SendSampleValues를 False로 하거나 MaxSampleValues를 0으로 설정합니다.
- 필드의 캡션은 같은 이름의 레이아웃 필드에 Caption을 지정했다면 그 값이고, 아니면 열 이름입니다.
- AskYourPivot 데모의 *Show prompt* 버튼처럼, BuildPrompt를 호출하면 전송되는 내용을 그대로 사용자에게 보여 줄 수 있습니다.

열이 세 개인 헤드리스 엔진, 그것이 만드는 사용자 메시지, 같은 검증기로 확인하는 저장된 레이아웃의 예:

응답

스키마는 닫혀 있습니다. 모든 객체가 모든 키를 나열하며 다른 키는 허용하지 않습니다. 응답의 키는 `rows`와 `columns` (field와 group: none, year, quarter, month, week, day), `data` (field 와 aggregation: sum, count, average, min, max, countDistinct, stdDev, var), `filters` (멤버 필터, 레이블 필터 또는 값 필터), `sort` (레이블 또는 값 기준, 오름차순 또는 내림차순), `top` (선택적 Others 멤버가 있는 Top N), `explanation`입니다. 측정값은 `data` 항목의 0부터 시작하는 인덱스입니다.
일부 제공자는 구조화된 출력 모드를 거부합니다. 제공자 오류가 스키마 또는 응답 형식을 언급하면, 쿼리는 스키마를 사용자 메시지에 붙여 일반 JSON 모드로 한 번 더 질문합니다. 변경하는 네 가지 채팅 설정(ResponseFormat, JSONSchema, JSONSchemaName, SystemMessage)은 질문이 끝날 때마다 복원되고 채팅 기록은 지워지므로, 채팅을 다른 코드와 공유할 수 있습니다.

검증

응답은 신뢰할 수 없는 입력입니다. ValidateLayoutJSON은 제공자와 엔진 사이의 경계이며, 모든 응답은 OnLayout 전에 여기를 거칩니다:

- 응답은 64 KB에서 잘립니다. 먼저 코드 펜스의 첫 블록을 찾고, 그다음 문자열 안의 중괄호를 건너뛰며 균형 잡힌 첫 JSON 객체를 찾습니다.
- 필드 이름은 대소문자를 구분하지 않고 바깥 공백을 무시한 채 엔진의 필드와 정확히 일치해야 합니다. 계산 필드는 값에서만 허용됩니다. 알 수 없는 필드, 키, 열거형 값은 조용히 버려집니다.
- 텍스트 또는 불리언 필드의 숫자 집계는 Count가 되며, 설명에 그 사실이 적힙니다. 날짜 그룹화는 날짜 필드에서만 유지됩니다.
- 최대 행 4개, 열 4개, 데이터 6개, 필터 8개, 정렬 4개, Top N 2개 항목, 필터당 멤버 50개, N은 1에서 100 사이입니다.
- 제어 문자는 제거되고, 텍스트는 200자, 설명은 300자로 잘리며, 꺾쇠 괄호는 없습니다.
- 행 또는 열에 있는 필드는 필터 영역에 함께 놓이지 않습니다. 어느 축에도 없는 필드의 멤버 필터는 필터 영역으로 갑니다.
- 축 필드는 있으나 데이터가 없는 응답에는 첫 행 필드의 Count가 추가됩니다. 행, 열, 데이터 필드가 하나도 남지 않은 응답은 통째로 거부되며, 아무것도 바뀌지 않고 OnError가 발생합니다.

같은 메서드가 저장된 레이아웃도 검증하므로, 애플리케이션은 LastLayout.ToJSON을 저장해 두었다가 나중에 현재 필드에 대해 다시 확인할 수 있습니다.

적용과 Undo

검증된 레이아웃은 한 번의 재계산으로 엔진의 레이아웃을 교체합니다. 행, 열, 데이터 필드를 순서대로 적용한 다음, 필터, 정렬, Top N을 그 이름의 첫 필드(행에 있으면 행, 없으면 열)에 적용합니다. Undo는 마지막으로 적용된 응답 이전의 레이아웃을 복원하며, 그 후 LastLayout은 비게 됩니다.

비동기 질문

제공자가 응답하는 데 몇 초가 걸릴 수 있습니다. AskAsync는 요청만 작업자 스레드에서 실행하므로 창이 계속 응답합니다. OnBeforeAsk는 요청이 시작되기 전에 메인 스레드에서 발생하고, 응답은 Synchronize를 통해 메인 스레드에서 검증 및 적용되므로 OnLayout, OnApplied, OnError도 그곳에서 발생합니다.

- 한 번에 질문 하나: Busy인 동안 AskAsync는 False를 반환하고 OnError가 이를 알립니다.
- Busy인 동안에는 자신의 코드에서 채팅을 사용하지 마십시오.
- 요청이 실행되는 동안 쿼리를 해제하면 작업자를 기다린 뒤 응답을 버립니다.

에디션과 요구 사항

요구 사항	세부 정보
에디션	Registered 전용입니다. sgcPivotGridWS 브리지는 Community 에디션에 포함되지 않습니다.
sgcWebSockets	브리지는 sgcWebSockets의 TsgcAIChat (유닛 sgcAI)을 통해 제공자와 통신하므로, AI 컴포넌트가 포함된 sgcWebSockets가 설치되어 있어야 합니다. 제공자는 TsgcAIChat의 제공자입니다: OpenAI, Anthropic, Gemini, DeepSeek, Ollama, Grok, Mistral.
플랫폼	Windows 전용, Win32 및 Win64, Delphi 7에서 13까지와 그에 맞는 C++Builder 버전.
패키지	Delphi는 sgcPivotGridWS<V>와 dclsgcPivotGridWS<V>, C++Builder는 sgcPivotGridWSC<V>와 dclsgcPivotGridWSC<V>. sgcPivotGrid, sgcUICore, sgcWebSockets가 필요합니다.
API 키	각 제공자는 채팅의 ChatOptions.ApiKey에 설정하는 자체 키가 필요합니다. Ollama는 로컬에서 실행되며 키가 필요 없습니다. 키를 폼 파일에 저장하지 마십시오.

TsgcAIChat은 sgcWebSockets를 `SGC_AI`와 함께 컴파일한 경우에만 존재합니다. sgcWebSockets 설치 프로그램이 빌드하는 패키지는 이를 정의합니다. sgcWebSockets 소스를 직접 컴파일하고 컴파일러가 sgcPivotGridWS_AI에서 TsgcAI_Chat을 선언되지 않은 식별자로 보고하면, 해당 빌드의 조건부 정의에 `SGC_CUSTOM_INDY`를 추가하십시오. `SGC_AI`가 그것에 의존합니다.

AskYourPivot 데모

`Demos\PivotGrid\Delphi\AskYourPivot`은 생성된 판매 행 100,000개를 가진 TsgcVCLPivotGrid를 질문 입력란 아래에 배치합니다. 제공자를 고르고, 모델을 확인하고, 메모리에만 보관되는 API 키를 입력합니다. Ask는 AskAsync를 사용하고, Undo는 이전 레이아웃을 복원하며, Examples에는 샘플 질문 네 개가 있고, Show prompt는 시스템 메시지, 사용자 메시지, 스키마를 엽니다. `--test`로 실행하면 네트워크와 키 없이 미리 준비된 응답으로 확인할 수 있습니다. 데모를 참조하십시오.

관련 항목

- TsgcPivotGridAIQuery
- 영역
- 필터
- Top N

.NET 개요

sgcPivotGrid .NET은 Delphi sgcPivotGrid 라이브러리를 감싸는 래퍼가 아니라 C#으로 작성된 관리형 포트입니다. 클래스는 **esegece.sgcPivotGrid** 네임스페이스에 있으며 Delphi 이름을 그대로 유지하므로 **TsgcPivotEngine**, **TsgcPivotField**, **TsgcPivotResult**, **TsgcPivotGridRenderer**, **TsgcPivotFormatRule**, **TsgcPivotSource_DataSet**, **TsgcPivotSource_Table**은 두 언어에서 똑같이 읽힙니다. 뷰는 Delphi 이름 규칙을 따릅니다. WinForms 는 **TsgcWinFormsPivotGrid** 와 **TsgcWinFormsPivotFieldList**, WPF 는 **TsgcWpfPivotGrid** 와 **TsgcWpfPivotFieldList**, Avalonia 는 **TsgcAvaloniaPivotGrid** 와 **TsgcAvaloniaPivotFieldList** 입니다. 모두 **sgcUICore**의 .NET 포트로 그리며 eSeGeCe 비주얼 제품군의 다른 제품과 같은 캔버스, 글꼴, 라이트 및 다크 테마를 사용하므로, 피벗은 모든 호스트와 헤드리스 PNG 및 SVG 출력에서 똑같이 보입니다.
엔진과 그 연산은 작업 하나하나가 Delphi와 같습니다. 형식이 지정된 열 저장소, 여덟 가지 집계, 날짜 그룹화, 소계와 총계, 레이블 및 값 필터, 정렬, 확장과 축소, 원본 행으로의 드릴다운, 조건부 서식 규칙, CSV, JSON, HTML 내보내기가 포함됩니다. 렌더러의 SVG 출력은 같은 장면에 대해 Delphi 출력과 바이트 단위로 동일하므로, Linux 서버에서 만든 보고서는 Delphi 데스크톱 애플리케이션이 그리는 것과 일치합니다.

패키지

포트는 NuGet 패키지로 나뉘어 있으므로 프로젝트는 사용하는 것만 가져옵니다.

PackageId	포함 내용	대상 프레임워크
esegece.sgcPivotGrid	엔진(**TsgcPivotEngine**)과 그 열 저장소, 필드, 집계, 날짜 그룹화, 합계, 필터, 정렬, 드릴다운, 스냅샷(**TsgcPivotResult**), 서식 규칙, SVG 출력을 갖춘 렌더러 **TsgcPivotGridRenderer**, 필드 목록 모델, 두 소스, CSV, JSON, HTML 내보내기. Registered 에디션에는 계산 필드, 값 표시 방식, Top N, 레이아웃 JSON도 포함됩니다. 네이티브 종속성은 없습니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcPivotGrid.Rendering	SkiaSharp를 통한 PNG 출력(**SaveToPNG**).	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcPivotGrid.WinForms	**TsgcWinFormsPivotGrid**, **TsgcWinFormsPivotFieldList** 및 멤버 팝업.	net472, net8.0-windows, net9.0-windows
esegece.sgcPivotGrid.Wpf	**TsgcWpfPivotGrid**, **TsgcWpfPivotFieldList** 및 멤버 팝업.	net472, net8.0-windows, net9.0-windows
esegece.sgcPivotGrid.Avalonia	**TsgcAvaloniaPivotGrid**, **TsgcAvaloniaPivotFieldList** 및 멤버 팝업. Avalonia 11.3용.	net8.0, net9.0

각 호스트 패키지는 esegece.sgcPivotGrid.Rendering과 같은 호스트의 sgcUICore 패키지에 종속되므로 호스트 패키지만 추가하면 됩니다.

에디션

에디션	패키지 ID	포함 내용
Registered	esegece.sgcPivotGrid, .Rendering, .WinForms, .Wpf, .Avalonia	모든 것: 세 그리드와 필드 목록, 엔진, 두 소스, 모든 집계, 날짜 그룹화, 합계, 필터, 정렬, 드릴다운, 서식 규칙, 내보내

		기, 헤드리스 출력, 그리고 네 가지 고급 기능인 계산 필드, 값 표시 방식, Others가 있는 Top N, 레이아웃 영속화.
Community	.Community 접미사가 붙은 같은 ID입니다. 예: esegece.sgcPivotGrid.WinForms.Community.	네 가지 고급 기능을 제외한 모든 것: **CalculatedFields**, **ShowAs**, **ShowAsAxis**, **ShowAsBaseItem** 속성, **TopN**, **TopNOthers**, **OthersCaption**, **SaveLayout** 및 **LoadLayout** 메서드, **GetExpandState**와 **SetExpandState**.

이는 Delphi 에디션과 같은 구분입니다. 에디션을 참조하십시오.

호스트

호스트	Delphi 대응 항목	플랫폼
WinForms	TsgcVCLPivotGrid와 TsgcVCLPivotFieldList	Windows, .NET Framework 4.7.2와 .NET 8, 9.
WPF	TsgcFMXPivotGrid와 TsgcFMXPivotFieldList	Windows, .NET Framework 4.7.2와 .NET 8, 9.
Avalonia	TsgcFMXPivotGrid와 TsgcFMXPivotFieldList	Windows, Linux, macOS, .NET 8과 9.
헤드리스	TsgcPivotGridRenderer	Linux 서버를 포함한 .NET Standard 2.0의 모든 플랫폼.

관련 항목

- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- .NET과 Delphi의 차이점
- .NET 데이터 바인딩
- .NET 헤드리스 렌더링
- .NET 데모

NuGet에서 설치

패키지는 nuget.org에 있습니다. 호스트의 패키지를 추가하면 코어와 렌더링 패키지가 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcPivotGrid.WinForms
dotnet add package esegece.sgcPivotGrid.Wpf
dotnet add package esegece.sgcPivotGrid.Avalonia
```

창이 없는 서버나 콘솔 애플리케이션은 코어를 추가하고, PNG가 필요하면 렌더링 패키지를 추가합니다.

```
dotnet add package esegece.sgcPivotGrid
dotnet add package esegece.sgcPivotGrid.Rendering
```

Community 에디션

```
dotnet add package esegece.sgcPivotGrid.WinForms.Community
dotnet add package esegece.sgcPivotGrid.Wpf.Community
dotnet add package esegece.sgcPivotGrid.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcPivotGrid.WinForms" Version="2026.10.0" />
</ItemGroup>
```

프로젝트에서는 Registered 패키지나 Community 패키지 중 하나만 참조하고 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있습니다. 고급 기능을 사용하는 코드는 Community 패키지에 대해 컴파일되지 않습니다. 한 프로젝트에서 두 에디션을 모두 빌드할 때는 .NET과 Delphi의 차이점의 예제처럼 `#if !SGC_COMMUNITY`로 감싸십시오.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`가 있는 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`가 있는 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서 네임스페이스를 `xmlns:pivot="clr-namespace:esegece.sgcPivotGrid;assembly=esegece.sgcPivotGrid.Wpf"`로 매핑합니다.
- Avalonia: Avalonia 11.3의 **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 패키지가 있는 `net8.0` 또는 `net9.0` 프로젝트로, 일반적인 Avalonia 데스크톱 애플리케이션과 같습니다. XAML에서 네임스페이스를 `xmlns:pivot="using:esegece.sgcPivotGrid"`로 매핑합니다.
- 헤드리스: 엔진, 두 소스, CSV, JSON, HTML, SVG에는 esegece.sgcPivotGrid만으로 충분합니다. PNG에는 esegece.sgcPivotGrid.Rendering을 추가하며, SkiaSharp와 그 네이티브 자산이 함께 설치됩니다.

C#에서는 모든 호스트에 `using esegece.sgcPivotGrid;`가 필요하고, **TsgcUITheme**, **sgcUICore_Theme** 및 기타 테마 형식에는 `using esegece.sgcUICore;`도 필요합니다.

관련 항목

- .NET 개요
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 헤드리스 렌더링
- .NET 데모

WinForms 빠른 시작

이 예제는 생성된 50,000개의 판매 행에 대해 피벗 그리드와 필드 목록이 있는 WinForms 폼을 만듭니다. 행에는 Region과 Product, 열에는 Date 열의 연도, 셀에는 판매 합계를 배치합니다. 확인란으로 그리드와 필드 목록의 라이트 테마와 다크 테마를 전환하고, 셀을 두 번 클릭하면 그 뒤에 있는 원본 행을 읽습니다.

단계

- `net8.0-windows` WinForms 프로젝트를 만들고 **esegece.sgcPivotGrid.WinForms** 패키지를 추가합니다. NuGet에서 설치를 참조하십시오.
- **TsgcWinFormsPivotGrid**를 만들고 도킹합니다. **Pivot** 속성은 행과 레이아웃을 보관하는 **TsgcPivotEngine**입니다.
- 행은 **BeginLoad**와 **EndLoad** 사이에서 로드합니다. **DefineColumn**이 각 열의 이름과 형식을 정하고, **AppendRow**가 열 순서대로 값을 추가합니다. 엔진은 가장 바깥쪽 EndLoad에서 한 번만 다시 계산합니다.
- 레이아웃은 **BeginUpdate**와 **EndUpdate** 사이에서 만듭니다. **AddField**는 열을 행, 열, 필터에 배치하며 선택적으로 날짜 그룹화를 지정하고, **AddDataField**는 집계가 있는 측정값을 추가합니다.
- **TsgcWinFormsPivotFieldList**를 만들고 **PivotGrid**를 그리드로 설정합니다. 사용자는 네 영역 사이에서 필드를 끌어 놓고 멤버를 필터링합니다.

예제

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcPivotGrid;
using esegece.sgcUICore;

namespace PivotQuickStartWinForms
{
    internal class MainForm : Form
    {
        private static readonly string[] Regions = { "North", "South", "East", "West" };
        private static readonly string[] Products = { "Laptop", "Phone", "Tablet", "Monitor" };
        private readonly TsgcWinFormsPivotGrid Grid = new TsgcWinFormsPivotGrid();
        private readonly TsgcWinFormsPivotFieldList FieldList = new TsgcWinFormsPivotFieldList();
        private readonly CheckBox chkDark = new CheckBox();

        public MainForm()
        {
            Text = "sgcPivotGrid Quick Start";
            ClientSize = new Size(1000, 600);

            Grid.Dock = DockStyle.Fill;
            Grid.AlternateRows = true;
            FieldList.Dock = DockStyle.Right;
            FieldList.Width = 260;
            chkDark.Text = "Dark theme";
            chkDark.Dock = DockStyle.Top;
            chkDark.CheckedChanged += (s, e) =>
            {
                Grid.Theme = chkDark.Checked ? TsgcUITheme.utDark : TsgcUITheme.utLight;
                FieldList.Theme = Grid.Theme;
            };
            Controls.Add(Grid);
            Controls.Add(FieldList);
            Controls.Add(chkDark);

            // Pivot is the engine owned by the grid: load the rows, then set the layout
            LoadSales(Grid.Pivot);
            Grid.Pivot.BeginUpdate();
```

```
            try
            {
                Grid.Pivot.AddField("Region", TsgcPivotArea.pvaRow);
                Grid.Pivot.AddField("Product", TsgcPivotArea.pvaRow);
                Grid.Pivot.AddField("Date", TsgcPivotArea.pvaColumn, TsgcPivotDateGrouping.pvdYear).Caption = "Year";
                Grid.Pivot.AddDataField("Sales", TsgcPivotAggregation.pvfSum).DisplayFormat = "#,##0";
            }
            finally
            {
                // one recalculation for the whole layout
                Grid.Pivot.EndUpdate();
            }

            // the user moves the fields between the areas with the field list
            FieldList.PivotGrid = Grid;
            // Enter or a double click on a cell
            Grid.OnDrillDown = DoDrillDown;
        }

        private static void LoadSales(TsgcPivotEngine aEngine)
        {
            Random oRandom = new Random(2026);
            aEngine.BeginLoad(true);
            try
            {
                aEngine.DefineColumn("Region", TsgcPivotDataType.pvtText);
                aEngine.DefineColumn("Product", TsgcPivotDataType.pvtText);
                aEngine.DefineColumn("Date", TsgcPivotDataType.pvtDateTime);
                aEngine.DefineColumn("Sales", TsgcPivotDataType.pvtFloat);
                for (int i = 0; i < 50000; i++)
                {
                    string vRegion = Regions[oRandom.Next(Regions.Length)];
                    string vProduct = Products[oRandom.Next(Products.Length)];
                    DateTime vDate = new DateTime(2024, 1, 1).AddDays(oRandom.Next(1096));
                    double vSales = Math.Round(50 + oRandom.NextDouble() * 950, 2);
                    aEngine.AppendRow(vRegion, vProduct, vDate, vSales);
                }
            }
            finally
            {
                // the engine recalculates once, at the outermost EndLoad
                aEngine.EndLoad();
            }
        }

        private void DoDrillDown(object Sender, int aRowItem, int aDataCol, ref bool aHandled)
        {
            // the source rows behind the cell, as an IsgcUITable
            TsgcPivotSourceView oRows = Grid.Pivot.DrillDown(aRowItem, aDataCol);
            Text = "sgcPivotGrid Quick Start: " + oRows.TableRowCount() + " source rows";
            aHandled = true;
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

Enter 키나 셀 더블 클릭은 **OnDrillDown**을 발생시킵니다. 엔진의 **DrillDown**은 셀의 원본 행을 테이블로 반환하며, sgcGrid 그리드나 두 번째 피벗이 이를 표시할 수 있습니다. .NET 데이터 바인딩을 참조하십시오. 멤버의 +/- 단추를 클릭하거나 행 머리글을 더블 클릭하면 확장되고 축소됩니다.

관련 항목

- NuGet에서 설치
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- TsgcVCLPivotGrid
- 드릴다운
- .NET 데모

WPF 빠른 시작

같은 피벗을 WPF에서 코드로 만듭니다. 위쪽에 테마 확인란, 오른쪽에 필드 목록, 창의 나머지에 그리드가 있는 **DockPanel** 입니다. Date 열은 연도와 분기로 그룹화하여 열에 두 번 넣고, 값 영역에는 같은 열의 두 측정값인 판매 합계와 평균을 둡니다. 셀을 클릭하면 그 텍스트가 제목 표시줄에 표시됩니다.

단계

- `net8.0-windows` WPF 프로젝트를 만들고 **esegece.sgcPivotGrid.Wpf** 패키지를 추가합니다.
- **TsgcWpfPivotGrid**와 **TsgcWpfPivotFieldList**를 만들어 둘 다 창에 추가하고, 필드 목록의 **PivotGrid**를 그리드로 설정합니다.
- **Pivot**은 WinForms 예제와 같은 방식으로 채웁니다. 엔진, 레이아웃, 이벤트는 모든 호스트에서 동일합니다.
- 결과는 엔진의 **Lock**과 **Unlock** 사이에서 읽습니다. **Result**는 변경할 수 없는 스냅샷이므로 다시 계산해도 읽고 있는 셀은 바뀌지 않습니다.

예제

```
using System;
using System.Windows;
using System.Windows.Controls;
using esegece.sgcPivotGrid;
using esegece.sgcUICore;

namespace PivotQuickStartWpf
{
    internal class MainWindow : Window
    {
        private static readonly string[] Regions = { "North", "South", "East", "West" };
        private static readonly string[] Channels = { "Online", "Retail", "Partner" };
        private readonly TsgcWpfPivotGrid Grid = new TsgcWpfPivotGrid();
        private readonly TsgcWpfPivotFieldList FieldList = new TsgcWpfPivotFieldList();
        private readonly CheckBox chkDark = new CheckBox();

        public MainWindow()
        {
            Title = "sgcPivotGrid Quick Start";
            Width = 1000;
            Height = 600;

            chkDark.Content = "Dark theme";
            chkDark.Margin = new Thickness(8, 4, 8, 4);
            chkDark.Click += (s, e) =>
            {
                Grid.Theme = chkDark.IsChecked == true ? TsgcUITheme.utDark : TsgcUITheme.utLight;
                FieldList.Theme = Grid.Theme;
            };
            FieldList.Width = 260;
            DockPanel oRoot = new DockPanel();
            DockPanel.SetDock(chkDark, Dock.Top);
            DockPanel.SetDock(FieldList, Dock.Right);
            oRoot.Children.Add(chkDark);
            oRoot.Children.Add(FieldList);
            // the last child fills the rest of the window
            oRoot.Children.Add(Grid);
            Content = oRoot;

            TsgcPivotEngine oEngine = Grid.Pivot;
            Random oRandom = new Random(2026);
            oEngine.BeginLoad(true);
            try
```

```
            {
                oEngine.DefineColumn("Region", TsgcPivotDataType.pvtText);
                oEngine.DefineColumn("Channel", TsgcPivotDataType.pvtText);
                oEngine.DefineColumn("Date", TsgcPivotDataType.pvtDateTime);
                oEngine.DefineColumn("Sales", TsgcPivotDataType.pvtFloat);
                for (int i = 0; i < 50000; i++)
                {
                    string vRegion = Regions[oRandom.Next(Regions.Length)];
                    string vChannel = Channels[oRandom.Next(Channels.Length)];
                    DateTime vDate = new DateTime(2024, 1, 1).AddDays(oRandom.Next(1096));
                    double vSales = Math.Round(50 + oRandom.NextDouble() * 950, 2);
                    oEngine.AppendRow(vRegion, vChannel, vDate, vSales);
                }
            }
            finally
            {
                oEngine.EndLoad();
            }

            oEngine.BeginUpdate();
            try
            {
                oEngine.AddField("Region", TsgcPivotArea.pvaRow);
                oEngine.AddField("Date", TsgcPivotArea.pvaColumn, TsgcPivotDateGrouping.pvdYear).Caption = "Year";
                oEngine.AddField("Date", TsgcPivotArea.pvaColumn, TsgcPivotDateGrouping.pvdQuarter).Caption = "Quarter";
                oEngine.AddField("Channel", TsgcPivotArea.pvaFilter);
                oEngine.AddDataField("Sales", TsgcPivotAggregation.pvfSum).DisplayFormat = "#,##0";
                oEngine.AddDataField("Sales", TsgcPivotAggregation.pvfAverage).DisplayFormat = "#,##0.00";
            }
            finally
            {
                oEngine.EndUpdate();
            }

            FieldList.PivotGrid = Grid;
            Grid.OnCellClick = DoCellClick;
        }

        private void DoCellClick(object Sender, TsgcPivotHitInfo aHit)
        {
            if (aHit.Part != TsgcPivotHitPart.pvhCell)
                return;
            // read the snapshot between Lock and Unlock
            Grid.Pivot.Lock();
            try
            {
                Title = "sgcPivotGrid Quick Start: " + Grid.Pivot.Result.CellText(aHit.RowItem, aHit.DataCol);
            }
            finally
            {
                Grid.Pivot.Unlock();
            }
        }
    }

    internal static class Program
    {
        [STAThread]
        private static void Main()
        {
            Application oApp = new Application();
            oApp.Run(new MainWindow());
        }
    }
}
```

WPF 그리드는 FireMonkey 그리드처럼 자체 스크롤 막대를 그리며 장치 독립적 픽셀로 동작합니다. **HitTest**는 DIP를 받습니다. 표시 옵션은 일반 CLR 속성이므로 코드나 XAML 특성으로 설정합니다.

관련 항목

- NuGet에서 설치
- WinForms 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- .NET과 Delphi의 차이점
- TsgcFMXPivotGrid

Avalonia 빠른 시작

같은 피벗을 Windows, Linux, macOS용 Avalonia에서 만듭니다. 지역별 네 제품의 수량을 표시합니다. 확인란은 그리드의 **AlternateRows**에 바인딩되고, 제목은 선택한 행을 따라갑니다.

단계

- **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 11.3이 있는 `net8.0` Avalonia 데스크톱 프로젝트를 만들고 **esegece.sgcPivotGrid.Avalonia** 패키지를 추가합니다.
- 애플리케이션 스타일에 **FluentTheme**을 추가합니다. 필드 목록의 멤버 팝업은 확인란과 검색 상자의 Fluent 템플릿을 사용합니다.
- **TsgcAvaloniaPivotGrid**와 **TsgcAvaloniaPivotFieldList**를 만들고, 필드 목록의 **PivotGrid**를 그리드로 설정하고 , **Pivot**은 다른 호스트와 같은 방식으로 채웁니다.

예제

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Data;
using Avalonia.Themes.Fluent;
using esegece.sgcPivotGrid;

namespace PivotQuickStartAvalonia
{
    internal class MainWindow : Window
    {
        private static readonly string[] Regions = { "North", "South", "East", "West" };
        private static readonly string[] Products = { "Laptop", "Phone", "Tablet", "Monitor" };
        private readonly TsgcAvaloniaPivotGrid Grid = new TsgcAvaloniaPivotGrid();
        private readonly TsgcAvaloniaPivotFieldList FieldList = new TsgcAvaloniaPivotFieldList();
        private readonly CheckBox chkAlternate = new CheckBox();

        public MainWindow()
        {
            Title = "sgcPivotGrid Quick Start";
            Width = 1000;
            Height = 600;

            chkAlternate.Content = "Alternate rows";
            chkAlternate.IsChecked = true;
            // the display options are styled properties: they take bindings and styles
            Grid.Bind(TsgcAvaloniaPivotGrid.AlternateRowsProperty, new Binding("IsChecked") { Source = chkAlternate });
            FieldList.Width = 260;
            DockPanel oRoot = new DockPanel();
            DockPanel.SetDock(chkAlternate, Dock.Top);
            DockPanel.SetDock(FieldList, Dock.Right);
            oRoot.Children.Add(chkAlternate);
            oRoot.Children.Add(FieldList);
            oRoot.Children.Add(Grid);
            Content = oRoot;

            TsgcPivotEngine oEngine = Grid.Pivot;
            Random oRandom = new Random(2026);
            oEngine.BeginLoad(true);
            try
            {
                oEngine.DefineColumn("Region", TsgcPivotDataType.pvtText);
                oEngine.DefineColumn("Product", TsgcPivotDataType.pvtText);
```

```
                oEngine.DefineColumn("Date", TsgcPivotDataType.pvtDateTime);
                oEngine.DefineColumn("Quantity", TsgcPivotDataType.pvtInteger);
                for (int i = 0; i < 50000; i++)
                {
                    string vRegion = Regions[oRandom.Next(Regions.Length)];
                    string vProduct = Products[oRandom.Next(Products.Length)];
                    DateTime vDate = new DateTime(2024, 1, 1).AddDays(oRandom.Next(1096));
                    int vQuantity = 1 + oRandom.Next(10);
                    oEngine.AppendRow(vRegion, vProduct, vDate, vQuantity);
                }
            }
            finally
            {
                oEngine.EndLoad();
            }

            oEngine.BeginUpdate();
            try
            {
                oEngine.AddField("Product", TsgcPivotArea.pvaRow);
                oEngine.AddField("Region", TsgcPivotArea.pvaColumn);
                oEngine.AddDataField("Quantity", TsgcPivotAggregation.pvfSum).DisplayFormat = "#,##0";
            }
            finally
            {
                oEngine.EndUpdate();
            }

            FieldList.PivotGrid = Grid;
            // SelectedRow and SelectedCol are direct properties that raise their change
            Grid.PropertyChanged += (s, e) =>
            {
                if (e.Property == TsgcAvaloniaPivotGrid.SelectedRowProperty)
                    Title = "sgcPivotGrid Quick Start: row item " + Grid.SelectedRow;
            };
        }
    }

    internal class App : Application
    {
        public override void Initialize()
        {
            Styles.Add(new FluentTheme());
        }

        public override void OnFrameworkInitializationCompleted()
        {
            IClassicDesktopStyleApplicationLifetime oDesktop =
                ApplicationLifetime as IClassicDesktopStyleApplicationLifetime;
            if (oDesktop != null)
                oDesktop.MainWindow = new MainWindow();
            base.OnFrameworkInitializationCompleted();
        }
    }

    internal static class Program
    {
        [STAThread]
        private static int Main(string[] args)
        {
            return AppBuilder.Configure<App>().UsePlatformDetect().LogToTrace()
                .StartWithClassicDesktopLifetime(args);
        }
    }
}
```

Avalonia 그리드의 모든 표시 옵션은 스타일 속성이므로 바인딩, 스타일, XAML 값을 받습니다. **SelectedRow**와 **SelectedCol**은 양방향 직접 속성입니다. macOS에서는 Cmd 키가 Ctrl+Home과 Ctrl+End에서 Ctrl처럼 동작하며, 가로 휠은 열을 스크롤합니다.

관련 항목

- NuGet에서 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- .NET 데이터 바인딩
- .NET과 Delphi의 차이점
- TsgcFMXPivotGrid

.NET 클래스

모든 클래스는 **esegece.sgcPivotGrid** 네임스페이스에 있습니다. 그리드와 필드 목록은 프레임워크 중립적인 엔진, 렌더러, 필드 목록 모델 위의 얇은 호스트이며, 이들은 호스트, 헤드리스 출력, 소스가 공유합니다.

호스트 클래스

호스트	클래스	기본 클래스	Delphi
WinForms	TsgcWinFormsPivotGrid	TsgcUIWinFormsControl	TsgcVCLPivotGrid
WinForms	TsgcWinFormsPivotFieldList	TsgcUIWinFormsControl	TsgcVCLPivotFieldList
WPF	TsgcWpfPivotGrid	TsgcUIWpfControl	TsgcFMXPivotGrid
WPF	TsgcWpfPivotFieldList	TsgcUIWpfControl	TsgcFMXPivotFieldList
Avalonia	TsgcAvaloniaPivotGrid	TsgcUIAvaloniaControl	TsgcFMXPivotGrid
Avalonia	TsgcAvaloniaPivotFieldList	TsgcUIAvaloniaControl	TsgcFMXPivotFieldList

각 필드 목록의 멤버 체크리스트는 **TsgcWinFormsPivotMemberPopup**(Form), **TsgcWpfPivotMemberPopup**(Window), **TsgcAvaloniaPivotMemberPopup**(Popup) 중 하나입니다. 세 그리드는 소스나 필드 목록이 엔진을 찾을 수 있도록 **IsgcPivotEngineHost**를, sgcCharts 차트나 다른 피벗이 결과를 읽을 수 있도록 **IsgcUITable**을 구현합니다.

그리드 멤버

세 그리드는 같은 멤버를 공유합니다.

Kind	멤버
엔진과 그리기	**Pivot**(**TsgcPivotEngine**), **Renderer**, **FormatRules**.
표시 옵션	**RowLayout**, **RowHeight**, **HeaderHeight**, **IndentWidth**, **MinColumnWidth**, **MaxColumnWidth**, **AutoColumnWidths**, **GridLines**, **AlternateRows**, **TotalsBold**, **ShowExpandButtons**, **Theme**, **AccentColor**, **FontSize**.
위치	**TopRow**, **ScrollX**, **SelectedRow**, **SelectedCol**, **ScrollToCell**, **HitTest**.
출력	**Recalculate**, **SaveToPNG**와 **SaveToSVG**(파일 또는 스트림, 너비와 높이는 선택 사항이며 0은 전체 뷰), **SaveToCSV**, **SaveToJSON**, **SaveToHTML**. Registered 에디션: **SaveLayoutToString**, **LoadLayoutFromString**, **SaveLayoutToFile**, **LoadLayoutFromFile**.
이벤트	**OnCellClick**, **OnCellDblClick**, **OnDrillDown**, **OnExpandChange**, **OnGetCellStyle**, **OnGetCellText**, **OnSelectionChange**, **OnColumnResized**.

필드 목록에는 **PivotGrid**(모든 **IsgcPivotEngineHost**), **Engine**, **Theme**, **AccentColor**, **FontSize**, **ShowFilterArea**, **ApplyChanges**가 있는 **DeferUpdate**, 그리고 **OnBeforeFieldMove**와 **OnFieldMoved** 이벤트가 있습니다.

프레임워크 중립 형식

형식	설명
TsgcPivotEngine	열 저장소, 레이아웃, 계산, 스냅샷, 드릴다운, 내보내기: **BeginLoad**, **DefineColumn**, **AppendRow**, **EndLoad**, **LoadFromTable**, **AddField**, **AddDataField**, **MoveField**, **RemoveField**, **ClearLayout**, **ExpandAll**, **CollapseAll**, **Lock**, **Unlock**, **Result**, **ResultTable**, **GetSourceRows**, **DrillDown**, **GetMemberLabels**, 그리고 **OnCalculated**, **OnLayoutChange**, **OnFilterRow** 이벤트. TsgcPivotEngine을 참조하십시오.
TsgcPivotFields, TsgcPivotField	레이아웃의 필드: **FieldName**, **Caption**, **Area**, **AreaIndex**, **DateGrouping**, **Aggregation**, **DisplayFormat**, **SortBy**, **SortOrder**, **SortMeasure**, **Subtotals**, **Expanded**, **FilterValues**, **FilterExclude**, **LabelFilter**, **ValueFilter**, 그리고 Registered 에디션에서는 **TopN**, **TopNOthers**, **ShowAs**, **ShowAsAxis**, **ShowAsBaseItem**. TsgcPivotField를 참조하십시오.
TsgcPivotCalculatedFields, TsgcPivotCalculatedField	식에서 원본 행마다 계산되는 숫자 열. Registered 에디션. TsgcPivotCalculatedField를 참조하십시오.
TsgcPivotResult, TsgcPivotItem	다시 계산의 변경할 수 없는 스냅샷: **ItemCount**, **Item**, **NodeLabel**, **NodePath**, **LevelCount**, **MeasureCount**, **DataColCount**, **CellText**, **CellValue**, **RawValue**. **TsgcPivotItem**은 **Kind**(`pvkMember` , `pvkTotal` , `pvkGrandTotal`), **Level**, **Node**, **Expandable**, **Expanded**를 가진 구조체입니다.
TsgcPivotResultTable, TsgcPivotSourceView	엔진의 결과와 드릴다운의 원본 행으로, 둘 다 **IsgcUITable**로 제공됩니다.
TsgcPivotSource_DataSet	`System.Data.DataTable` 을 로드하고 테이블이 바뀌면 다시 로드합니다. .NET 데이터 바인딩을 참조하십시오.
TsgcPivotSource_Table	모든 **IsgcUITable**을 로드하고 버전이 바뀌면 다시 로드합니다. TsgcPivotSource_Table을 참조하십시오.
TsgcPivotGridRenderer, TsgcPivotRenderOptions, TsgcPivotRenderState	스냅샷을 **TsgcUICanvas**에 그리고, 히트 테스트를 수행하며, SVG로 또는 esegece.sgcPivotGrid.Rendering을 사용해 PNG로 기록합니다. .NET 헤드리스 렌더링을 참조하십시오.
TsgcPivotFormatRules, TsgcPivotFormatRule	조건부 서식: 값 규칙과 색상 스케일. TsgcPivotFormatRule을 참조하십시오.
TsgcPivotFieldListModel	창이 없는 필드 목록: 영역, 칩, 끌어서 놓기, 필터 상태를 세 필드 목록이 그립니다.
sgcPivotGrid_IO	결과의 정적 CSV, JSON, HTML 내보내기이며 Delphi 기본 인수를 사용합니다. 엔진과 그리드는 이를 **SaveToCSV**, **SaveToJSON**, **SaveToHTML**로 감쌉니다.

열거형

형식	값
TsgcPivotArea	`pvaRow` , `pvaColumn` , `pvaData` , `pvaFilter`
TsgcPivotAggregation	`pvfSum` , `pvfCount` , `pvfAverage` , `pvfMin` , `pvfMax` , `pvfCountDistinct` , `pvfStdDev` , `pvfVar`
TsgcPivotDateGrouping	`pvdNone` , `pvdYear` , `pvdQuarter` , `pvdMonth` , `pvdWeek` , `pvdDay`
TsgcPivotDataType	`pvtText` , `pvtInteger` , `pvtFloat` , `pvtDateTime` , `pvtBoolean`

TsgcPivotAxisKind	`pvxRows`, `pvxColumns`
TsgcPivotSortBy, TsgcPivotSortOrder	`pvsLabel`, `pvsValue`; `pvoAscending`, `pvoDescending`
TsgcPivotRowLayout	`pvlCompact`, `pvlTabular`
TsgcPivotShowAs	`pvwValue`, `pvwPercentOfRow`, `pvwPercentOfColumn`, `pvwPercentOfGrandTotal`, `pvwRunningTotal`, `pvwDifferenceFrom`, `pvwRank`. Registered 에디션.

관련 항목

- .NET 개요
- .NET과 Delphi의 차이점
- .NET 데이터 바인딩
- .NET 헤드리스 렌더링
- 아키텍처

.NET과 Delphi의 차이점

.NET 포트는 Delphi의 이름, 기본값, 연산, 렌더링 규칙을 유지합니다. SVG 출력과 일반 데이터의 CSV, JSON, HTML 내보내기는 Delphi 출력과 바이트 단위로 동일합니다. 아래의 차이점은 호스트 프레임워크와 .NET 클래스 라이브러리에서 비롯됩니다.

컴포넌트 스트리밍 없음

.NET에는 DFM이 없으므로 디자인 타임 컬렉션 편집기, FieldName 드롭다운 목록, **Recalculate** 및 **Clear Layout** 동사가 없습니다. 엔진이 한 번만 다시 계산하도록 레이아웃은 **BeginUpdate**와 **EndUpdate** 사이에서 **AddField**와 **AddDataField**로 코드에서 만드십시오. Delphi 엔진은 폼이 로드되는 동안 작업을 미루지만, .NET에서는 **BeginUpdate**와 **BeginLoad**가 그 역할을 합니다. 세션 사이에 레이아웃을 유지하려면 Registered 에디션의 레이아웃 JSON을 사용하십시오. 레이아웃 영속화를 참조하십시오.

인덱스 속성, 이벤트, 목록

- Delphi의 인덱스 속성은 메서드가 됩니다. `Result.Item(axis, i)`, `NodeLabel(axis, node)`, `SourceColumnName(i)`, 그리고 멤버 팝업에서는 `Members(i)`와 `SetMemberChecked(i, value)`가 있는 `MemberChecked(i)`입니다. Delphi 함수는 메서드로 남습니다. `SourceRowCount()`, `MeasureCount()`, `DataColCount()`. 컬렉션은 `Fields[0]`처럼 인덱서를 유지합니다.
- 이벤트는 Delphi와 같은 이름의 델리게이트 속성입니다(**OnCalculated**, **OnDrillDown** 등). 처리기 하나는 `=`로 할당하고 여러 개는 `+=`로 연결합니다. Delphi의 `var` 매개변수는 `ref` 매개변수(`aHandled`, `aAccept`, `aAllow`)가 되며, `ref` 매개변수가 있는 람다는 모든 매개변수의 형식을 선언합니다. **TNotifyEvent**는 **TsgcUINotifyEvent**입니다.
- **TStrings** 매개변수는 `IList<string>`가 됩니다. **FilterValues**는 **TsgcPivotFilterValues**로, TStringList처럼 **IndexOf**가 대소문자를 구분하지 않는 순서 있는 문자열 목록입니다. 필터 자체는 멤버 레이블을 대소문자를 구분하여 비교합니다.
- 다른 클래스의 필드, 필터, 규칙, 계산 필드를 **Assign**하면 Delphi의 EConvertError에 해당하는 `InvalidCastException`이 발생합니다.

```
using System;
using System.Collections.Generic;
using esegece.sgcPivotGrid;

namespace PivotDifferences
{
    internal static class GridWiring
    {
        public static void Wire(TsgcWinFormsPivotGrid aGrid, Action<string> aLog)
        {
            TsgcPivotEngine oEngine = aGrid.Pivot;
            // events are delegate properties: assign one handler with =, or chain with +=
            oEngine.OnCalculated = Sender => aLog("recalculated");
            aGrid.OnExpandChange += (Sender, aAxis, aItem) => aLog("expanded or collapsed " + aItem);
            // a lambda with a ref parameter declares every parameter type
            oEngine.OnFilterRow = (object Sender, int aRow, ref bool aAccept) =>
                aAccept = oEngine.SourceValue(aRow, 0) != null;

            // Delphi indexed properties are methods: Item(axis, i), NodeLabel(axis, node)
            oEngine.Lock();
            try
            {
                TsgcPivotResult oResult = oEngine.Result;
                if (oResult != null && oResult.ItemCount(TsgcPivotAxisKind.pvxRows) > 0)
                {
                    TsgcPivotItem vItem = oResult.Item(TsgcPivotAxisKind.pvxRows, 0);
                    if (vItem.Kind != TsgcPivotItemKind.pvkGrandTotal)
                        aLog(oResult.NodeLabel(TsgcPivotAxisKind.pvxRows, vItem.Node));
                }
            }
            finally
```

```
            {
                oEngine.Unlock();
            }

            // a TStrings parameter is an IList<string>
            if (oEngine.Fields.Count > 0)
            {
                List<string> vMembers = new List<string>();
                oEngine.GetMemberLabels(oEngine.Fields[0], vMembers);
                aLog(vMembers.Count + " members");
            }

#if !SGC_COMMUNITY
            // layout persistence is part of the Registered edition only
            string vLayout = aGrid.SaveLayoutToString();
            aGrid.LoadLayoutFromString(vLayout);
#endif
        }
    }
}
```

값, 날짜, 문화권

AppendRow는 CLR 값을 받습니다. null과 DBNull은 빈 셀, 정수 형식은 `pvtInteger`, `float`, `double`, `decimal`은 `pvtFloat`, `DateTime`은 `pvtDateTime`, `bool`은 `pvtBoolean`, `string`과 `char`는 텍스트입니다. 날짜는 Delphi의 TDateTime인 OLE 날짜로 저장되므로 날짜 그룹화와 주 번호가 Delphi와 일치합니다. 텍스트는 인바리언트 문화권으로만 구문 분석되며 Delphi는 사용자 로캘도 시도합니다. 멤버 레이블, **CellText**, 내보내기는 숫자와 날짜를 인바리언트 설정(소수점은 마침표, 날짜는 MM/dd/yyyy)으로 서식화하며 스레드의 문화권은 사용하지 않습니다.

TDataSet 대신 DataTable

TsgcPivotSource_DataSet은 Delphi TDataSet에 대응하는 .NET의 `System.Data.DataTable`을 바인딩합니다. DataTable은 항상 열려 있고 커서가 없으므로 현재 레코드, 책갈피, DisableControls가 없습니다. Delphi 데이터 링크 대신 `RowChanged`, `RowDeleted`, `TableCleared` 이벤트를 사용합니다. Delphi가 게시되지 않은 편집과 삽입을 무시하는 것처럼, 끝나지 않은 편집과 분리된 새 행은 무시됩니다. .NET 데이터 바인딩을 참조하십시오.

해제 알림 없음

소스나 필드 목록의 **PivotGrid**가 WinForms 그리드처럼 `IComponent`이면, 소스는 그 `Disposed` 이벤트를 수신하고 그때 **PivotGrid**를 지웁니다. WPF와 Avalonia 그리드는 컴포넌트가 아닙니다. 그리드를 버리기 전에 **PivotGrid**를 null로 설정하십시오. **Engine**은 감시되지 않으므로 엔진이 사라지기 전에 지우십시오. **TsgcPivotSource_DataSet**과 **TsgcPivotSource_Table**은 `IDisposable`이며, **Dispose**가 테이블에서 리스너를 제거합니다.

스레드

계산은 Delphi처럼 동기식입니다. 다시 계산은 그것을 일으킨 스레드에서 실행되고 새 스냅샷도 그곳에서 게시됩니다. 그런 다음 그리드는 UI 스레드에서 다시 그려지며, WinForms에서는 게시된 창 메시지, WPF에서는 `Dispatcher.BeginInvoke`, Avalonia에서는 `Dispatcher.UIThread.Post`를 사용합니다. 다른 스레드에서 **Result**를 읽는 코드는 엔진의 **Lock**과 **Unlock**을 잡고 있어야 합니다.

호스트별 스크롤 막대, 단위, 속성

- **TsgcWinFormsPivotGrid**는 VCL 그리드처럼 네이티브 Windows 스크롤 막대와 픽셀을 사용합니다. **TsgcWpfPivotGrid**와 **TsgcAvaloniaPivotGrid**는 FireMonkey 그리드처럼 썸 끌기와 트랙 페이지 이동을 지원하는 자체 스크롤 막대를 그리고 장치 독립적 픽셀로 동작합니다.
- Delphi의 **BorderStyle**은 WPF와 Avalonia에서 **BorderThickness**(기본값 1)와 **BorderBrush**입니다. **TsgcWpfPivotGrid**는 `new`로 자체 **FontSize**를 선언하고, **TsgcAvaloniaPivotGrid**는 Avalonia 컨트롤 테마를 가리

는 **TsgcUITheme** 형식의 자체 **Theme**를 선언합니다. `Control` 형식의 참조를 통하면 이들은 프레임워크 속성이며, 그리드는 이를 읽지 않습니다.
- WPF 옵션은 CLR 속성입니다. Avalonia 옵션은 바인딩과 스타일을 받는 스타일 속성입니다. .NET 데이터 바인딩을 참조하십시오.
- 필드 목록의 멤버 팝업은 WinForms에서는 Form, WPF에서는 필드 목록 창이 소유하는 테두리 없는 Window, Avalonia에서는 바깥을 클릭하면 닫히는 Popup입니다. macOS에서는 Cmd 키가 Ctrl처럼 동작합니다.

관련 항목

- .NET 개요
- .NET 클래스
- .NET 데이터 바인딩
- WPF 빠른 시작
- Avalonia 빠른 시작
- TsgcPivotEngine

.NET 데이터 바인딩

엔진은 원본 행의 형식이 지정된 자체 복사본을 보관하므로, 피벗은 계산하거나 그리는 동안 데이터 소스를 읽지 않습니다. 이를 채우는 방법은 네 가지입니다.

방법	용도
BeginLoad, **DefineColumn**, **AppendRow**, **EndLoad**	코드에서 생성하거나 임의의 리더에서 읽는 행. 모든 빠른 시작이 이를 사용합니다.
TsgcPivotSource_DataSet	`System.Data.DataTable`. 행이 바뀌면 다시 로드됩니다.
TsgcPivotSource_Table	모든 **IsgcUITable**: sgcGrid .NET의 그리드와 행 저장소, 다른 피벗 엔진 또는 그 결과. 테이블 버전이 바뀌면 다시 로드됩니다.
LoadFromTable	**IsgcUITable**의 일회성 복사본. 예를 들어 드릴다운의 원본 행입니다.

소스는 **PivotGrid**를 설정하면 그리드의 엔진을, **Engine**을 설정하면 헤드리스 엔진을 채웁니다. 둘 다 설정하면 **Engine**이 우선합니다. 로드할 때마다 행이 하나의 로드 구간에서 교체되므로 엔진은 한 번만 다시 계산하며 레이아웃을 유지합니다.

DataTable

```
using System;
using System.Data;
using esegece.sgcPivotGrid;

namespace PivotDataTable
{
    internal static class SalesTable
    {
        public static DataTable CreateTable()
        {
            DataTable oTable = new DataTable("Sales");
            oTable.Columns.Add("Region", typeof(string));
            oTable.Columns.Add("Date", typeof(DateTime));
            oTable.Columns.Add("Sales", typeof(decimal));
            oTable.Columns.Add("Notes", typeof(string));
            oTable.Rows.Add("North", new DateTime(2025, 3, 1), 120.50m, "");
            oTable.Rows.Add("South", new DateTime(2025, 7, 9), 80m, "");
            oTable.Rows.Add("North", new DateTime(2026, 1, 15), 42.25m, "");
            return oTable;
        }

        public static void Run()
        {
            DataTable oTable = CreateTable();
            TsgcPivotEngine oEngine = new TsgcPivotEngine();
            using (TsgcPivotSource_DataSet oSource = new TsgcPivotSource_DataSet())
            {
                // only these columns, in this order; empty loads every column
                oSource.Columns.Assign(new[] { "Region", "Date", "Sales" });
                // reload after every committed change of the DataTable
                oSource.LiveUpdates = true;
                // Engine for a headless engine, or PivotGrid = a WinForms, WPF or Avalonia pivot grid
                oSource.Engine = oEngine;
                oSource.DataSet = oTable;
                oSource.Active = true;

                oEngine.BeginUpdate();
                try
```

```
                {
                    oEngine.AddField("Region", TsgcPivotArea.pvaRow);
                    oEngine.AddField("Date", TsgcPivotArea.pvaColumn, TsgcPivotDateGrouping.pvdYear);
                    oEngine.AddDataField("Sales", TsgcPivotAggregation.pvfSum);
                }
                finally
                {
                    oEngine.EndUpdate();
                }

                // a committed row reloads the engine, which recalculates once
                oTable.Rows.Add("East", new DateTime(2026, 2, 2), 10m, "");
                Console.WriteLine("{0} source rows, {1} columns", oEngine.SourceRowCount(),
                    oEngine.SourceColumnCount());
            }
        }
    }
}
```

이 예제는 `4 source rows, 3 columns`를 출력합니다. Notes 열은 **Columns**에 없고, 바인딩 후에 추가된 행이 엔진을 다시 로드했습니다. DataTable 열의 형식이 피벗 열의 형식을 정합니다.

DataTable 열 형식	피벗 열 형식
`byte` , `short` , `int` , `long` 및 기타 정수 형식	`pvtInteger`
`float` , `double` , `decimal`	`pvtFloat`
`DateTime` , `DateTimeOffset` , `TimeSpan`	`pvtDateTime`
`bool`	`pvtBoolean`
이진 열(`byte[]` 와 `char[]` 를 제외한 기타 배열)	로드되지 않습니다.
그 외 모든 것	`pvtText` , 인바리언트 문화권 사용.

LiveUpdates를 사용하면 커밋된 변경으로 엔진이 다시 로드됩니다. 추가, 변경, 롤백, 삭제된 행과 비워진 테이블이 해당합니다. 편집 모드의 행은 현재 값을 로드하므로 로드가 보류 중인 편집을 커밋하지 않으며, 새 행은 테이블에 추가될 때까지 무시됩니다. **Load**는 언제든 로드하고 **Refresh**는 **Active**가 true일 때 다시 로드합니다.

IsgcUITable

```
using esegece.sgcPivotGrid;

namespace PivotTableSource
{
    internal static class PivotSources
    {
        // a second pivot over the source rows behind one cell of the first
        public static TsgcPivotEngine PivotOfCell(TsgcPivotEngine aEngine, int aRowItem, int aDataCol)
        {
            TsgcPivotEngine oDetail = new TsgcPivotEngine();
            // DrillDown returns the rows as an IsgcUITable; LoadFromTable copies them once
            oDetail.LoadFromTable(aEngine.DrillDown(aRowItem, aDataCol));
            oDetail.BeginUpdate();
            try
            {
                oDetail.AddField("Channel", TsgcPivotArea.pvaRow);
                oDetail.AddDataField("Sales", TsgcPivotAggregation.pvfCount);
            }
            finally
            {
                oDetail.EndUpdate();
```

```
            }
            return oDetail;
        }

        // a pivot that follows any IsgcUITable: here the result of another pivot,
        // reloaded after every recalculation of aEngine
        public static TsgcPivotSource_Table Follow(TsgcPivotEngine aEngine, TsgcPivotEngine aTarget)
        {
            TsgcPivotSource_Table oSource = new TsgcPivotSource_Table();
            oSource.LiveUpdates = true;
            oSource.Engine = aTarget;
            oSource.Table = aEngine;
            oSource.Active = true;
            return oSource;
        }
    }
}
```

PivotOfCell은 상세 창을 위해 한 셀 뒤에 있는 행에 대해 두 번째 피벗을 만듭니다. **Follow**는 피벗의 피벗을 만듭니다. 엔진은 자신의 결과에 대한 **IsgcUITable**이며, 행 필드마다 한 열, 데이터 열마다 한 열을 가지고, 열 멤버와 측정값에서 이름을 따옵니다. 예를 들면 `Online | Sum of Sales`입니다. 첫 번째 엔진이 다시 계산될 때마다 테이블 버전이 올라가고 두 번째가 다시 로드됩니다. 같은 **Table** 속성은 GridToPivotToChart 데모처럼 사용자가 행을 편집하는 sgcGrid .NET 그리드도 받습니다.

표시 옵션 바인딩

Avalonia에서는 그리드의 모든 표시 옵션(**RowLayout**, **RowHeight**, **HeaderHeight**, **IndentWidth**, **MinColumnWidth**, **MaxColumnWidth**, **AutoColumnWidths**, **GridLines**, **AlternateRows**, **TotalsBold**, **ShowExpandButtons**, **Theme**, **AccentColor**, **FontSize**, **BorderThickness**, **BorderBrush**)이 스타일 속성이고, **SelectedRow** 와 **SelectedCol**은 양방향 직접 속성입니다. 이들은 바인딩, 스타일, XAML 값을 받습니다. Avalonia 빠른 시작을 참조하십시오. **Renderer.Options**에 코드로 가한 변경은 속성으로 다시 복사됩니다. Avalonia 필드 목록도 **Theme**, **AccentColor**, **FontSize**, **ShowFilterArea**, **DeferUpdate**를 스타일 속성으로 가집니다.
WPF와 WinForms에서는 옵션이 CLR 속성입니다. 코드로 설정하거나 WPF에서는 XAML 특성으로 설정합니다.

관련 항목

- .NET 개요
- .NET 클래스
- .NET과 Delphi의 차이점
- TsgcPivotSource_DataSet
- TsgcPivotSource_Table
- 차트 연동

.NET 헤드리스 렌더링

엔진과 렌더러는 창이 필요 없으므로 Windows, Linux, macOS의 서비스, 웹 애플리케이션, 콘솔 작업이 피벗을 만들어 PNG, SVG, CSV, JSON, HTML로 기록할 수 있습니다. 이 예제는 10,000개의 판매 행을 로드하고, 필터 필드로 온라인 채널만 남기고, 결과를 출력하고, 셀 뒤에 있는 원본 행을 읽고, 다섯 개의 파일을 기록합니다.

예제

```
using System;
using System.IO;
using esegece.sgcPivotGrid;
using esegece.sgcUICore;

namespace PivotHeadless
{
    internal static class Program
    {
        private static readonly string[] Regions = { "North", "South", "East", "West" };
        private static readonly string[] Channels = { "Online", "Retail" };

        private static string DoLabel(TsgcPivotEngine aEngine, TsgcPivotResult aResult, TsgcPivotAxisKind aAxis,
            int aIndex)
        {
            TsgcPivotItem vItem = aResult.Item(aAxis, aIndex);
            if (vItem.Kind == TsgcPivotItemKind.pvkGrandTotal)
                return aEngine.GrandTotalCaption;
            return aResult.NodeLabel(aAxis, vItem.Node);
        }

        private static void Main(string[] args)
        {
            string vOutDir = args.Length > 0 ? args[0] : AppContext.BaseDirectory;
            TsgcPivotEngine oEngine = new TsgcPivotEngine();

            // 1. the source rows: define the columns, then append the rows
            Random oRandom = new Random(2026);
            oEngine.BeginLoad(true);
            try
            {
                oEngine.DefineColumn("Region", TsgcPivotDataType.pvtText);
                oEngine.DefineColumn("Channel", TsgcPivotDataType.pvtText);
                oEngine.DefineColumn("Date", TsgcPivotDataType.pvtDateTime);
                oEngine.DefineColumn("Sales", TsgcPivotDataType.pvtFloat);
                for (int i = 0; i < 10000; i++)
                {
                    string vRegion = Regions[oRandom.Next(Regions.Length)];
                    string vChannel = Channels[oRandom.Next(Channels.Length)];
                    DateTime vDate = new DateTime(2024, 1, 1).AddDays(oRandom.Next(731));
                    double vSales = Math.Round(oRandom.NextDouble() * 500, 2);
                    oEngine.AppendRow(vRegion, vChannel, vDate, vSales);
                }
            }
            finally
            {
                oEngine.EndLoad();
            }

            // 2. the layout: the regions on the rows, the years on the columns, online sales only
            oEngine.BeginUpdate();
            try
            {
                oEngine.AddField("Region", TsgcPivotArea.pvaRow);
```

```
            oEngine.AddField("Date", TsgcPivotArea.pvaColumn, TsgcPivotDateGrouping.pvdYear).Caption = "Year";
            oEngine.AddField("Channel", TsgcPivotArea.pvaFilter).FilterValues.Add("Online");
            oEngine.AddDataField("Sales", TsgcPivotAggregation.pvfSum).DisplayFormat = "#,##0.00";
        }
        finally
        {
            oEngine.EndUpdate();
        }

        // 3. read the snapshot between Lock and Unlock
        oEngine.Lock();
        try
        {
            TsgcPivotResult oResult = oEngine.Result;
            Console.Write("{0,-12}", "Region");
            for (int c = 0; c < oResult.ItemCount(TsgcPivotAxisKind.pvxColumns); c++)
                Console.Write("{0,14}", DoLabel(oEngine, oResult, TsgcPivotAxisKind.pvxColumns, c));
            Console.WriteLine();
            for (int r = 0; r < oResult.ItemCount(TsgcPivotAxisKind.pvxRows); r++)
            {
                Console.Write("{0,-12}", DoLabel(oEngine, oResult, TsgcPivotAxisKind.pvxRows, r));
                for (int c = 0; c < oResult.DataColCount(); c++)
                    Console.Write("{0,14}", oResult.CellText(r, c));
                Console.WriteLine();
            }
        }
        finally
        {
            oEngine.Unlock();
        }

        // 4. drill-down: the source rows behind the first cell
        int[] vRows;
        int vCount = oEngine.GetSourceRows(0, 0, out vRows);
        Console.WriteLine("The first cell adds up {0} source rows", vCount);

        // 5. the whole view as PNG and SVG, the result as CSV, JSON and HTML
        TsgcPivotGridRenderer oRenderer = new TsgcPivotGridRenderer();
        oRenderer.Options.AlternateRows = true;
        TsgcPivotFormatRule oRule = oRenderer.FormatRules.Add();
        oRule.Kind = TsgcPivotFormatRuleKind.pvrColorScale;
        oRule.MinColor = 0;
        oRule.MaxColor = sgcUICore_Theme.sgcUIARGB(0xFF, 0x63, 0xBE, 0x7B);
        oRule.Scope = TsgcPivotFormatScopes.pveCells;
        TsgcPivotRenderState vState = sgcPivotGrid_Render.sgcPivotRenderState();
        // SaveToPNG is an extension method of the esegece.sgcPivotGrid.Rendering package
        oRenderer.SaveToPNG(oEngine, vState, Path.Combine(vOutDir, "pivot.png"));
        oRenderer.SaveToSVG(oEngine, vState, Path.Combine(vOutDir, "pivot.svg"));
        oEngine.SaveToCSV(Path.Combine(vOutDir, "pivot.csv"));
        oEngine.SaveToJSON(Path.Combine(vOutDir, "pivot.json"));
        oEngine.SaveToHTML(Path.Combine(vOutDir, "pivot.html"));
    }
  }
}
```

콘솔 출력:

```
Region                2024          2025   Grand Total
East            145,862.61    155,105.09    300,967.70
North           162,608.24    155,080.86    317,689.10
South           154,901.83    155,794.10    310,695.93
West            156,547.53    157,665.06    314,212.59
Grand Total     619,920.21    623,645.11  1,243,565.32
The first cell adds up 580 source rows
```

참고

- 셀은 행 항목과 데이터 열로 지정합니다. 측정값이 여러 개이면 한 열 항목의 데이터 열은 나란히 놓입니다. 데이터 열은 열 항목에 **MeasureCount**를 곱하고 측정값 번호를 더한 값입니다.
- **TsgcPivotRenderState**는 스크롤 위치, 선택, 열 너비를 담고 있으며, `sgcPivotGrid_Render.sgcPivotRenderState()`는 기본 상태를 반환합니다. 기본값인 너비와 높이 0은 한 화면 대신 전체 뷰를 그립니다.
- **SaveToSVG**는 esegece.sgcPivotGrid에 포함되어 있으며 네이티브 종속성이 없습니다. **SaveToPNG**는 SkiaSharp로 그리는 esegece.sgcPivotGrid.Rendering의 확장 메서드입니다.
- 그리드에도 같은 메서드가 있습니다. `Grid.SaveToPNG(fileName)`은 그리드의 옵션과 서식 규칙을 반영한 전체 뷰를 기록합니다.
- CSV 내보내기는 원시 값을 전체 정밀도로 기록하고, HTML 내보내기는 서식이 적용된 셀 텍스트를 기록합니다.

관련 항목

- .NET 개요
- .NET 클래스
- TsgcPivotGridRenderer
- 내보내기
- 빠른 시작 헤드리스
- .NET 데모

.NET 데모

데모는 `.net\demos\PivotGrid`에 있으며, 데모마다 프로젝트, `Program.cs`, README.md가 있는 폴더가 하나씩 있습니다. 모든 데모에는 `--test` 스위치가 있어 화면 밖에서 자체 점검을 실행하고, PASS와 FAIL 줄을 기록하며, 실패한 점검의 수를 종료 코드로 반환합니다. Delphi 데모의 `/test` 스위치와 같습니다.

데모	호스트	보여 주는 내용
SalesPivot	WinForms	필드 목록이 있는 생성된 100,000개의 판매 행. 행에는 Region과 Country, 열에는 Date 열을 Year, Quarter, Month로 배치하고, 채널을 필터로 사용하며, 값으로 정렬, 색상 스케일, 확장과 축소, CSV, HTML, PNG 내보내기를 보여 줍니다.
SalesPivotWpf	WPF	같은 피벗을 **TsgcWpfPivotGrid**에서.
SalesPivotAvalonia	Avalonia	같은 피벗을 Windows, Linux, macOS용 **TsgcAvaloniaPivotGrid**에서.
GridToPivotToChart	WinForms	Registered 에디션에서 편집할 수 있는 sgcGrid .NET 그리드가 **TsgcPivotSource_Table**을 통해 피벗에 데이터를 공급하고, 피벗 결과는 **IsgcUITable**을 통해 sgcCharts .NET 차트에 데이터를 공급합니다.
MillionRows	WinForms	생성된 100만 행과 세 가지 측정값(합계, 100,000명 고객의 고유 개수, 표준 편차). 로드와 각 재계산에 걸린 시간을 표시합니다.
HeadlessReport	콘솔	CSV 파일을 읽거나, 없으면 먼저 생성된 5,000건의 주문이 담긴 샘플 orders.csv를 기록합니다. 그리고 창 없이 **TsgcPivotGridRenderer**로 피벗을 PNG와 SVG로, 엔진으로 HTML, CSV, JSON으로 기록합니다. Windows, Linux, macOS에서 실행됩니다.

데모 빌드

제품 트리 안에서는 데모가 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip은 기본적으로 Community 에디션의 NuGet 패키지를 사용합니다. 어떤 데모도 고급 기능을 사용하지 않으므로 모든 데모가 두 에디션에서 빌드되고 실행됩니다. Registered 패키지로 데모를 빌드하려면 **SgcPivotGridEdition**을 설정하십시오.

```
dotnet build SalesPivot\SalesPivot.csproj -p:SgcPivotGridEdition=Registered
dotnet run --project HeadlessReport\HeadlessReport.csproj -- --test
```

GridToPivotToChart는 같은 에디션의 sgcGrid와 sgcCharts 패키지도 복원합니다.

관련 항목

- .NET 개요
- NuGet에서 설치
- WinForms 빠른 시작
- .NET 헤드리스 렌더링
- 데모

문제 해결

그리드가 비어 있습니다

데이터가 도착했는지 확인하십시오. 소스가 Active가 아니거나 DataSet, Table, PivotGrid가 없으면 엔진의 SourceRowCount가 0입니다. 이름도 확인하십시오. FieldName이 어떤 소스 열과도 일치하지 않는 행, 열, 필터 필드는 무시되고, 알 수 없는 이름의 데이터 필드는 비어 있습니다. 이름은 대소문자를 구분하지 않고 비교하며, SourceColumnName은 불러온 열을 나열합니다. AutoRecalculate가 False이면 Recalculate를 호출하십시오. 소스 행이 하나도 남지 않는 필터도 빈 피벗을 만듭니다.

피벗이 데이터셋을 따라가지 않습니다

TsgcPivotSource_DataSet은 Active와 LiveUpdates가 True인 동안, 데이터셋이 탐색 상태에서의 데이터 변경(Post, Append, Delete, Refresh, First, Last)을 알릴 때 다시 불러옵니다. Next, Prior, MoveBy는 다시 불러오지 않으며, 편집은 Post된 후에만 다시 불러옵니다.

불러오기가 느립니다

BeginLoad와 EndLoad 브래킷 밖의 AppendRow는 행마다 다시 계산합니다. 브래킷 안에서 불러오십시오. 성능을 참조하십시오.

EsgcPivotError: The pivot layout needs more than ... cells

레이아웃이 MaxCells보다 많은 셀을 만듭니다. 대개 멤버가 많은 필드가 행에 있고 다른 필드가 열에 있기 때문입니다. 날짜를 그룹화하거나, 필드를 필터로 옮기거나, Top N을 사용하거나, MaxCells를 늘리십시오.

EsgcPivotError: Calculated field ...

해당 계산 필드의 식에 구문 오류가 있거나 계산 필드가 서로를 순환 참조합니다. 메시지에 필드 이름이 나옵니다. 아직 불러오지 않은 필드 이름은 오류가 아니며, 그 열이 도착할 때까지 계산 필드는 비어 있습니다. 계산 필드를 참조하십시오.

EsgcPivotError: Invalid pivot layout

LoadLayoutFromString 또는 LoadLayoutFromFile이 잘못된 JSON 또는 개체가 아닌 루트를 받았습니다. 레이아웃은 바뀌지 않습니다.

Undeclared identifier CalculatedFields, TopN, ShowAs 또는 SaveLayoutToString

Community 에디션에는 계산 필드, 값 표시 형식, Top N, 레이아웃 저장이 빠져 있으므로, 이 멤버들은 OthersCaption, TopNOthers, GetExpandState, SetExpandState와 함께 존재하지 않습니다. 이 속성들을 저장한 폼은 Community 빌드에서 불러올 수 없습니다. 에디션을 참조하십시오.

빈 셀에 아무것도 표시되지 않습니다

엔진의 EmptyText를, 예를 들어 대시로 설정하십시오. 그리드, 렌더러, HTML 내보내기, ResultTable이 이를 표시하고, OnGetCellText가 이를 받아 대체할 수도 있습니다. SaveToCSV는 빈 필드를, SaveToJSON은 null을 기록하므로 내보낸 데이터는 숫자로 유지됩니다.

합계가 셀의 합이 아닙니다

합계는 소스 행에서 계산됩니다. 평균은 행의 평균이고, 고유 개수는 각 값을 한 번만 셉니다. 값 필터나 Top N 이후에는 합계에 남은 멤버만 포함됩니다. 계산 필드는 행마다 평가된 다음 집계됩니다. 집계와 합계를 참조하십시오.

날짜가 그룹화되지 않거나 숫자가 텍스트처럼 정렬됩니다

DateGrouping은 날짜 열에서만 동작하고, 레이블 정렬이 숫자를 값으로 비교하는 것은 숫자 열에서뿐입니다. 첫 번째 값이 문자열이었던 열은 텍스트 열입니다. AppendRow에 TDateTime과 숫자 값을 전달하거나, 첫 행 전에 DefineColumn(aName, aType)으로 형식을 고정하십시오.

드릴다운 뷰가 비어 있거나 종료 시 접근 위반이 발생합니다

TsgcPivotSourceView는 소스 데이터가 바뀐 후에는 행이 없고 엔진의 저장소를 읽으므로 엔진보다 먼저 해제하십시오. 드릴다운을 참조하십시오.

다른 스레드가 Result를 읽을 때 잘못된 값이나 예외가 발생합니다

Result는 엔진의 Lock과 Unlock 사이에서만 읽으십시오. 재계산은 새 스냅샷을 게시하고 이전 것을 해제합니다.

C++Builder: TsgcPivotEngine 또는 TsgcPivotGridRenderer에 맞는 생성자가 없습니다

생성된 헤더는 이 생성자들의 기본 인수를 유지하지 않습니다. NULL을 명시적으로 전달하십시오: `new TsgcPivotEngine(NULL)`과 `new TsgcPivotGridRenderer(NULL, NULL)`.

TsgcFMXPivotGrid가 없거나 비어 있습니다

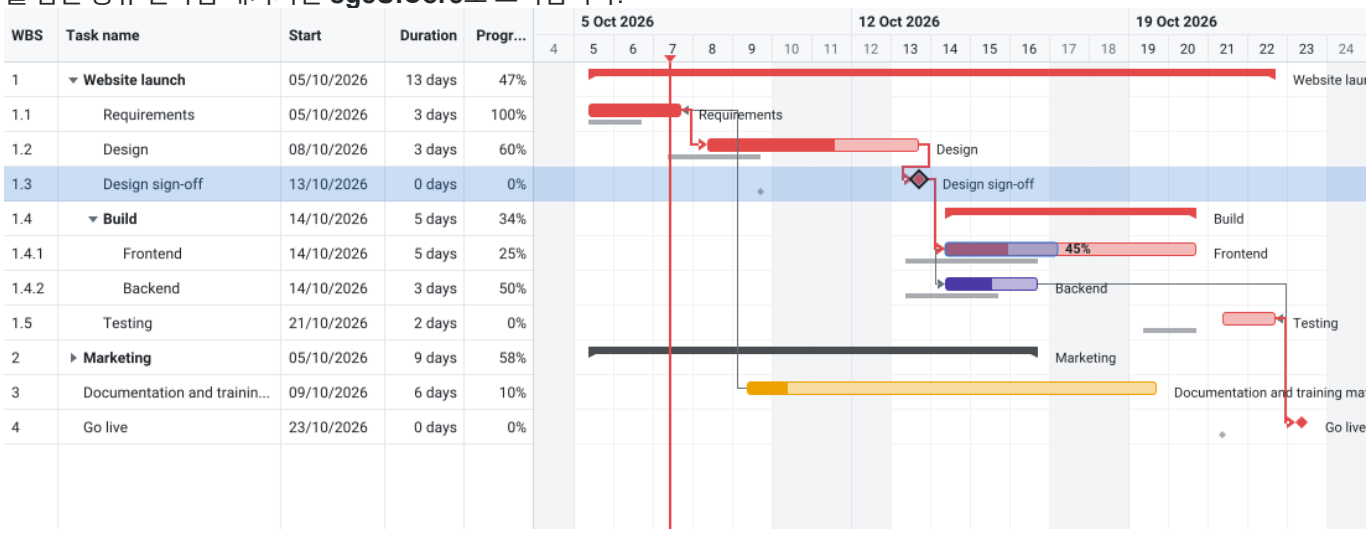
TsgcFMXPivotGrid와 TsgcFMXPivotFieldList는 sgcVer.inc가 SGC_PIVOTGRID_FMX를 정의하는 Delphi XE6 이상에 있습니다. 더 오래된 컴파일러에서는 sgcPivotGrid_FMX 유닛이 이들 없이 컴파일됩니다. 디자인 타임에는 FireMonkey 폼이 열려 있는 동안 팔레트의 SGC PivotGrid 페이지에 있습니다. 플랫폼을 참조하십시오.

필드 목록에 필드가 표시되지 않습니다

필드 목록의 PivotGrid를 설정하거나, 그리드가 없는 엔진에는 Engine을 설정하십시오. 목록은 엔진의 소스 열을 표시하므로 데이터를 불러오면 채워집니다.

소개

sgcPlanner는 VCL과 FireMonkey용 Delphi 및 C++Builder 애플리케이션에 일정 관리와 프로젝트 계획 기능을 추가합니다. 데이터 객체 하나인 **TsgcPlannerData**가 항목, 리소스, 상태, 종속성, 근무 달력을 보관하고, 세 가지 종류의 뷰가 이를 동시에 보여 줍니다. **TsgcVCLScheduler**와 **TsgcFMXScheduler**(일, 주 중, 주, 월, 안건 목록, 리소스 Timeline), **TsgcVCLGantt**와 **TsgcFMXGantt**(막대, 종속성, 자동 일정 계산, 크리티컬 패스, 기준선을 갖춘 작업 트리), **TsgcVCLKanban**과 **TsgcFMXKanban**(상태 열, 스윔레인, WIP 제한을 갖춘 보드)입니다. 카드를 다른 열로 끌어 놓으면 Gantt와 Scheduler에 새 상태가 즉시 표시됩니다.
데이터는 코드에서, **TsgcPlannerSource_DataSet**을 통한 **TDataSet**에서, JSON 파일에서, **TsgcPlannerICalendar**를 통한 iCalendar(.ics) 파일에서, 또는 **TsgcPlannerMSProject**를 통한 Microsoft Project XML 파일에서 가져옵니다. **TsgcPlannerRenderer**는 서버나 Linux64에서 창 없이 어떤 뷰든 PNG와 SVG로 그립니다.
sgcPlanner는 eSeGeCe 비주얼 스위트의 일부입니다. 캔버스, 래스터라이저, 글꼴, SVG 작성기, 밝은 테마와 어두운 테마를 담은 공유 런타임 패키지인 **sgcUICore**로 그려집니다.



에디션

sgcPlanner는 Registered, Trial, Community 에디션으로 제공됩니다. Community 에디션은 세 가지 뷰, 끌어서 놓기, 요약, 마일스톤, 진행률, DataSet 소스, JSON, 헤드리스 출력을 제공하며, 반복, 종속성 화살표를 포함한 자동 일정 계산, 크리티컬 패스, 기준선, 리소스 Timeline, 스윔레인, WIP 제한, iCalendar, MS Project XML은 제외됩니다. 에디션을 참조하십시오.

시작하는 곳

- 아키텍처: 데이터 객체, 변경 알림, 뷰, 플랫폼.
- VCL 빠른 시작, FMX 빠른 시작, C++Builder 빠른 시작: 몇 분 만에 동기화된 세 가지 뷰를 만듭니다.
- TsgcPlannerData: 항목, 리소스, 상태, 종속성, 근무 달력.
- 뷰: TsgcVCLScheduler, TsgcVCLGantt, TsgcVCLKanban 및 FireMonkey 대응 컴포넌트.
- 비주얼이 아닌 컴포넌트: TsgcPlannerSource_DataSet, TsgcPlannerICalendar, TsgcPlannerMSProject, TsgcPlannerRenderer.
- 설치와 플랫폼: 패키지, 폴더, 지원되는 IDE와 플랫폼.
- 데모: 여덟 개의 Delphi 데모와 각각이 보여 주는 내용.
- 기능: 공유 데이터, 동기화된 뷰, 근무 달력, 반복, 시간 눈금과 확대/축소, 끌어서 놓기, 종속성, 자동 일정 계산, 크리티컬 패스, 기준선, 스윔레인과 WIP 제한, DataSet 바인딩, JSON 저장, iCalendar, MS Project XML, 헤드리스 렌더링, 테마, 에디션별 기능.
- 문제 해결: 흔한 문제와 그 원인.

아키텍처

sgcPlanner는 데이터를 뷰와 분리하고, 뷰를 프레임워크와 분리합니다. 데이터 객체, 엔진, 뷰 모델은 RTL과 sgcUICore만 사용합니다. VCL과 FMX 컨트롤은 뷰 모델을 둘러싼 얇은 호스트이고, 헤드리스 렌더러는 같은 뷰 모델을 사용하므로 PNG는 폼이 보여 주는 것과 픽셀 단위로 일치합니다.

데이터 흐름

1. **소스.** 코드가 TsgcPlannerData를 직접 채우거나, TsgcPlannerSource_DataSet가 TDataSet에서 로드하고 편집을 되씁니다. LoadFromJSON은 저장된 프로젝트를 읽고, TsgcPlannerICalendar는 .ics 파일을, TsgcPlannerMSProject는 Microsoft Project XML을 가져오고 내보냅니다.
2. **데이터.** TsgcPlannerData는 Items, Resources, Statuses, Dependencies, Calendar의 다섯 부분을 가진 TComponent입니다. Id 인덱스, 부모와 자식 구조, 선택, 변경 때마다 증가하는 Version 번호를 유지합니다.
3. **엔진.** 근무 달력은 근무 분을 세고, 반복 엔진은 RRULE 규칙을 발생 항목으로 전개하며, 일정 계산 엔진은 작업을 선행 작업 뒤에 배치하고 여유 시간을 계산하고 크리티컬 패스를 표시합니다. 뒤의 두 가지는 Registered 에디션에 포함됩니다.
4. **뷰 모델.** 각 뷰는 스스로를 데이터의 리스너로 등록합니다. 변경이 있으면 필요한 것만 다시 만듭니다: 선택 변경은 오버레이를 다시 그리고, 항목 변경은 다음 프레임에 레이아웃을 다시 계산하며, 구조 변경은 행, 열, 레인을 다시 만듭니다.
5. **출력.** 뷰 모델은 sgcUICore의 TsgcUICanvas에 그려집니다. 여섯 컨트롤은 이를 스크롤, 마우스, 키보드, 터치와 함께 폼에 보여 주며, TsgcPlannerRenderer는 PNG와 SVG로 그립니다.

클래스

클래스	유닛	역할
TsgcPlannerData	sgcPlanner_Data	공유 데이터: 항목, 리소스, 상태, 종속성, 달력, 선택, 알림, JSON.
TsgcPlannerItem	sgcPlanner_Data	작업, 카드 또는 이벤트: 날짜, 상태, 진행률, 부모, 리소스, 태그, 반복, 제약, 기준선.
TsgcPlannerCalendar	sgcPlanner_Calendar	근무일, 근무 시간, 휴일, 근무 분 연산.
TsgcPlannerTimeScale	sgcPlanner_TimeScale	Gantt와 Timeline의 시간 축: 확대/축소, 눈금 너비, 눈금, 비근무 띠, 스냅.
TsgcVCLScheduler, TsgcFMXScheduler	sgcPlanner_Scheduler_VCL, sgcPlanner_Scheduler_FMX	달력 뷰: 일, 주중, 주, 월, 안건 목록, 리소스 Timeline.
TsgcVCLGantt, TsgcFMXGantt	sgcPlanner_Gantt_VCL, sgcPlanner_Gantt_FMX	종속성, 크리티컬 패스, 기준선을 갖춘 작업 트리와 막대 차트.
TsgcVCLKanban, TsgcFMXKanban	sgcPlanner_Kanban_VCL, sgcPlanner_Kanban_FMX	보드: 상태 열, 스윔레인, WIP 제한, 카드.
TsgcPlannerSource_DataSet	sgcPlanner_Source_DataSet	TDataSet에서 데이터를 로드하고, 그 변경을 따라가며, 편집을 되씁니다.
TsgcPlannerICalendar	sgcPlanner_ICal	RFC 5545 VEVENT 가져오기와 내보내기. Registered 에디션.
TsgcPlannerMSProject	sgcPlanner_MSProject	Microsoft Project XML(MSPDI) 가져오기와 내보내기. Registered 에디션.

TsgcPlannerRenderer	sgcPlanner_Renderer	Scheduler, Gantt 또는 Kanban 뷰를 창 없이 캔버스, PNG 또는 SVG로 그립니다.
TsgcPlannerScheduleEngine	sgcPlanner_AutoSchedule	정방향 및 역방향 처리, 여유 시간, 크리티컬 패스, 기준선, 순환 감지. Registered 에디션.

변경 알림

- 데이터의 모든 setter가 알립니다. 변경 묶음을 BeginUpdate와 EndUpdate로 감싸십시오: 리스너와 OnChange는 가장 바깥쪽 EndUpdate 이후에 한 번만 발생하며, 변경의 종류(pckItems, pckStructure, pckResources, pckStatuses, pckDependencies, pckCalendar, pckSelection, pckSchedule), 정확히 한 항목이 바뀐 경우 그 항목의 Id, 그리고 Version을 전달합니다.
- 뷰는 리스너이며 데이터를 읽기만 합니다. OnChange 핸들러는 데이터를 다시 변경해도 됩니다: 중첩된 변경은 현재 라운드 뒤에 한 라운드 더 도착하며 최대 8라운드까지이고, 그 뒤에는 EsgcPlannerError가 발생합니다.
- 마우스로 한 편집은 놓을 때마다 데이터의 OnItemChanging을 한 번, 변경될 필드와 함께 발생시키므로 애플리케이션이 이를 거부할 수 있습니다. 그다음 편집이 한 번의 업데이트로 커밋되므로 놓기는 바인딩된 모든 뷰에 알림 한 번을 줍니다.
- 같은 데이터의 둘 이상의 뷰는 보조를 맞춥니다: SyncSelection이 있으면 데이터의 SelectedId인 선택도 공유됩니다.

시간

모델은 시간을 분 수인 Int64 TsgcPlannerTime으로 유지하므로 연산이 정확하고 초는 버려집니다. 게시된 속성은 TDateTime이며 각각 TsgcPlannerTime 대응 속성이 있습니다: Start와 StartTime, Finish와 FinishTime, ProjectStart와 ProjectStartTime. sgcPlanner_DateUtils의 sgcPlannerTimeOf와 sgcPlannerDateTimeOf가 둘 사이를 변환합니다. 항목의 Finish는 포함되지 않습니다(배타적): 10월 5일의 종일 항목은 10월 6일 자정에 끝납니다. 뷰는 함수 변수 sgcPlannerNow에서 현재 시간을 읽으며, 테스트는 이를 교체하여 시계를 고정할 수 있습니다.

플랫폼

대상	지원
Delphi VCL	Delphi 7부터 Delphi 13까지, Win32와 Win64.
Delphi FireMonkey	Delphi XE6부터 Delphi 13까지(SGC_PLANNER_FMX), FireMonkey가 지원하는 모든 플랫폼.
Linux64	프레임워크에 독립적인 유닛은 Delphi 10.2 이상(SGC_PLANNER_LINUX): 데이터, 엔진, JSON, iCalendar, MS Project XML, 헤드리스 렌더러.
C++Builder	C++Builder 2007부터 C++Builder 13까지, 같은 컴포넌트. C++Builder 빠른 시작을 참조하십시오.

Delphi 7과 Delphi 2007은 Ansi 문자열을 사용하므로 iCalendar와 MS Project XML 컴포넌트는 파일의 UTF-8 텍스트를 시스템 코드 페이지로 변환하며, 그 코드 페이지 밖의 문자는 손실됩니다.

성능

개발 중 Win64 릴리스 빌드를 1920x1080, 배율 1에서 측정한 값입니다. Win32는 약 1.5배에서 1.7배 느립니다.

시나리오	Win64
링크 15,000개와 휴일 30개를 가진 작업 10,000개(요약 1,000개, 3단계)의 자동 일정 계산, 크리티컬 패스 포함	11 ms

링크 15,000개를 가진 작업 10,000개를 스크롤하는 동안의 Gantt 프레임	11 ms
8개 열과 6개 레인에 카드 5,000개의 Kanban 레이아웃	5 ms
시간 지정 항목 2,000개의 Scheduler 주 뷰, 레이아웃	1 ms
TsgcPlannerSource_DataSet으로 10,000행 로드	26 ms
링크 15,000개를 가진 항목 10,000개의 JSON 저장과 로드	48 ms와 56 ms
.ics 파일에서 시간대가 있는 VEVENT 5,000개 가져오기	84 ms
MS Project XML에서 링크 15,000개를 가진 작업 10,000개 가져오기	264 ms

관련 항목

- 에디션
- TsgcPlannerData
- TsgcPlannerRenderer
- 플랫폼
- 공유 데이터
- 동기화된 뷰

에디션

sgcPlanner는 모든 에디션에서 같은 소스로 빌드됩니다.

에디션	포함 내용
Registered	모든 것: VCL과 FMX용 세 가지 뷰, 데이터 객체, 근무 달력, 끌어서 놓기, 요약, 마일스톤, 진행률, DataSet 소스, JSON, 헤드리스 PNG 및 SVG 출력, 그리고 아래에 나열된 고급 기능. SGC Planner 팔레트 페이지에는 컴포넌트 11개가 표시됩니다.
Trial	Registered와 같은 기능. SGC_TRIAL로 컴파일한 빌드는 빌드에 명시된 릴리스 기간이 지난 뒤 컴포넌트가 생성될 때 체험판이라는 메시지를 표시합니다. 그 외에는 아무것도 달라지지 않습니다.
Community	고급 기능을 제외한 모든 것: 리소스별 그룹화가 있는 Scheduler의 일, 주중, 주, 월, 안건 뷰, 작업 트리, 요약, 마일스톤, 진행률을 갖춘 Gantt, 상태 열을 갖춘 Kanban 보드, 모든 뷰의 끌어서 놓기, DataSet 소스, JSON, 렌더러. 팔레트에는 TsgcPlannerICalendar와 TsgcPlannerMSProject를 제외한 컴포넌트 9개가 표시되며 체험판 안내는 없습니다.

고급 기능

고급 기능은 sgcVer.inc가 Community 에디션에서 끄는 SGC_PLANNER_ADVANCED 정의 아래에서 컴파일됩니다. 여섯 유닛은 Community 배포판에 포함되지 않습니다: sgcPlanner_Recurrence, sgcPlanner_AutoSchedule, sgcPlanner_Scheduler_Timeline, sgcPlanner_XML, sgcPlanner_ICal, sgcPlanner_MSProject. 뷰의 속성은 모든 에디션에서 게시되므로 Community에서는 유지되지만 효과가 없습니다:

기능	Community 에디션에서
반복	GetOccurrences는 반복 항목을 자신의 Start에 한 번만 반환합니다. RecurrenceRule과 RecurrenceExceptions는 여전히 보관되고 저장됩니다. Scheduler의 OnRecurrenceEdit과 OnRecurrenceDelete는 발생하지 않습니다.
종속성과 자동 일정 계산	링크는 보관되고 저장되지만 Gantt는 화살표를 그리지 않으며 끌어서 링크 만들기도 없습니다. ShowDependencies와 AutoSchedule은 효과가 없고, Gantt의 Schedule과 SetBaseline은 EsgcPlannerError를 발생시킵니다.
크리티컬 패스와 기준선	ShowCriticalPath와 ShowBaseline은 효과가 없습니다. BaselineStart와 BaselineFinish는 보관만 됩니다.
리소스 Timeline	Scheduler의 psvTimeline 뷰는 가운데 정렬된 안내 메시지를 그립니다.
스윔레인과 WIP 제한	SwimlaneBy는 pslNone처럼 동작합니다. WIPLimit는 배지를 표시하지 않으며 카드를 막지 않습니다.
iCalendar와 MS Project XML	TsgcPlannerICalendar와 TsgcPlannerMSProject는 존재하지 않습니다.

에디션은 sgcVer.inc에서 컴파일 시점에 선택합니다. SGC_COMMUNITY는 SGC_TRIAL도 끕니다. TsgcPlannerICalendar, TsgcPlannerMSProject 또는 여섯 유닛의 다른 클래스를 사용하면서 Community에 대해서도 컴파일되어야 하는 코드는 이를 {$IFDEF SGC_PLANNER_ADVANCED} 안에 둡니다. 이 두 컴포넌트 중 하나를 포함한 폼은 Community 빌드에서 로드되지 않습니다.

관련 항목

- 아키텍처
- 소개
- 에디션별 기능
- 설치

설치

sgcPlanner는 sgcUI 배포판의 일부입니다. 설치 프로그램 하나가 모든 sgcUI 제품을 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치하기

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi와 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcPlanner**와 원하는 다른 제품을 선택합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 잠금 해제합니다.
4. 설치 프로그램은 파일을 복사하고, 선택한 각 IDE에 대해 컴퓨터에서 의존성 순서대로 패키지를 컴파일하며, 디자인 타임 패키지를 설치합니다. 컴포넌트는 **SGC Planner** 팔레트 페이지에 나타납니다.

sgcPlanner와 함께 제공되는 것

비주얼 제품의 공유 기반인 **sgcUICore**는 sgcPlanner와 함께 자동으로 설치됩니다. 자체 확인란은 없습니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	sgcPlanner와 sgcUICore를 포함해 설치된 모든 제품의 소스 코드. 이것이 sgcUI의 단일 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지, 제품과 IDE 버전마다 한 세트.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛, 예를 들어 `LibD13` .
`Demos\Planner`	sgcPlanner 데모. 데모를 참조하십시오.

설치 프로그램은 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 추가합니다.

체험판과 Community

- **체험판**: sgcUI 체험판 설치 프로그램은 sgcPlanner를 모든 기능이 포함된 체험판 모드로 설치합니다.
- **Community**: 무료 sgcUI Community 설치 프로그램은 고급 기능(반복, 자동 일정 계산, 종속성 화살표, 크리티컬 패스, 기준선, 리소스 Timeline, 스윔레인, WIP 제한, iCalendar, MS Project XML)을 제외한 sgcPlanner를 포함합니다. 에디션을 참조하십시오.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 IDE 버전에 맞는 패키지를 여십시오. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcPlannerD13.dpk`입니다.

패키지	내용

`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcPlanner<V>`	sgcPlanner 런타임 패키지: 데이터, 엔진, 뷰, VCL 컨트롤, 그리고 Delphi XE6부터는 FireMonkey 컨트롤. rtl, vcl, dbrtl, fmx(XE6 이상), sgcUICore가 필요합니다.
`dclsgcPlanner<V>`	SGC Planner 팔레트 페이지에 컴포넌트를 등록하는 디자인 타임 패키지.

의존성 순서대로 빌드하십시오: `sgcUICore`, 그다음 `sgcPlanner`, 그다음 `dclsgcPlanner`. 그런 다음 `dclsgcPlanner`를 설치하고 Delphi 버전의 라이브러리 경로에 `sgcUI\Source`를 추가하십시오(Tools > Options > Language > Delphi Options > Library > Library path). C++Builder에서는 버전에 맞는 `.cbproj` 프로젝트를 열고(예: `sgcPlannerC13.cbproj`) 빌드하십시오. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 나타납니다.

관련 항목

- 플랫폼
- 에디션
- VCL 빠른 시작

플랫폼

sgcPlanner는 모든 IDE와 플랫폼을 위한 하나의 소스 트리입니다. 데이터, 엔진, 뷰 모델은 RTL과 sgcUICore만 사용하므로 모든 대상에서 같은 코드이며, 여섯 컨트롤만 프레임워크에 묶여 있습니다.

대상	지원
Delphi VCL	Delphi 7부터 Delphi 13까지, Win32와 Win64: TsgcVCLScheduler, TsgcVCLGantt, TsgcVCLKanban.
Delphi FireMonkey	Windows, macOS, iOS, Android용 Delphi XE6부터 Delphi 13까지: TsgcFMXScheduler, TsgcFMXGantt, TsgcFMXKanban. SGC_PLANNER_FMX 아래에서 컴파일됩니다.
Linux64	Delphi 10.2 이상, SGC_PLANNER_LINUX 아래에서 컴파일: 데이터, 근무 달력, 엔진, JSON, iCalendar, MS Project XML, TsgcPlannerRenderer이므로 서버가 계획의 PNG와 SVG 파일을 만들 수 있습니다.
C++Builder	C++Builder 2007부터 C++Builder 13까지, 같은 컴포넌트와 생성된 .hpp 헤더. C++Builder 빠른 시작을 참조하십시오.
.NET	WinForms, WPF, Avalonia에서 동일한 API를 제공하는 C# 포트로, 모든 호스트에서 Scheduler, Gantt, Kanban 보드를 사용할 수 있으며 창 없는 PNG 및 SVG 출력을 지원합니다. .NET Framework 4.7.2와 .NET 8 및 9에서 동작합니다. .NET 개요를 참조하십시오.

프레임워크별 유닛

유닛	프레임워크
sgcPlanner_Const, sgcPlanner_DateUtils, sgcPlanner_Calendar, sgcPlanner_Data, sgcPlanner_TimeScale, sgcPlanner_Recurrence, sgcPlanner_AutoSchedule, sgcPlanner_JSON, sgcPlanner_XML, sgcPlanner_ICal, sgcPlanner_MSProject	없음. 모델, 엔진, 파일 형식.
sgcPlanner_View, sgcPlanner_Scheduler, sgcPlanner_Scheduler_Render, sgcPlanner_Scheduler_Timeline, sgcPlanner_Gantt, sgcPlanner_Gantt_Render, sgcPlanner_Kanban, sgcPlanner_Kanban_Render, sgcPlanner_Renderer	없음. 뷰 모델과 그 페인터로, sgcUICore의 TsgcUICanvas에 그립니다.
sgcPlanner_Source_DataSet	DB 유닛만(dbrtl), VCL이나 FireMonkey 유닛은 없음.
sgcPlanner_VCL, sgcPlanner_Scheduler_VCL, sgcPlanner_Gantt_VCL, sgcPlanner_Kanban_VCL	VCL, Windows 전용.
sgcPlanner_FMX, sgcPlanner_Scheduler_FMX, sgcPlanner_Gantt_FMX, sgcPlanner_Kanban_FMX	FireMonkey, Delphi XE6 이상.

패키지

sgcPlanner에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcPlannerD13.dpk, C++Builder 13에서는 sgcPlannerC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcPlanner	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛. 데이터, 근무 달력, 엔진, 뷰 모델과 그 페인터, JSON, iCalendar, MS Project XML, TsgcPlannerRenderer, TsgcPlannerSource_DataSet.	rtl, dbrtl, sgcUICore
sgcPlannerVCL	런타임, VCL	sgcPlanner_VCL, sgcPlanner_Scheduler_VCL, sgcPlanner_Gantt_VCL 및 sgcPlanner_Kanban_VCL (TsgcVCLScheduler, TsgcVCLGantt, TsgcVCLKanban)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcPlanner
sgcPlannerFMX	런타임, FireMonkey, XE6부터	sgcPlanner_FMX, sgcPlanner_Scheduler_FMX, sgcPlanner_Gantt_FMX 및 sgcPlanner_Kanban_FMX (TsgcFMXScheduler, TsgcFMXGantt, TsgcFMXKanban)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcPlanner
dclsgcPlanner	디자인 타임, VCL	**SGC Planner** 팔레트 페이지에 등록하는 컴포넌트는 TsgcPlannerData, TsgcVCLScheduler, TsgcVCLGantt, TsgcVCLKanban, TsgcPlannerRenderer, TsgcPlannerSource_DataSet이며, Community 에디션이 아닌 경우 TsgcPlannerICalendar와 TsgcPlannerMSProject도 등록합니다.	designide, rtl, vcl, sgcPlanner, sgcPlannerVCL
dclsgcPlannerFMX	디자인 타임, FireMonkey, XE6부터	**SGC Planner** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXScheduler, TsgcFMXGantt 및 TsgcFMXKanban입니다.	designide, rtl, fmx, sgcPlanner, sgcPlannerFMX, dclsgcPlanner
sgcPlannerWS, dclsgcPlannerWS	런타임 및 디자인 타임, sgcWebSockets 브리지	TsgcPlannerGoogleSync와, Community 에디션이 아닌 경우 TsgcPlannerICalSubscription, 프레임워크 독립이며 **SGC Planner** 팔레트 페이지에 있습니다.	rtl, sgcPlanner, sgcWebSockets

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcPlannerFMX에는 dclsgcPlanner가 필요하므로 dclsgcPlanner를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcPlanner	sgcPlannerVCL	sgcPlannerFMX	dclsgcPlanner, dclsgcPlannerFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32

10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcPlanner, sgcPlannerVCL	sgcPlannerFMX	dclsgcPlanner, dclsgcPlannerFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcPlanner만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcPlanner만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcPlannerC13.cbproj, sgcPlannerVCLC13.cbproj, sgcPlannerFMXC13.cbproj, dclsgcPlannerC13.cbproj, dclsgcPlannerFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXScheduler, TsgcFMXGantt 및 TsgcFMXKanban)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcPlannerVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcPlanner_Scheduler_FMX, sgcPlanner_Gantt_FMX 또는 sgcPlanner_Kanban_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcPlanner는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 데이터, 근무 달력, 엔진, JSON, iCalendar, MS Project XML, TsgcPlannerRenderer(예: 계획의 PNG 및 SVG 파일을 만들 때)입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

플랫폼별 참고

- **고해상도 DPI.** RowHeight, TreeWidth, SlotHeight 같은 크기는 DIP 단위입니다. VCL 컨트롤은 모니터의 PPI로, FireMonkey 컨트롤은 폼의 배율로, 렌더러는 Scale 속성으로 이를 조정합니다.
- **터치.** 컨트롤은 터치 입력을 마우스 입력으로 처리하므로 손가락으로 카드, 막대 또는 이벤트를 끌 수 있습니다. VCL 컨트롤은 자체 팬 제스처를 유지하므로 부모 스크롤 상자가 끌기를 가로채지 않습니다.
- **로캘.** 요일과 월 이름, 날짜와 시간 형식, 소수점 구분 기호는 컴퓨터에서 초기화되는 sgcPlannerLocale에서 옵니다. 따라서 Linux 서버나 다른 언어의 컴퓨터는 다른 레이블을 표시합니다. 어디서나 같은 출력을 얻으려면 시작할 때 sgcPlannerLocale의 속성을 설정하십시오.
- **Delphi 7과 Delphi 2007.** 이 버전들은 Ansi 문자열을 사용하므로 JSON, iCalendar, MS Project XML 리더는 UTF-8 텍스트를 시스템 코드 페이지로 변환하며, 그 밖의 문자는 손실됩니다.
- **에디션.** Community 에디션에서 sgcVer.inc가 끄는 SGC_PLANNER_ADVANCED가 모든 플랫폼에서 고급 기능을 결정합니다. 에디션별 기능을 참조하십시오.

관련 항목

- 아키텍처
- 설치
- 헤드리스 렌더링

VCL 빠른 시작

Delphi 버전에 맞는 sgcPlanner 패키지를 설치합니다. 컴포넌트는 **SGC Planner** 팔레트 페이지에 있습니다.

디자인 타임에

1. 폼에 **TsgcPlannerData**를 놓습니다. Statuses 컬렉션에 상태(예: todo, doing, done)를, Resources에 사람을, Items에 작업을 추가합니다: 각각 Id, Title, Start, Finish, Status를 지정합니다.
2. **TsgcVCLGantt**를 놓고, Align을 alTop으로 설정하고, Data 속성을 데이터 객체로 설정합니다.
3. **TsgcVCLKanban**을 놓고, Align을 alRight로 설정하고, Data를 설정합니다. 보드는 상태마다 열 하나를 보여 줍니다.
4. **TsgcVCLScheduler**를 놓고, Align을 alClient로 설정하고, Data를 설정하고, View(예: psvWorkWeek)를 고릅니다.
5. 애플리케이션을 실행합니다. 카드를 다른 열로, Gantt의 막대를, 또는 Scheduler의 이벤트를 끌어 보십시오: 세 뷰가 데이터를 공유하므로 변경이 즉시 나타납니다.

코드에서

모든 뷰가 알림을 한 번만 받도록 데이터를 BeginUpdate와 EndUpdate 사이에서 채우십시오. 항목의 Finish는 포함되지 않습니다(배타적).

관련 항목

- TsgcPlannerData
- TsgcVCLGantt
- TsgcVCLKanban
- TsgcVCLScheduler
- 데모
- 설치

FMX 빠른 시작

FireMonkey 컴포넌트는 Delphi XE6 이상이 필요합니다. FireMonkey 폼의 **SGC Planner** 팔레트 페이지에 비주얼이 아닌 컴포넌트와 함께 있습니다.

디자인 타임에

1. **TsgcPlannerData**를 놓고 VCL 빠른 시작에서처럼 채웁니다.
2. **TsgcFMXGantt**, **TsgcFMXKanban**, **TsgcFMXScheduler**를 놓고, Align을 설정하고, 각각의 Data 속성을 데이터 객체로 설정합니다.

FMX 컨트롤은 자체 스크롤 막대를 그리고, 마우스 휠로 스크롤하며, 터치 입력을 마우스 입력으로 처리하므로 손가락으로 카드나 막대를 끌 수 있습니다. 속성, 메서드, 이벤트는 VCL과 같습니다.

코드에서

관련 항목

- TsgcFMXScheduler
- TsgcFMXGantt
- TsgcFMXKanban
- 데모
- 플랫폼

C++Builder 빠른 시작

C++Builder 버전에 맞는 sgcPlanner 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 및 라이브러리 경로에 추가한 뒤, **SGC Planner** 팔레트 페이지의 컴포넌트를 Delphi와 똑같이 사용합니다. 설치 프로그램이 두 경로를 IDE에 자동으로 추가합니다. sgcPlanner와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 폴더 하나에 있으므로 sgcUICore에는 별도의 경로가 필요 없습니다.

C++에서 알아둘 점

- 생성자에는 기본 인수가 없습니다. 소유자 없이 비주얼이 아닌 컴포넌트를 만들 때는 NULL을 사용하십시오: `new TsgcPlannerRenderer(NULL)`.
- 이 도움말의 샘플은 메서드의 모든 인수를 명시적으로 전달하므로(예: `AddStatus(Id, Caption, 0)`과 `AddDependency(FromId, ToId, pdkFS, 0)`) 생성된 헤더의 어떤 버전에서도 컴파일됩니다.
- 이벤트는 클로저이므로 핸들러는 클래스의 __fastcall 멤버 함수입니다. aAccept 같은 var 매개변수는 bool 참조이고, const string 매개변수는 `const String`입니다.
- 열거형은 범위가 지정되지 않습니다: `psvWorkWeek`, `prvGantt` 또는 `pwmBlock`처럼 씁니다.

코드에서

관련 항목

- TsgcPlannerData
- VCL 빠른 시작
- TsgcPlannerRenderer
- 데모
- 플랫폼

데모

데모는 `sgcUI\Demos\Planner\Delphi`에 있으며, 데모마다 폴더 하나와 자세한 내용이 담긴 `readme.html`이 있습니다. 대부분의 데모는 2026년 10월 5일 주간을 중심으로 한 작은 결정적 생성기로 데이터를 만듭니다.

데모	컴포넌트	보여 주는 내용
ProjectPlanner	TsgcVCLGantt, TsgcVCLKanban, TsgcVCLScheduler	하나의 TsgcPlannerData에 바인딩된 세 가지 뷰로 보는 웹사이트 재출시. 카드를 To do에서 Done으로 끌면 Gantt의 상태 열, 막대 색, Scheduler의 이벤트 색이 따라갑니다. BarColorMode pbcStatus로 상태 색을 사용하고 SelectedId로 선택을 공유합니다.
TeamScheduler	TsgcVCLScheduler	여섯 뷰(일, 주중, 주, 월, 안건 목록, 리소스 Timeline)에서 보는 다섯 사람의 3주간 회의. RecurrenceRule에서 만든 매일 스탠드업과 주간 동기화, OnItemCreate로 끌어서 만들기, Options.GroupByResource로 그룹화.
ProjectGantt	TsgcVCLGantt, TsgcPlannerMSProject	ParentId로 만든 세 WBS 단계가 있는 소프트웨어 릴리스 계획, OnLinkCreating으로 끌어서 링크 만들기, 근무 달력에서의 Schedule과 AutoSchedule, 크리티컬 패스, 이동한 막대가 얼마나 지연되었는지 보여 주는 기준선, Microsoft Project XML 열기와 저장. Registered 에디션.
TaskBoard	TsgcVCLKanban	Columns의 열 너비와 WIP 제한이 있는 카드 26장의 백로그, 경고 및 차단 WIP 모드, 리소스, 부모 또는 태그별 스윔레인, 키보드 탐색, 접을 수 있는 Done 열.
DataSetPlanner	TsgcVCLGantt, TsgcVCLKanban, TsgcPlannerSource_DataSet	자체 필드 이름을 가진 TClientDataSet 테이블 세 개(작업, 사람, 링크)를 ItemFields, ResourceFields, DependencyFields로 매핑합니다. TDBGrid에서 게시한 레코드는 해당 항목을 업데이트하고, 뷰에서 이동한 카드나 막대는 테이블에 기록되며, 그리드가 레코드를 편집하는 동안 기록은 대기합니다. MidasLib가 링크되므로 midas.dll이 필요 없습니다.
ICalendarSync	TsgcVCLScheduler, TsgcPlannerICalendar	시리즈의 RRULE이 있는 .ics로 내보내기, 종일 항목을 DATE 값으로, ExportUTC로 UTC 또는 부동 시간으로. Google Calendar, Outlook 또는 모든 RFC 5545 파일에서 시간대를 로컬 시간으로 변환하여 가져오기, ImportMode로 바꾸기 또는 병합, OnImportWarning의 경고. Registered 에디션.
GoogleCalendarSync	TsgcVCLScheduler, TsgcPlannerGoogleSync	sgcWebSockets의 TsgcHTTPGoogleCloud_Calendar_Client를 통해 Google 캘린더와 양방향으로 동기화하는 주 뷰. OAuth2 클라이언트 ID와 보안 비밀은 실행 시 입력되며 저장되지 않습니다. Direction, OnConflict 선택, SyncAsync를 통한 지금 동기화, AutoSyncInterval을 통한 자동 동기화, OnSyncDone, OnConflict, OnError 로그를 갖추고 있습니다. Registered 에디션, sgcWebSockets가 필요합니다.
ICalSubscription	TsgcVCLScheduler, TsgcPlannerICalSubscription	두 로컬 항목 옆에 병합된 iCalendar URL. URL, 새로고침 간격, RefreshAsync를 통한 새로 고침, Active를 통한 자동 새로 고침, OnItemChanging과 IsSubscriptionItem을 통한 읽기 전용 피드 이벤트, OnRefreshed, OnError, OnImportWarning 로그를

		갖추고 있습니다. Registered 에디션, sgcWebSockets가 필요합니다.
PlannerFMX	TsgcFMXGantt, TsgcFMXScheduler, TsgcFMXKanban, TsgcPlannerRenderer	Windows, macOS, iOS, Android용 하나의 데이터 객체 위의 세 가지 FireMonkey 뷰: 한 뷰에서 이동한 카드, 막대 또는 이벤트는 다른 두 뷰에서도 이동합니다. Scheduler 뷰, Gantt 확대/축소와 맞춤, 선택 항목을 Done으로, 어두운 테마를 갖춘 도구 모음, 그리고 컨트롤의 옵션으로 헤드리스 렌더링한 각 뷰.
HeadlessReport	TsgcPlannerRenderer	VCL이나 FireMonkey 유닛이 없는 Windows 및 Linux64용 콘솔 애플리케이션입니다. sgcPlannerJSONLoadFromFile로 프로젝트를 로드하거나, 먼저 샘플 프로젝트를 project.json으로 기록하고, 렌더러가 소유한 뷰에 ScrollToTime, ExpandAll, ZoomToFit을 호출한 뒤 scheduler, gantt, kanban 페이지를 PNG와 SVG로 저장합니다. 사용법: `HeadlessReport [project.json] [output folder]`.

명령줄 스위치

모든 데모에는 빌드 검증에 사용하는 스위치가 두 개 있습니다. `/snapshot <file.png>`는 데모의 이미지를 저장하고 종료합니다: VCL 뷰 데모는 TsgcPlannerRenderer로 뷰를 렌더링하고, DataSetPlanner, ICalendarSync, PlannerFMX는 창을 저장하며(Delphi 7에서는 BMP), HeadlessReport는 샘플 프로젝트의 Gantt 페이지를 기록합니다. `/test [file.txt]` 또는 `--test`는 데모의 주요 동작에 대한 자체 테스트를 실행하고, 검사마다 PASS 또는 FAIL 한 줄을 기록하며, 실패 수를 종료 코드로 반환합니다. 뷰 데모는 실제 마우스 및 키 메시지로 컨트롤을 구동합니다.
VCL 데모는 Delphi 7부터 Delphi 13까지, Win32와 Win64로 빌드됩니다. PlannerFMX는 Delphi XE6부터 Delphi 13까지 빌드됩니다.

관련 항목

- VCL 빠른 시작
- FMX 빠른 시작
- 설치

TsgcPlannerData

유닛: sgcPlanner_Data
TsgcPlannerData는 sgcPlanner의 공유 데이터이며 비주얼이 아닌 컴포넌트입니다. 항목, 리소스, 상태, 종속성, 근무 달력을 보관하고 여기에 바인딩된 모든 뷰에 알립니다. Scheduler, Gantt, Kanban, DataSet 소스, iCalendar 및 MS Project 컴포넌트, 렌더러가 모두 같은 데이터 객체를 동시에 사용할 수 있습니다.

속성

속성	형식	설명
Items	TsgcPlannerItems	작업, 카드, 이벤트. 아래의 Items를 참조하십시오.
Resources	TsgcPlannerResources	사람 또는 회의실: Id, Name, Color, Initials(Name이 비어 있으면 이름에서 계산). AddResource(aId, aName)와 Find(aId).
Statuses	TsgcPlannerStatuses	워크플로 상태: Id, Caption, Color. Kanban의 기본 열입니다. AddStatus(aId, aCaption, aColor = 0)와 Find(aId).
Dependencies	TsgcPlannerDependencies	항목 사이의 링크: FromId, ToId, Kind(pdkFS 기본값, pdkSS, pdkFF, pdkSF), 작업 분 단위의 Lag(선행이면 음수). AddDependency(aFromId, aToId, aKind = pdkFS, aLag = 0).
Calendar	TsgcPlannerCalendar	근무 달력: WorkDays(기본값은 월요일부터 금요일, 비어 있으면 매일), WorkTime(기본값 `08:00-12:00,13:00-17:00` , 최대 4개 구간, 비어 있으면 하루 전체), Holidays(Date, Name), FirstDayOfWeek(기본값 pwdMonday). 메서드 IsWorkingDay, IsWorkingTime, WorkingMinutesPerDay, NextWorkingTime, PrevWorkingTime, AddWorkingMinutes, WorkingMinutesBetween.
ProjectStart	TDateTime	자동 일정 계산을 위한 프로젝트 시작일. 0이면 작업의 가장 이른 Start를 사용합니다. ProjectStartTime은 이에 대응하는 TsgcPlannerTime 속성입니다.
SelectedId	string	Public. 선택된 항목의 Id이며 SyncSelection이 있는 뷰들이 공유합니다.
Version	Int64	Public, 읽기 전용. 선택만 바뀌는 경우를 제외한 모든 변경 때마다 증가합니다.
UpdateCount	Integer	Public, 읽기 전용. BeginUpdate의 깊이.

Items

TsgcPlannerItem은 Gantt에서는 작업, Kanban에서는 카드, Scheduler에서는 이벤트입니다.

속성	형식	설명
Id	string	항목의 고유 키. 부모, 링크, DataSet 소스의 리소스, 가져온 파일이 이를 참조합니다.
Title, Notes	string	항목의 텍스트.

Start, Finish	TDateTime	날짜. Finish는 포함되지 않습니다(배타적). StartTime과 FinishTime은 정확한 TsgcPlannerTime 값입니다.
AllDay	Boolean	종일 이벤트. Scheduler의 종일 영역에 표시됩니다.
Status	string	상태의 Id: 카드의 Kanban 열. 대소문자를 구분하여 일치시킵니다.
Progress	Double	완료율(퍼센트), 0에서 100.
Color	TsgcUIColor	$AARRGGBB. 0(기본값)이면 상태의 색, 그다음 첫 번째 리소스의 색, 그다음 Id로 고른 팔레트 색을 사용합니다.
ParentId	string	부모 항목의 Id. 자식이 있는 항목은 요약 항목입니다.
Tags, Resources	string	쉼표로 구분한 목록: 태그와 리소스 Id. TagCount, TagAt, ResourceCount, ResourceAt으로 읽습니다.
Rank	Integer	Kanban 열에서 카드의 위치.
Milestone, Manual	Boolean	마일스톤은 마름모로 그려집니다. 수동 작업은 프로젝트 일정을 계산할 때 날짜를 유지합니다.
Constraint, ConstraintDate	TsgcPlannerConstraint, TDateTime	pcASAP(기본값) 또는 pcSNET, 즉 ConstraintDate보다 일찍 시작하지 않음. 자동 일정 계산에 사용됩니다.
BaselineStart, BaselineFinish	TDateTime	저장된 계획, 설정하지 않으면 0. 그리려면 Registered 에디션이 필요합니다.
RecurrenceRule, RecurrenceExceptions	string	RFC 5545의 RRULE(예: `FREQ=WEEKLY;BYDAY=MO,WE`)과 제외할 날짜 또는 날짜 시간의 쉼표 목록. Registered 에디션에서 전개됩니다.
RecurrenceParentId, RecurrenceDate	string, TDateTime	수정된 발생 항목에 설정합니다: 소속된 시리즈와 대체하는 발생 항목의 시작.
IsSummary, Level, Duration, TotalSlack, Critical		Public, 읽기 전용. 요약 항목은 자식이 있으며, Level은 트리에서의 깊이, Duration은 달력상의 작업 분, TotalSlack과 Critical은 일정 계산 엔진의 결과입니다.

메서드

메서드	반환값	설명
BeginUpdate, EndUpdate		변경을 묶습니다. 리스너와 OnChange는 가장 바깥쪽 EndUpdate 이후에 한 번만 발생합니다.
AddItem(aId = '')	TsgcPlannerItem	해당 Id로 항목을 추가합니다.
FindItem(aId), IndexOfItem(aId)	TsgcPlannerItem, Integer	정렬된 인덱스를 통해 Id로 항목을 찾습니다. 없으면 nil 또는 -1.
DeleteItem(aId)	Boolean	항목, 그 수정된 발생 항목, 그 링크를 삭제합니다. 자식은 부모로 이동하고, 선택된 항목이었다면 선택이 해제됩니다. 알 수 없는 Id이면 False.
MoveItem(aId, aStatus, aRank)	Boolean	카드를 상태와 위치로 이동하고 두 열의 순위를 다시 매깁니다. Kanban에서 놓을 때 일어나는 동작입니다.
ChildrenOf(aId, var aList)	Integer	항목의 자식, 또는 Id가 비어 있으면 루트 항목.
UpdateSummaries		잎 항목의 날짜를 요약 항목으로 올리며, 진행률은 잎의 작업 기간으로 가중합니다.
GetOccurrences(aStart, aFinish, var aList)	Integer	반열린 구간 [aStart, aFinish)의 발생 항목을 TsgcPlannerTime으로, 시작 순으로 정렬해 반환합니다. Registered 에디션에서 반복 항목은 규칙의 날짜마다 발생 항목 하나를 만들며, 예외와 수정된 발생 항목은 제외됩니다.
GetRange(out aStart, aFinish)	Boolean	항목의 가장 이른 시작과 가장 늦은 종료. 항목이 없으면 False.
ItemChanging(Sender, aItem, aFields)	Boolean	OnItemChanging을 발생시키고 aAccept를 반환합니다. 사용자 지정 뷰가 편집을 커밋하기 전에 호출합니다.
AddListener, RemoveListener(aListener)		모든 변경을 통지받는 TsgcPlannerChangeEvent를 등록합니다. 뷰가 이를 사용합니다. 리스너는 데이터를 변경해서는 안 됩니다.
Clear		항목, 리소스, 상태, 종속성을 삭제합니다. 달력과 ProjectStart는 유지됩니다.
LoadFromJSON, SaveToJSON, LoadFromFile, SaveToFile		전체 데이터를 JSON(BOM 없는 UTF-8)으로 처리합니다. 읽을 수 없는 문서는 EsgcPlannerError를 발생시키고 데이터를 변경하지 않습니다.

이벤트

이벤트	형식	설명
OnChange	TsgcPlannerChangeEvent	변경의 가장 바깥쪽 EndUpdate 이후, 리스너 다음에 발생합니다: Kinds, ItemId(정확히 한 항목이 바뀐 경우), Version을 가진 aChange. 핸들러가 데이터를 다시 변경해도 됩니다.
OnItemChanging	TsgcPlannerItemChangingEvent	뷰가 마우스로 한 편집을 커밋하기 전에 발생합니다: aItem, aFields(pifStart, pifFinish, pifStatus, pifProgress 및 기타 항목 필드의 집합), var aAccept. aAccept를 False로 설정하면 놓기가 취소됩니다.

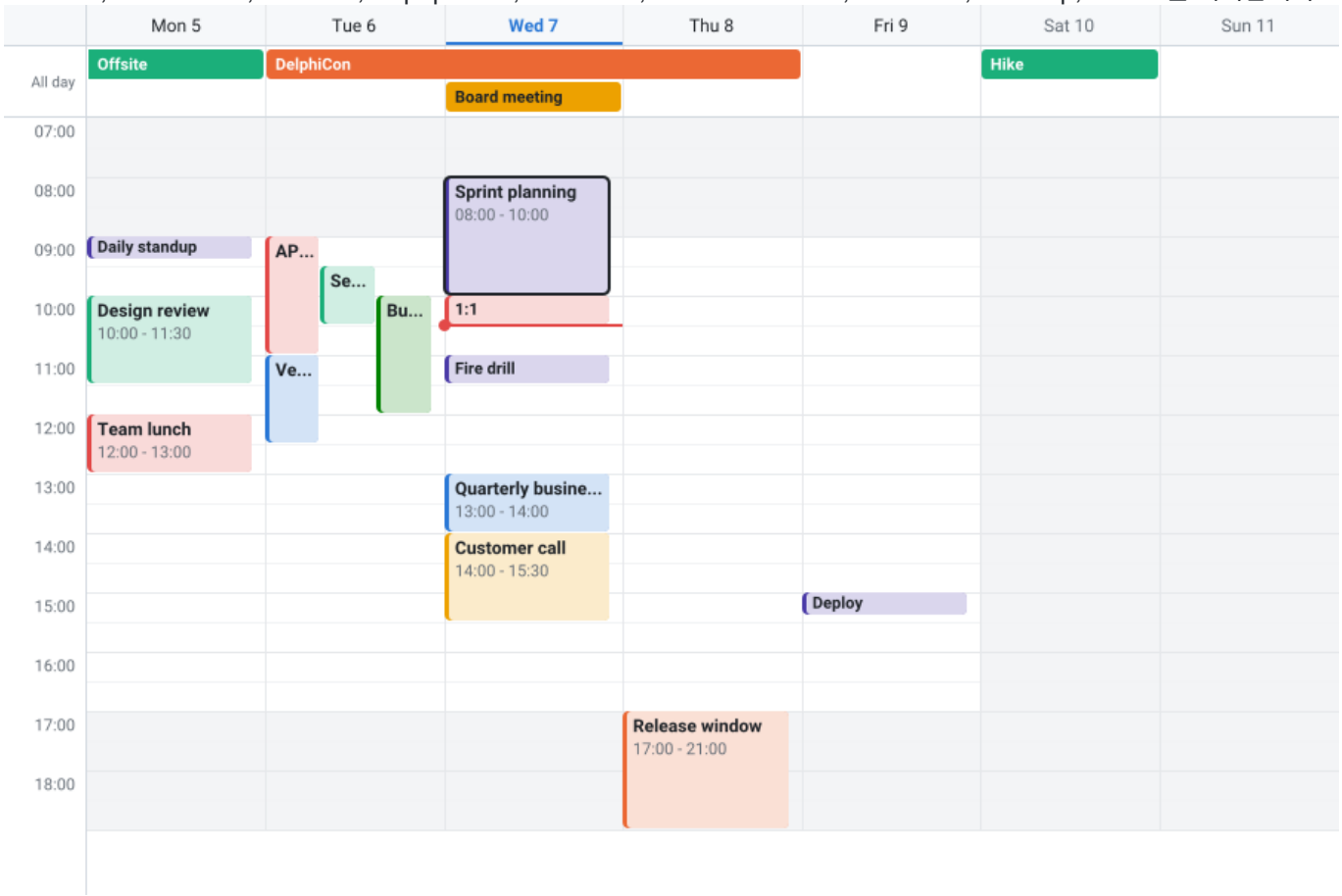
예제: 근무 달력, 반복 회의, 변경 추적

관련 항목

- 아키텍처
- VCL 빠른 시작
- TsgcPlannerSource_DataSet
- 공유 데이터
- 근무 달력
- 반복
- 종속성
- JSON 저장

TsgcVCLScheduler

유닛: sgcPlanner_Scheduler_VCL
TsgcVCLScheduler는 Windows 전용 VCL Scheduler입니다. TsgcPlannerData의 항목을 일, 주중, 주, 월 또는 안건 뷰로 보여 주며, 종일 영역, 현재 시간 선, 나란히 놓인 겹치는 이벤트가 있고, Options.GroupByResource를 사용하면 리소스마다 열 하나가 있습니다. Registered 에디션에는 리소스 Timeline도 있습니다: 시간 눈금 위에서 리소스마다 한 행입니다. 사용자는 빈 공간 위로 끌어 이벤트를 만들고, 이동하고 크기를 바꾸며, 놓을 때 SlotMinutes에 맞춰집니다. 다른 리소스의 열로 이동하면 리소스가 바뀝니다. 일 뷰에서 Ctrl과 마우스 휠은 슬롯 높이를 바꿉니다. TsgcUIVCLControl에서 상속하며 Align, Anchors, Constraints, Enabled, PopupMenu, ShowHint, ParentShowHint, TabOrder, TabStop, Visible을 게시합니다.



속성

속성	형식	설명
Data	TsgcPlannerData	컨트롤이 보여 주는 데이터. 여러 컨트롤이 하나의 데이터 객체를 공유할 수 있습니다. nil이면 컨트롤은 자체 내부 데이터 객체를 사용합니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. utCustom은 밝은 색에서 시작합니다.
Colors	TsgcPlannerViewColors	테마 색의 재정의: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 0(기본값)이면 테마의 색을 유지합니다.
ReadOnly	Boolean	모든 끌기와 Delete 키를 끕니다. 기본값 False.

DragOptions	TsgcPlannerDragOptions	사용자가 할 수 있는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다.
SyncSelection	Boolean	데이터의 SelectedId를 통해 같은 데이터의 다른 뷰와 선택을 공유합니다. 기본값 True.
MaxFPS	Integer	컨트롤이 그리는 초당 최대 프레임 수.
BorderStyle	TBorderStyle	bsSingle(기본값) 또는 bsNone.
View	TsgcPlannerSchedulerViewKind	psvDay, psvWorkWeek(달력의 근무일), psvWeek(기본값), psvMonth(6주), psvAgenda(AgendaDays일의 목록) 또는 psvTimeline(리소스마다 한 행, Registered 에디션).
Date	TDateTime	뷰가 보여 주는 날짜. 0(기본값)이면 시계를 따릅니다.
옵션	TsgcPlannerSchedulerOptions	DayStartHour와 DayEndHour(기본값 0과 24), SlotMinutes(30), SlotHeight(20 DIP), ShowAllDayBand(True), ShowCurrentTime(True), GroupByResource(False), MonthMaxItems(0, 들어가는 만큼), AgendaDays(7), ResourceRowHeight(40 DIP, Timeline).
TimeScale	TsgcPlannerTimeScale	Timeline의 시간 축: Zoom(pz5Minutes에서 pzYears까지, 기본값 pzDays), TickWidth, Origin, Snap, ShowNonWorking, MajorFormat, MinorFormat. 확대/축소는 Timeline의 기간도 정합니다: pzHours까지는 하루, pzDays에서는 4주, pzWeeks에서는 3개월, pzMonths에서는 1년.
SchedulerView	TsgcPlannerSchedulerView	Public, 읽기 전용. 날짜, 열, 항목의 레이아웃을 가진 뷰 모델.

메서드

메서드	반환값	설명
GoToday		Date를 오늘로 설정합니다.
Next, Prior		Date를 뷰의 다음 또는 이전 기간으로 이동합니다.
ScrollToTime(aTime)		aTime의 날짜를 표시하고 시간 그리드를 해당 시간으로 스크롤합니다.
VisibleRange(out aStart, aFinish)		뷰가 보여 주는 기간.
ItemAt(X, Y)	TsgcPlannerItem	컨트롤의 한 지점 아래에 있는 항목, 없으면 nil.
ScrollToItem(aId)	Boolean	항목이 보일 때까지 스크롤합니다. 항목이 표시되지 않으면 False.
SaveToPNG, SaveToSVG(aFileName 또는 aStream, aWidth = 0, aHeight = 0, aScale = 0)		뷰를 이미지로 내보냅니다. 같은 데이터, 옵션, 크기, 배율을 가진 TsgcPlannerRenderer와 같은 이미지입니다. 크기는 DIP 단위이며 0은 클라이언트 크기, 배율 0은 컨트롤의 배율입니다. 스크롤 위치는 유지됩니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

이벤트

이벤트	형식	설명
OnItemClick, OnItemDblClick	TsgcPlannerItemEvent	항목의 클릭 또는 더블 클릭: aItem.
OnItemDelete	TsgcPlannerItemAcceptEvent	선택된 항목에서의 Delete 키: aItem과 var aAccept. aAccept를 False로 설정하면 항목이 유지됩니다. 삭제된 항목은 링크도 함께 삭제하며 자식은 부모로 이동합니다.
OnSelectionChange	TNotifyEvent	이 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 선택된 항목이 바뀌었습니다.
OnItemCreate	TsgcPlannerSchedulerItemCreateEvent	빈 공간 위에서의 끌기: aStart, aFinish, aResourceId(리소스별로 그룹화한 경우의 열), var aTitle(기본값 "New event"), var aAccept(기본값 True). 새 항목이 선택됩니다.
OnRecurrenceEdit	TsgcPlannerRecurrenceEditEvent	반복 시리즈의 발생 항목 위에 놓기: aItem(시리즈), aOccurrence, var aScope. prsOccurrence(기본값)는 수정된 발생 항목을 만들고, prsSeries는 전체 시리즈를 이동하며, prsCancel은 놓기를 취소합니다. Registered 에디션.
OnRecurrenceDelete	TsgcPlannerRecurrenceEditEvent	반복 시리즈의 발생 항목에서의 Delete 키, 무엇이든 삭제되기 전에 발생합니다. prsOccurrence(기본값)는 발생 항목을 시리즈의 예외에 추가하고, prsSeries는 OnItemDelete 뒤에 시리즈를 삭제하며, prsCancel은 아무것도 삭제하지 않습니다. Registered 에디션.
OnDateChange, OnViewChange	TNotifyEvent	Date 또는 View가 바뀌었습니다. 월 보기에서 "+N more"를 클릭해 그 날을 여는 경우에도 발생합니다.

예제: 업무 시간, 리소스 열, 반복 이벤트

관련 항목

- TsgcFMXScheduler
- TsgcPlannerData
- VCL 빠른 시작
- 반복
- 시간 눈금과 확대/축소
- 끌어서 놓기
- 동기화된 뷰

TsgcFMXScheduler

유닛: sgcPlanner_Scheduler_FMX
TsgcFMXScheduler는 Delphi XE6 부터 사용할 수 있는 FireMonkey Scheduler 입니다(SGC_PLANNER_FMX). TsgcVCLScheduler와 같은 뷰, 속성, 메서드, 이벤트를 갖습니다. 자체 스크롤 막대를 그리며 터치 입력을 마우스 입력으로 처리합니다. TsgcUIFMXControl에서 상속하며 일반적인 FireMonkey 레이아웃 속성과 이벤트를 게시합니다.

속성

속성	형식	설명
Data	TsgcPlannerData	컨트롤이 보여 주는 데이터. 여러 컨트롤이 하나의 데이터 객체를 공유할 수 있습니다. nil이면 컨트롤은 자체 내부 데이터 객체를 사용합니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. utCustom은 밝은 색에서 시작합니다.
Colors	TsgcPlannerViewColors	테마 색의 재정의: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 0(기본값)이면 테마의 색을 유지합니다.
ReadOnly	Boolean	모든 끌기와 Delete 키를 끕니다. 기본값 False.
DragOptions	TsgcPlannerDragOptions	사용자가 할 수 있는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다.
SyncSelection	Boolean	데이터의 SelectedId를 통해 같은 데이터의 다른 뷰와 선택을 공유합니다. 기본값 True.
MaxFPS	Integer	컨트롤이 그리는 초당 최대 프레임 수.
View	TsgcPlannerSchedulerViewKind	psvDay, psvWorkWeek(달력의 근무일), psvWeek(기본값), psvMonth(6주), psvAgenda(AgendaDays일의 목록) 또는 psvTimeline(리소스마다 한 행, Registered 에디션).
Date	TDateTime	뷰가 보여 주는 날짜. 0(기본값)이면 시계를 따릅니다.
옵션	TsgcPlannerSchedulerOptions	DayStartHour와 DayEndHour(기본값 0과 24), SlotMinutes(30), SlotHeight(20 DIP), ShowAllDayBand(True), ShowCurrentTime(True), GroupByResource(False), MonthMaxItems(0, 들어가는 만큼), AgendaDays(7), ResourceRowHeight(40 DIP, Timeline).
TimeScale	TsgcPlannerTimeScale	Timeline의 시간 축: Zoom(pz5Minutes에서 pzYears까지, 기본값 pzDays), TickWidth, Origin, Snap, ShowNonWorking, MajorFormat, MinorFormat. 확대/축소는 Timeline의 기간도 정합니다: pzHours까지는 하루, pzDays에서는 4주, pzWeeks에서는 3개월, pzMonths에서는 1년.
SchedulerView	TsgcPlannerSchedulerView	Public, 읽기 전용. 날짜, 열, 항목의 레이아웃을 가진 뷰 모델.

메서드

메서드	반환값	설명
GoToday		Date를 오늘로 설정합니다.
Next, Prior		Date를 뷰의 다음 또는 이전 기간으로 이동합니다.
ScrollToTime(aTime)		aTime의 날짜를 표시하고 시간 그리드를 해당 시간으로 스크롤합니다.
VisibleRange(out aStart, aFinish)		뷰가 보여 주는 기간.
ItemAt(X, Y (Single))	TsgcPlannerItem	컨트롤의 한 지점 아래에 있는 항목, 없으면 nil.
ScrollToItem(aId)	Boolean	항목이 보일 때까지 스크롤합니다. 항목이 표시되지 않으면 False.
SaveToPNG, SaveToSVG(aFileName 또는 aStream, aWidth = 0, aHeight = 0, aScale = 0)		뷰를 이미지로 내보냅니다. 같은 데이터, 옵션, 크기, 배율을 가진 TsgcPlannerRenderer와 같은 이미지입니다. 크기는 DIP 단위이며 0은 클라이언트 크기, 배율 0은 컨트롤의 배율입니다. 스크롤 위치는 유지됩니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

이벤트

이벤트	형식	설명
OnItemClick, OnItemDblClick	TsgcPlannerItemEvent	항목의 클릭 또는 더블 클릭: aItem.
OnItemDelete	TsgcPlannerItemAcceptEvent	선택된 항목에서의 Delete 키: aItem과 var aAccept. aAccept를 False로 설정하면 항목이 유지됩니다. 삭제된 항목은 링크도 함께 삭제하며 자식은 부모로 이동합니다.
OnSelectionChange	TNotifyEvent	이 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 선택된 항목이 바뀌었습니다.
OnItemCreate	TsgcPlannerSchedulerItemCreateEvent	빈 공간 위에서의 끌기: aStart, aFinish, aResourceId(리소스별로 그룹화한 경우의 열), var aTitle(기본값 "New event"), var aAccept(기본값 True). 새 항목이 선택됩니다.
OnRecurrenceEdit	TsgcPlannerRecurrenceEditEvent	반복 시리즈의 발생 항목 위에 놓기: aItem(시리즈), aOccurrence, var aScope. prsOccurrence(기본값)는 수정된 발생 항목을 만들고, prsSeries는 전체 시리즈를 이동하며, prsCancel은 놓기를 취소합니다. Registered 에디션.
OnRecurrenceDelete	TsgcPlannerRecurrenceEditEvent	반복 시리즈의 발생 항목에서의 Delete 키, 무엇이든 삭제되기 전에 발생합니다. prsOccurrence(기본값)는 발생 항목을 시리즈의 예외에 추가하고, prsSeries는 OnItemDelete 뒤에 시리즈를 삭제하며,

		prsCancel은 아무것도 삭제하지 않습니다. Registered 에디션.
OnDateChange, OnViewChange	TNotifyEvent	Date 또는 View가 바뀌었습니다. 월 보기에서 "+N more"를 클릭해 그 날을 여는 경우에도 발생합니다.

예제: PNG로 저장한 리소스 Timeline

관련 항목

- TsgcVCLScheduler
- FMX 빠른 시작
- 반복
- 시간 눈금과 확대/축소
- 끌어서 놓기

TsgcVCLGantt

유닛: sgcPlanner_Gantt_VCL
TsgcVCLGantt는 Windows 전용 VCL Gantt 차트입니다. TsgcPlannerData의 항목을 WBS 번호와 열을 가진 작업 트리와, 2단 시간 눈금 위의 막대 차트로 보여 줍니다. 진행률이 있는 막대, 마일스톤은 마름모, 요약은 대괄호, 비근무 시간은 음영, 오늘은 선으로 표시됩니다. 사용자는 막대를 이동하고(근무 기간이 유지됩니다), 크기를 바꾸고, 진행률을 끌고, 막대 끝에서 다른 막대로 끌어 링크를 만들고, 요약을 열고 닫습니다. 이동하면 작업이 Manual이 아닌 한 ConstraintDate보다 일찍 시작하지 않음 제약이 설정되며, 요약은 자식을 따라갑니다. Registered 에디션에서는 링크가 화살표로 그려지고, Gantt가 프로젝트 일정을 계산하고 크리티컬 패스를 표시하며 계획을 기준선과 비교할 수 있습니다. Ctrl과 마우스 휠로 시간 눈금을 확대/축소합니다. TsgcUIVCLControl 에서 상속하며 Align, Anchors, Constraints, Enabled, PopupMenu, ShowHint, ParentShowHint, TabOrder, TabStop, Visible을 게시합니다.

속성

속성	형식	설명
Data	TsgcPlannerData	컨트롤이 보여 주는 데이터. 여러 컨트롤이 하나의 데이터 객체를 공유할 수 있습니다. nil이면 컨트롤은 자체 내부 데이터 객체를 사용합니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. utCustom은 밝은 색에서 시작합니다.
Colors	TsgcPlannerViewColors	테마 색의 재정의: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 0(기본값)이면 테마의 색을 유지합니다.
ReadOnly	Boolean	모든 끌기와 Delete 키를 끕니다. 기본값 False.
DragOptions	TsgcPlannerDragOptions	사용자가 할 수 있는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다.
SyncSelection	Boolean	데이터의 SelectedId를 통해 같은 데이터의 다른 뷰와 선택을 공유합니다. 기본값 True.
MaxFPS	Integer	컨트롤이 그리는 초당 최대 프레임 수.
BorderStyle	TBorderStyle	bsSingle(기본값) 또는 bsNone.
Columns	TsgcPlannerGanttColumns	작업 트리의 열. 각 열에는 Field(pgcId, pgcWBS, pgcTitle, pgcStart, pgcFinish, pgcDuration, pgcProgress, pgcStatus, pgcResources), Caption(비어 있으면 기본 캡션), Width, Visible, Alignment가 있습니다. 기본 열은 ID, 작업 이름, 시작, 종료, 기간, 진행률입니다. AddColumn(aField, aWidth = 100, aAlignment = taLeftJustify)과 SetDefaults.
TreeWidth	Integer	작업 트리의 너비(DIP). 기본값 550. 사용자가 분할선을 끌어 변경합니다.
RowHeight	Integer	행의 높이(DIP). 기본값 28.
TimeScale	TsgcPlannerTimeScale	시간 축: Zoom(pz5Minutes, pz15Minutes, pzHours, pzDays 기본값, pzWeeks, pzMonths, pzQuarters, pzYears), TickWidth(0이면 확대/축소 수준의 기본값), Origin, Snap(psNone, psTick 기본값, ps5Min, ps15Min,

		psHour, psDay), ShowNonWorking(True), 머리글 두 단의 MajorFormat과 MinorFormat.
DurationUnit	TsgcPlannerDurationUnit	기간 열의 단위: pduMinutes, pduHours, pduDays(기본값) 또는 pduWeeks, 근무 시간으로 계산됩니다. 소수점 구분 기호는 Planner 로캘의 것입니다.
BarColorMode	TsgcPlannerBarColorMode	pbcItem(기본값)은 항목과 같은 색으로, pbcStatus는 상태 색으로, pbcResource는 첫 번째 리소스 색으로 막대를 칠합니다.
ShowProgress, ShowToday	Boolean	막대 안에 진행률을, 오늘 위치에 선을 그립니다. 기본값 True.
ShowDependencies	Boolean	링크를 꺾인 화살표로 그립니다. 기본값 True. Registered 에디션.
ShowCriticalPath, ShowBaseline	Boolean	크리티컬 작업과 링크를 크리티컬 색으로, 각 막대 아래에 기준선을 그립니다. 기본값 False. Registered 에디션.
AutoSchedule	Boolean	차트에서 편집할 때마다 프로젝트 일정을 다시 계산합니다. 기본값 False. Registered 에디션.
GanttView	TsgcPlannerGanttView	Public, 읽기 전용. 행, 막대 지오메트리, 링크를 가진 뷰 모델.
View	TsgcPlannerView	Public, 읽기 전용. 컨트롤이 호스팅하는 뷰 모델.

메서드

메서드	반환값	설명
Expand(aId), Collapse(aId)	Boolean	요약 항목을 열거나 닫습니다.
ExpandAll, CollapseAll		모든 요약 항목을 열거나 닫습니다.
ZoomIn, ZoomOut	Boolean	차트 중심을 기준으로 확대/축소 수준을 한 단계 더 세밀하게 또는 더 거칠게 합니다. 마지막 수준이면 False.
ZoomToFit	Boolean	모든 행이 보이는 가장 세밀한 확대/축소 수준을 고르고 첫 시작으로 스크롤합니다. 행이 없으면 False.
ScrollToTime(aTime)		aTime이 왼쪽 가장자리에 오도록 차트를 스크롤합니다.
Schedule	Boolean	데이터에 일정 계산 엔진을 실행합니다: ProjectStart부터 근무 달력에 따라 작업을 선행 작업 뒤에 배치하고, 여유 시간과 크리티컬 패스를 계산합니다. 링크에 순환이 있으면 OnScheduleError가 발생하고 순환에 속한 작업은 날짜를 유지합니다. Community에서는 EsgcPlannerError가 발생합니다.
SetBaseline		모든 작업의 Start와 Finish를 BaselineStart와 BaselineFinish로 복사합니다. Community에서는 EsgcPlannerError가 발생합니다.
ItemAt(X, Y)	TsgcPlannerItem	컨트롤의 한 지점 아래에 있는 항목, 없으면 nil.
ScrollToItem(aId)	Boolean	항목이 보일 때까지 스크롤합니다. 항목이 표시되지 않으면 False.

SaveToPNG, SaveToSVG(aFileName 또는 aStream, aWidth = 0, aHeight = 0, aScale = 0)		뷰를 이미지로 내보냅니다. 같은 데이터, 옵션, 크기, 배율을 가진 TsgcPlannerRenderer와 같은 이미지입니다. 크기는 DIP 단위이며 0은 클라이언트 크기, 배율 0은 컨트롤의 배율입니다. 스크롤 위치는 유지됩니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

이벤트

이벤트	형식	설명
OnItemClick, OnItemDblClick	TsgcPlannerItemEvent	항목의 클릭 또는 더블 클릭: aItem.
OnItemDelete	TsgcPlannerItemAcceptEvent	선택된 항목에서의 Delete 키: aItem과 var aAccept. aAccept를 False로 설정하면 항목이 유지됩니다. 삭제된 항목은 링크도 함께 삭제하며 자식은 부모로 이동합니다.
OnSelectionChange	TNotifyEvent	이 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 선택된 항목이 바뀌었습니다.
OnLinkCreating	TsgcPlannerGanttLinkCreatingEvent	막대의 끝에서 다른 막대로 끌기: aFrom, aTo, var aKind(선택한 끝에서 결정), var aAccept. 항목 자신, 조상 또는 자손으로의 링크와 중복은 이벤트 전에 거부됩니다. Registered 에디션.
OnScheduleError	TsgcPlannerGanttScheduleErrorEvent	엔진이 순환을 발견했습니다: aError와 aCycleIds(순환에 속한 항목의 Id). 그 날짜는 유지됩니다.
OnExpandChange	TsgcPlannerGanttExpandEvent	요약이 열리거나 닫혔습니다: aItem과 aExpanded. ExpandAll과 CollapseAll은 aItem을 nil로 하여 한 번만 발생시킵니다.

예제: WBS 열, 자동 일정 계산, 크리티컬 패스

관련 항목

- TsgcFMXGantt
- TsgcPlannerData
- TsgcPlannerMSProject
- 종속성
- 자동 일정 계산
- 크리티컬 패스
- 기준선
- 시간 눈금과 확대/축소

TsgcFMXGantt

유닛: sgcPlanner_Gantt_FMX
TsgcFMXGantt 는 Delphi XE6 부터 사용할 수 있는 FireMonkey Gantt 차트입니다(SGC_PLANNER_FMX). TsgcVCLGantt와 같은 트리, 차트, 속성, 메서드, 이벤트를 갖습니다. 자체 스크롤 막대를 그리며 터치 입력을 마우스 입력으로 처리합니다. TsgcUIFMXControl에서 상속하며 일반적인 FireMonkey 레이아웃 속성과 이벤트를 게시합니다.

속성

속성	형식	설명
Data	TsgcPlannerData	컨트롤이 보여 주는 데이터. 여러 컨트롤이 하나의 데이터 객체를 공유할 수 있습니다. nil이면 컨트롤은 자체 내부 데이터 객체를 사용합니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. utCustom은 밝은 색에서 시작합니다.
Colors	TsgcPlannerViewColors	테마 색의 재정의: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 0(기본값)이면 테마의 색을 유지합니다.
ReadOnly	Boolean	모든 끌기와 Delete 키를 끕니다. 기본값 False.
DragOptions	TsgcPlannerDragOptions	사용자가 할 수 있는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다.
SyncSelection	Boolean	데이터의 SelectedId를 통해 같은 데이터의 다른 뷰와 선택을 공유합니다. 기본값 True.
MaxFPS	Integer	컨트롤이 그리는 초당 최대 프레임 수.
Columns	TsgcPlannerGanttColumns	작업 트리의 열. 각 열에는 Field(pgcId, pgcWBS, pgcTitle, pgcStart, pgcFinish, pgcDuration, pgcProgress, pgcStatus, pgcResources), Caption(비어 있으면 기본 캡션), Width, Visible, Alignment가 있습니다. 기본 열은 ID, 작업 이름, 시작, 종료, 기간, 진행률입니다. AddColumn(aField, aWidth = 100, aAlignment = taLeftJustify)과 SetDefaults.
TreeWidth	Integer	작업 트리의 너비(DIP). 기본값 550. 사용자가 분할선을 끌어 변경합니다.
RowHeight	Integer	행의 높이(DIP). 기본값 28.
TimeScale	TsgcPlannerTimeScale	시간 축: Zoom(pz5Minutes, pz15Minutes, pzHours, pzDays 기본값, pzWeeks, pzMonths, pzQuarters, pzYears), TickWidth(0이면 확대/축소 수준의 기본값), Origin, Snap(psNone, psTick 기본값, ps5Min, ps15Min, psHour, psDay), ShowNonWorking(True), 머리글 두 단의 MajorFormat과 MinorFormat.
DurationUnit	TsgcPlannerDurationUnit	기간 열의 단위: pduMinutes, pduHours, pduDays(기본값) 또는 pduWeeks, 근무 시간으로 계산됩니다. 소수점 구분 기호는 Planner 로캘의 것입니다.

BarColorMode	TsgcPlannerBarColorMode	pbcItem(기본값)은 항목과 같은 색으로, pbcStatus는 상태 색으로, pbcResource는 첫 번째 리소스 색으로 막대를 칠합니다.
ShowProgress, ShowToday	Boolean	막대 안에 진행률을, 오늘 위치에 선을 그립니다. 기본값 True.
ShowDependencies	Boolean	링크를 꺾인 화살표로 그립니다. 기본값 True. Registered 에디션.
ShowCriticalPath, ShowBaseline	Boolean	크리티컬 작업과 링크를 크리티컬 색으로, 각 막대 아래에 기준선을 그립니다. 기본값 False. Registered 에디션.
AutoSchedule	Boolean	차트에서 편집할 때마다 프로젝트 일정을 다시 계산합니다. 기본값 False. Registered 에디션.
GanttView	TsgcPlannerGanttView	Public, 읽기 전용. 행, 막대 지오메트리, 링크를 가진 뷰 모델.
View	TsgcPlannerView	Public, 읽기 전용. 컨트롤이 호스팅하는 뷰 모델.

메서드

메서드	반환값	설명
Expand(aId), Collapse(aId)	Boolean	요약 항목을 열거나 닫습니다.
ExpandAll, CollapseAll		모든 요약 항목을 열거나 닫습니다.
ZoomIn, ZoomOut	Boolean	차트 중심을 기준으로 확대/축소 수준을 한 단계 더 세밀하게 또는 더 거칠게 합니다. 마지막 수준이면 False.
ZoomToFit	Boolean	모든 행이 보이는 가장 세밀한 확대/축소 수준을 고르고 첫 시작으로 스크롤합니다. 행이 없으면 False.
ScrollToTime(aTime)		aTime이 왼쪽 가장자리에 오도록 차트를 스크롤합니다.
Schedule	Boolean	데이터에 일정 계산 엔진을 실행합니다: ProjectStart부터 근무 달력에 따라 작업을 선행 작업 뒤에 배치하고, 여유 시간과 크리티컬 패스를 계산합니다. 링크에 순환이 있으면 OnScheduleError가 발생하고 순환에 속한 작업은 날짜를 유지합니다. Community에서는 EsgcPlannerError가 발생합니다.
SetBaseline		모든 작업의 Start와 Finish를 BaselineStart와 BaselineFinish로 복사합니다. Community에서는 EsgcPlannerError가 발생합니다.
ItemAt(X, Y (Single))	TsgcPlannerItem	컨트롤의 한 지점 아래에 있는 항목, 없으면 nil.
ScrollToItem(aId)	Boolean	항목이 보일 때까지 스크롤합니다. 항목이 표시되지 않으면 False.
SaveToPNG, SaveToSVG(aFileName 또는 aStream, aWidth = 0, aHeight = 0, aScale = 0)		뷰를 이미지로 내보냅니다. 같은 데이터, 옵션, 크기, 배율을 가진 TsgcPlannerRenderer와 같은 이미지입니다. 크기는 DIP 단위이며 0은 클라이언트 크기, 배율 0은 컨트롤의 배율입니다. 스크롤 위치는 유지됩니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

이벤트

이벤트	형식	설명
OnItemClick, OnItemDblClick	TsgcPlannerItemEvent	항목의 클릭 또는 더블 클릭: aItem.
OnItemDelete	TsgcPlannerItemAcceptEvent	선택된 항목에서의 Delete 키: aItem과 var aAccept. aAccept를 False로 설정하면 항목이 유지됩니다. 삭제된 항목은 링크도 함께 삭제하며 자식은 부모로 이동합니다.
OnSelectionChange	TNotifyEvent	이 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 선택된 항목이 바뀌었습니다.
OnLinkCreating	TsgcPlannerGanttLinkCreatingEvent	막대의 끝에서 다른 막대로 끌기: aFrom, aTo, var aKind(선택한 끝에서 결정), var aAccept. 항목 자신, 조상 또는 자손으로의 링크와 중복은 이벤트 전에 거부됩니다. Registered 에디션.
OnScheduleError	TsgcPlannerGanttScheduleErrorEvent	엔진이 순환을 발견했습니다: aError와 aCycleIds(순환에 속한 항목의 Id). 그 날짜는 유지됩니다.
OnExpandChange	TsgcPlannerGanttExpandEvent	요약이 열리거나 닫혔습니다: aItem과 aExpanded. ExpandAll과 CollapseAll은 aItem을 nil로 하여 한 번만 발생시킵니다.

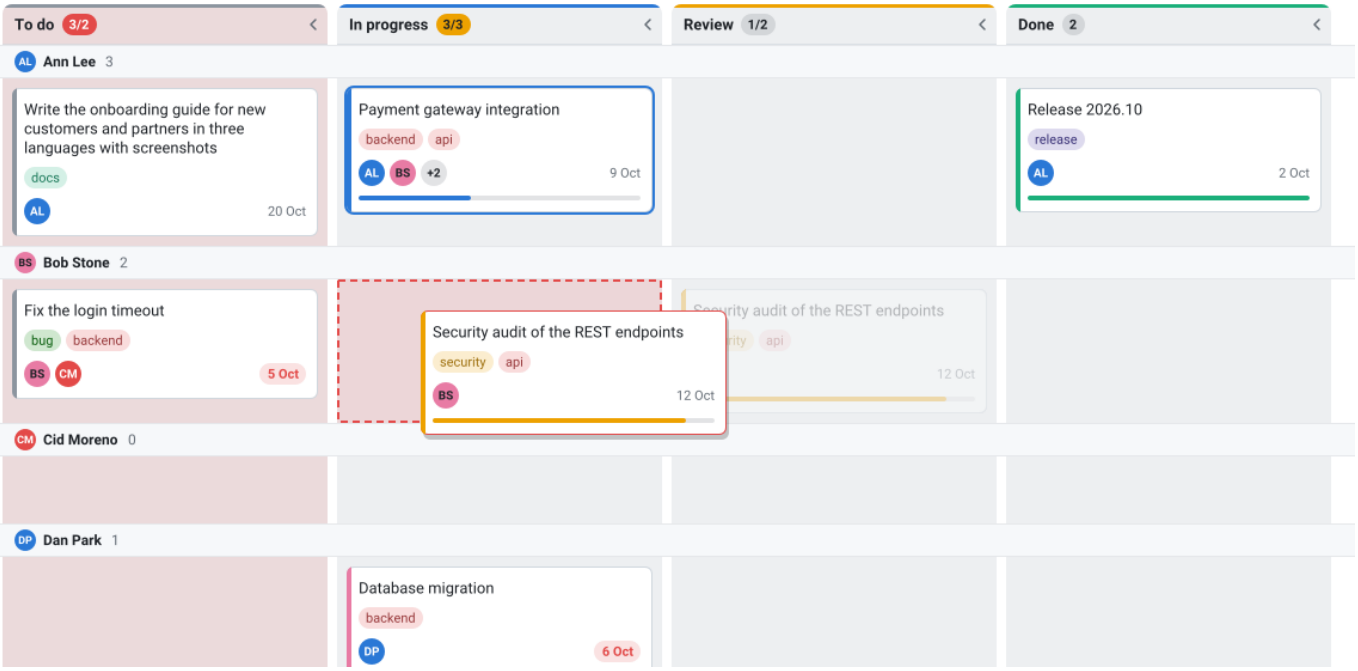
예제: 시간 단위, 15분 스냅, 열린 단계 하나

관련 항목

- TsgcVCLGantt
- FMX 빠른 시작
- 종속성
- 자동 일정 계산
- 크리티컬 패스
- 기준선
- 시간 눈금과 확대/축소

TsgcVCLKanban

유닛: sgcPlanner_Kanban_VCL
TsgcVCLKanban은 Windows 전용 VCL Kanban 보드입니다. TsgcPlannerData의 항목을 보드의 카드로 보여 줍니다: 상태마다 열 하나, 개수 배지, 선택적 WIP 제한, 접기 화살표가 있으며, Registered 에디션에서는 리소스, 부모, 태그 또는 이벤트별 스윔레인이 있습니다. 카드는 여러 줄의 제목, 태그, 리소스의 이니셜, 마감일, 진행률을 보여 줍니다. 사용자가 카드를 다른 위치, 열 또는 레인으로 끌면 놓기가 항목의 Status와 Rank를 변경합니다. 요약 항목과 수정된 발생 항목은 카드로 표시되지 않습니다. DragOptions에 pdoReorder가 있고 pdoMove가 없으면 끌기는 열 안의 카드 순서만 바꿉니다. TsgcUIVCLControl에서 상속하며 Align, Anchors, Constraints, Enabled, PopupMenu, ShowHint, ParentShowHint, TabOrder, TabStop, Visible을 게시합니다.



속성

속성	형식	설명
Data	TsgcPlannerData	컨트롤이 보여 주는 데이터. 여러 컨트롤이 하나의 데이터 객체를 공유할 수 있습니다. nil이면 컨트롤은 자체 내부 데이터 객체를 사용합니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. utCustom은 밝은 색에서 시작합니다.
Colors	TsgcPlannerViewColors	테마 색의 재정의: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 0(기본값)이면 테마의 색을 유지합니다.
ReadOnly	Boolean	모든 끌기와 Delete 키를 끕니다. 기본값 False.
DragOptions	TsgcPlannerDragOptions	사용자가 할 수 있는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다.
SyncSelection	Boolean	데이터의 SelectedId를 통해 같은 데이터의 다른 뷰와 선택을 공유합니다. 기본값 True.

MaxFPS	Integer	컨트롤이 그리는 초당 최대 프레임 수.
BorderStyle	TBorderStyle	bsSingle(기본값) 또는 bsNone.
ColumnSource	TsgcPlannerKanbanColumnSource	pkcStatuses(기본값)는 데이터의 상태마다 열 하나를 보여 주고, pkcList는 Columns 컬렉션만 보여 줍니다.
Columns	TsgcPlannerKanbanColumns	열: StatusId, Caption, Color(비어 있거나 0이면 상태의 값), WIPLimit(0이면 없음), Collapsed, Width(0이면 기본값). pkcStatuses에서는 항목이 해당 상태의 열에 설정만 추가합니다. AddColumn(aStatusId, aCaption = '', aWIPLimit = 0)과 Find(aStatusId).
ShowUnknownColumn	Boolean	상태에 열이 없는 카드를 위한 Other 열을 표시합니다. 기본값 True. 카드를 여기에 놓을 수는 없습니다.
SwimlaneBy	TsgcPlannerSwimlaneBy	pslNone(기본값), pslResource(리소스마다 레인 하나, 다른 레인에 놓으면 첫 번째 리소스를 대체), pslParent(부모마다 레인 하나, 놓으면 ParentId 설정), pslTag(태그마다 레인 하나) 또는 pslEvent(OnGetSwimlane이 레인을 지정). Registered 에디션.
WIPMode	TsgcPlannerWIPMode	pwmWarn(기본값)은 WIPLimit에 도달했거나 초과한 열의 개수 배지에만 색을 입히고, pwmBlock은 제한을 넘게 되는 카드도 거부합니다. 열 안에서 순서를 바꾸는 것은 항상 허용됩니다. Registered 에디션.
CardOptions	TsgcPlannerKanbanCardOptions	ShowTags, ShowAssignees, ShowDueDate, ShowProgress(기본값 True)와 MaxTitleLines. 마감일은 항목의 Finish이며 지났는데 카드가 완료되지 않았으면 빨간색으로 표시됩니다.
KanbanView	TsgcPlannerKanbanView	Public, 읽기 전용. 열, 레인, 카드를 가진 뷰 모델.
View	TsgcPlannerView	Public, 읽기 전용. 컨트롤이 호스팅하는 뷰 모델.

메서드

메서드	반환값	설명
CollapseColumn(aStatusId), ExpandColumn(aStatusId)	Boolean	열을 좁은 레일로 접거나 다시 엽니다. 사용자는 머리글의 화살표를 클릭합니다.
SelectNextCard(aKey)	Boolean	키 코드로 선택을 이동합니다: 화살표는 카드, 열, 레인을 이동하고, Home과 End는 첫 카드와 마지막 카드로 이동합니다.
ItemAt(X, Y)	TsgcPlannerItem	컨트롤의 한 지점 아래에 있는 항목, 없으면 nil.
ScrollToItem(aId)	Boolean	항목이 보일 때까지 스크롤합니다. 항목이 표시되지 않으면 False.

SaveToPNG, SaveToSVG(aFileName 또는 aStream, aWidth = 0, aHeight = 0, aScale = 0)		뷰를 이미지로 내보냅니다. 같은 데이터, 옵션, 크기, 배율을 가진 TsgcPlannerRenderer와 같은 이미지입니다. 크기는 DIP 단위이며 0은 클라이언트 크기, 배율 0은 컨트롤의 배율입니다. 스크롤 위치는 유지됩니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

이벤트

이벤트	형식	설명
OnItemClick, OnItemDblClick	TsgcPlannerItemEvent	항목의 클릭 또는 더블 클릭: aItem.
OnItemDelete	TsgcPlannerItemAcceptEvent	선택된 항목에서의 Delete 키: aItem과 var aAccept. aAccept를 False로 설정하면 항목이 유지됩니다. 삭제된 항목은 링크도 함께 삭제하며 자식은 부모로 이동합니다.
OnSelectionChange	TNotifyEvent	이 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 선택된 항목이 바뀌었습니다.
OnCardMoving	TsgcPlannerKanbanCardMovingEvent	놓기 전에 발생합니다: aItem, aToStatus, aToLane, var aAccept. 데이터의 OnItemChanging보다 먼저 발생합니다.
OnCardMoved	TsgcPlannerItemEvent	놓기가 커밋된 후에 발생합니다: 새 상태와 순위를 가진 aItem.
OnGetSwimlane	TsgcPlannerKanbanSwimlaneEvent	pslEvent에서: aItem과 var aKey, aCaption. 카드의 레인.
OnColumnCollapse	TsgcPlannerKanbanColumnEvent	열이 접히거나 열렸습니다: aStatusId와 aCollapsed.

예제: WIP 제한, 사람별 레인, Done 규칙

관련 항목

- TsgcFMXKanban
- TsgcPlannerData
- VCL 빠른 시작
- 스윔레인과 WIP 제한
- 끌어서 놓기
- 테마

TsgcFMXKanban

유닛: sgcPlanner_Kanban_FMX
TsgcFMXKanban 은 Delphi XE6 부터 사용할 수 있는 FireMonkey Kanban 보드입니다(SGC_PLANNER_FMX). TsgcVCLKanban과 같은 보드, 속성, 메서드, 이벤트를 갖습니다. 자체 스크롤 막대를 그리며 터치 입력을 마우스 입력으로 처리하므로 손가락으로 카드를 끌 수 있습니다. TsgcUIFMXControl에서 상속하며 일반적인 FireMonkey 레이아웃 속성과 이벤트를 게시합니다.

속성

속성	형식	설명
Data	TsgcPlannerData	컨트롤이 보여 주는 데이터. 여러 컨트롤이 하나의 데이터 객체를 공유할 수 있습니다. nil이면 컨트롤은 자체 내부 데이터 객체를 사용합니다.
Theme	TsgcUITheme	utLight(기본값), utDark 또는 utCustom. utCustom은 밝은 색에서 시작합니다.
Colors	TsgcPlannerViewColors	테마 색의 재정의: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 0(기본값)이면 테마의 색을 유지합니다.
ReadOnly	Boolean	모든 끌기와 Delete 키를 끕니다. 기본값 False.
DragOptions	TsgcPlannerDragOptions	사용자가 할 수 있는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다.
SyncSelection	Boolean	데이터의 SelectedId를 통해 같은 데이터의 다른 뷰와 선택을 공유합니다. 기본값 True.
MaxFPS	Integer	컨트롤이 그리는 초당 최대 프레임 수.
ColumnSource	TsgcPlannerKanbanColumnSource	pkcStatuses(기본값)는 데이터의 상태마다 열 하나를 보여 주고, pkcList는 Columns 컬렉션만 보여 줍니다.
Columns	TsgcPlannerKanbanColumns	열: StatusId, Caption, Color(비어 있거나 0이면 상태의 값), WIPLimit(0이면 없음), Collapsed, Width(0이면 기본값). pkcStatuses에서는 항목이 해당 상태의 열에 설정만 추가합니다. AddColumn(aStatusId, aCaption = '', aWIPLimit = 0)과 Find(aStatusId).
ShowUnknownColumn	Boolean	상태에 열이 없는 카드를 위한 Other 열을 표시합니다. 기본값 True. 카드를 여기에 놓을 수는 없습니다.
SwimlaneBy	TsgcPlannerSwimlaneBy	pslNone(기본값), pslResource(리소스마다 레인 하나, 다른 레인에 놓으면 첫 번째 리소스를 대체), pslParent(부모마다 레인 하나, 놓으면 ParentId 설정), pslTag(태그마다 레인 하나) 또는 pslEvent(OnGetSwimlane이 레인을 지정). Registered 에디션.

WIPMode	TsgcPlannerWIPMode	pwmWarn(기본값)은 WIPLimit에 도달했거나 초과한 열의 개수 배지에만 색을 입히고, pwmBlock은 제한을 넘게 되는 카드도 거부합니다. 열 안에서 순서를 바꾸는 것은 항상 허용됩니다. Registered 에디션.
CardOptions	TsgcPlannerKanbanCardOptions	ShowTags, ShowAssignees, ShowDueDate, ShowProgress(기본값 True)와 MaxTitleLines. 마감일은 항목의 Finish이며 지났는데 카드가 완료되지 않았으면 빨간색으로 표시됩니다.
KanbanView	TsgcPlannerKanbanView	Public, 읽기 전용. 열, 레인, 카드를 가진 뷰 모델.
View	TsgcPlannerView	Public, 읽기 전용. 컨트롤이 호스팅하는 뷰 모델.

메서드

메서드	반환값	설명
CollapseColumn(aStatusId), ExpandColumn(aStatusId)	Boolean	열을 좁은 레일로 접거나 다시 엽니다. 사용자는 머리글의 화살표를 클릭합니다.
SelectNextCard(aKey)	Boolean	키 코드로 선택을 이동합니다: 화살표는 카드, 열, 레인을 이동하고, Home과 End는 첫 카드와 마지막 카드로 이동합니다.
ItemAt(X, Y (Single))	TsgcPlannerItem	컨트롤의 한 지점 아래에 있는 항목, 없으면 nil.
ScrollToItem(aId)	Boolean	항목이 보일 때까지 스크롤합니다. 항목이 표시되지 않으면 False.
SaveToPNG, SaveToSVG(aFileName 또는 aStream, aWidth = 0, aHeight = 0, aScale = 0)		뷰를 이미지로 내보냅니다. 같은 데이터, 옵션, 크기, 배율을 가진 TsgcPlannerRenderer와 같은 이미지입니다. 크기는 DIP 단위이며 0은 클라이언트 크기, 배율 0은 컨트롤의 배율입니다. 스크롤 위치는 유지됩니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

이벤트

이벤트	형식	설명
OnItemClick, OnItemDblClick	TsgcPlannerItemEvent	항목의 클릭 또는 더블 클릭: aItem.
OnItemDelete	TsgcPlannerItemAcceptEvent	선택된 항목에서의 Delete 키: aItem과 var aAccept. aAccept를 False로 설정하면 항목이 유지됩니다. 삭제된 항목은 링크도 함께 삭제하며 자식은 부모로 이동합니다.
OnSelectionChange	TNotifyEvent	이 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 선택된 항목이 바뀌었습니다.

OnCardMoving	TsgcPlannerKanbanCardMovingEvent	놓기 전에 발생합니다: aItem, aToStatus, aToLane, var aAccept. 데이터의 OnItemChanging보다 먼저 발생합니다.
OnCardMoved	TsgcPlannerItemEvent	놓기가 커밋된 후에 발생합니다: 새 상태와 순위를 가진 aItem.
OnGetSwimlane	TsgcPlannerKanbanSwimlaneEvent	pslEvent에서: aItem과 var aKey, aCaption. 카드의 레인.
OnColumnCollapse	TsgcPlannerKanbanColumnEvent	열이 접히거나 열렸습니다: aStatusId와 aCollapsed.

예제: 고정된 열 목록과 접힌 Done 열

관련 항목

- TsgcVCLKanban
- FMX 빠른 시작
- 스윔레인과 WIP 제한
- 끌어서 놓기

TsgcPlannerSource_DataSet

유닛: sgcPlanner_Source_DataSet
TsgcPlannerSource_DataSet는 TsgcPlannerData를 TDataSet에 바인딩합니다: DataSet은 항목을, ResourceDataSet은 리소스를, DependencyDataSet은 종속성을 담습니다. DB 유닛만 필요하므로 VCL, FireMonkey, 서비스 어디서든 모든 데이터셋과 함께 동작합니다. 모든 에디션에 포함됩니다.

로드

- Active를 True로 하면 열려 있는 모든 DataSet을 로드하고, Active인 동안 DataSet을 열면 그것을 로드합니다. 로드는 항목, 리소스, 종속성을 한 번의 업데이트로 교체하므로 바인딩된 모든 뷰가 알림을 한 번 받습니다. 이후에 각 DataSet의 현재 레코드가 복원됩니다.
- 날짜는 날짜 및 시간 필드, 숫자 필드 또는 ISO 8601 텍스트에서 가져옵니다. 불리언은 불리언 또는 숫자 필드나 텍스트(1, T, True, Y, Yes)에서 가져옵니다. 색은 숫자 필드(ARGB) 또는 텍스트(#RRGGBB, #AARRGGBB, $AARRGGBB)에서 가져옵니다. null은 기본값을 주며, null인 Finish는 Finish = Start가 되고, 종일 항목이면 다음 자정이 됩니다.
- DependencyDataSet이 없으면 DependsOnField가 항목의 선행 작업 Id를 쉼표로 구분해 담으며, 종료-시작 링크로 읽습니다.

실시간 업데이트와 되쓰기

- LiveUpdates를 사용하면 게시된 편집이나 삽입이 해당 항목 하나를 업데이트하거나 추가하며, 레코드 수의 변경(삭제, 필터, 새로 고침)은 Id로 항목을 동기화합니다. 스크롤과 게시되지 않은 편집은 아무것도 바꾸지 않습니다.
- WriteBack을 사용하면 뷰나 코드에서 한 편집이 DataSet에 기록됩니다: Id 필드에서 Locate로 레코드를 찾고, 변경된 매핑 필드를 할당한 뒤 즉시 게시합니다. 새 항목은 레코드를 추가하고 삭제된 항목은 해당 레코드를 삭제합니다. 리소스, 계산 필드, 조회 필드, 읽기 전용 필드는 기록되지 않습니다. DataSet이 편집 또는 삽입 상태인 동안에는 아무것도 기록되지 않으며, 보류 중인 변경은 찾아보기 상태로 돌아오면 기록됩니다.

속성

속성	형식	설명
Data	TsgcPlannerData	채울 데이터.
DataSet	TDataSet	항목, 항목마다 레코드 하나.
ItemFields	TsgcPlannerSourceItemFields	각 항목 값의 필드: IdField, TitleField, StartField, FinishField, AllDayField, StatusField, ProgressField, ColorField, ParentField, TagsField, ResourcesField, RankField, NotesField, RecurrenceField, DependsOnField. 기본값은 Id, Title, Start, Finish, AllDay, Status, Progress, Color, ParentId, Tags, Resources, Rank, Notes, RecurrenceRule, DependsOn입니다. 비어 있거나 DataSet에서 찾을 수 없는 이름은 무시됩니다.
ResourceDataSet, ResourceFields	TDataSet, TsgcPlannerSourceResourceFields	리소스: IdField, NameField, ColorField, InitialsField, 기본값 Id, Name, Color, Initials. 읽기 전용.
DependencyDataSet, DependencyFields	TDataSet, TsgcPlannerSourceDependencyFields	링크: FromField, ToField, KindField(FS, SS, FF, SF 또는 서수), LagField(작업 분), 기본값 FromId, ToId, Kind, Lag.

Active	Boolean	열려 있는 DataSet을 로드하고 따라갑니다. 기본값 False.
LiveUpdates	Boolean	DataSet의 게시, 삽입, 삭제, 새로 고침을 따라갑니다. 기본값 True.
WriteBack	Boolean	데이터의 변경을 DataSet과 DependencyDataSet에 기록합니다. 기본값 True.

메서드

메서드	반환값	설명
Load		Active 여부와 관계없이 열려 있는 모든 DataSet을 지금 로드합니다. Data가 nil이면 아무 일도 일어나지 않습니다.
Refresh		Active이면 다시 로드합니다.

이벤트

이벤트	형식	설명
OnLoadItem	TsgcPlannerSourceLoadItemEvent	레코드의 매핑된 필드를 항목으로 복사한 후에 발생합니다: 해당 레코드 위의 aDataSet과 aItem으로, 애플리케이션이 다른 항목 속성을 채울 수 있습니다.

예제: 두 테이블의 작업과 사람

관련 항목

- TsgcPlannerData
- VCL 빠른 시작
- DataSet 바인딩

TsgcPlannerICalendar

유닛: sgcPlanner_ICal
TsgcPlannerICalendar 는 Google Calendar, Outlook, Apple Calendar 의 형 식 인 iCalendar 파 일(RFC 5545)의 VEVENT를 TsgcPlannerData로 가져오고 그로부터 내보냅니다. Registered 에디션(SGC_PLANNER_ADVANCED)에 포함됩니다.

- 항목의 Id는 이벤트의 UID입니다. RECURRENCE-ID가 있는 이벤트는 해당 시리즈의 수정된 발생 항목이 됩니다.
- UTC(Z) 및 TZID가 있는 시간은 파일의 VTIMEZONE을 사용해 로컬 벽시계 시간으로 변환됩니다. 알 수 없는 TZID는 경고 하나를 내고 로컬 시간으로 읽습니다.
- STATUS는 항목의 Status로, RESOURCES는 리소스로(없는 리소스는 생성됨), RELTYPE=PARENT가 있는 RELATED-TO는 ParentId로, RRULE과 EXDATE는 반복으로 매핑됩니다. 나머지 항목 필드는 X-SGC 속성으로 전달되므로 내보낸 뒤 가져오면 같은 데이터가 됩니다. 종속성은 내보내지 않습니다.
- BEGIN:VCALENDAR가 없는 텍스트는 EsgcPlannerError를 발생시키고 데이터를 변경하지 않습니다. 가져오기는 하나의 업데이트이므로 뷰는 알림을 한 번 받습니다.

속성

속성	형식	설명
Data	TsgcPlannerData	가져오거나 내보낼 데이터.
ImportMode	TsgcPlannerICalImportMode	pimReplace(기본값)는 먼저 항목과 종속성을 삭제하고 리소스, 상태, 달력, ProjectStart는 유지합니다. pimMerge는 같은 Id의 항목을 업데이트하고 나머지를 추가합니다.
ExportUTC	Boolean	반복되지 않는 이벤트의 시간을 UTC로 기록합니다. 시리즈와 수정된 발생 항목은 항상 부동 로컬 시간으로 기록되어 일광 절약 시간 변경에도 올바르게 유지됩니다. 기본값 True.
ProductId	string	내보낸 달력의 PRODID. 기본값 `-//eSeGeCe//sgcPlanner//EN`.
WarningCount	Integer	Public, 읽기 전용. 마지막 가져오기의 경고.

메서드

메서드	반환 값	설명
ImportFromFile, ImportFromStream, ImportFromString		달력을 읽습니다. 파일과 스트림은 BOM 유무와 관계없이 UTF-8입니다. 잘못된 UTF-8은 경고와 함께 Latin-1로 읽습니다.
ExportToFile, ExportToStream		데이터를 하나의 달력으로 UTF-8로 기록합니다.
ExportToString	string	iCalendar 텍스트로 된 데이터.

이벤트

이벤트	형식	설명

OnImportWarning	TsgcPlannerICalWarningEvent	건너뛰었거나 근사적으로 읽은 파일의 일부: aLine과 aMessage. 예를 들어 지원되는 하위 집합 밖의 RRULE, RDATE 또는 알 수 없는 시간대.

예제: 달력을 병합하고 다시 기록하기

관련 항목

- TsgcPlannerMSProject
- TsgcPlannerData
- 에디션
- iCalendar
- 반복

TsgcPlannerMSProject

유닛: sgcPlanner_MSProject
TsgcPlannerMSProject는 Microsoft Project가 저장하고 여는 XML 형식인 Microsoft Project XML 파일(MSPDI)을 TsgcPlannerData로 가져오고 그로부터 내보냅니다. Registered 에디션(SGC_PLANNER_ADVANCED)에 포함됩니다.

- 작업은 항목이 됩니다: Id는 UID이고, 개요가 ParentId를 결정하며, 날짜, 진행률, 마일스톤, 수동 플래그, 제약, 기준선은 파일을 따릅니다. 선행 작업 링크는 종류와 지연이 있는 종속성이 되고, 할당이 각 작업의 리소스를 결정합니다.
- 프로젝트 달력이 근무일, 근무 시간, 휴일을 결정하고, 프로젝트 시작일이 ProjectStart를 결정합니다.
- sgcPlanner에는 ASAP와 ConstraintDate보다 일찍 시작하지 않음 제약만 있습니다. 지정일에 시작은 일찍 시작하지 않음으로, 나머지 제약 유형은 ASAP로 읽으며 각각 경고가 표시됩니다.
- 내보내기는 Standard 달력 하나, 개요 순서의 작업, 링크, 리소스를 기록합니다. 상태, 태그, 색, 순위, 종일, 반복은 기록되지 않습니다.
- 형식이 올바르지 않은 문서는 줄 번호와 함께 EsgcPlannerError를 발생시키고 데이터를 변경하지 않습니다.

속성

속성	형식	설명
Data	TsgcPlannerData	가져오거나 내보낼 데이터.
ImportMode	TsgcPlannerMSProjectImportMode	pmiReplace(기본값)는 항목과 링크를 삭제하고 달력과 프로젝트 시작일을 파일에서 가져옵니다. 리소스와 상태는 유지되며 파일의 리소스가 추가됩니다. pmiMerge는 달력과 ProjectStart를 유지하고, 같은 Id의 항목을 업데이트하고 나머지를 추가합니다.
WarningCount	Integer	Public, 읽기 전용. 마지막 가져오기의 경고.

메서드

메서드	반환 값	설명
ImportFromFile, ImportFromStream, ImportFromString		프로젝트를 읽습니다. 파일과 스트림은 BOM 유무와 관계없이 UTF-8입니다.
ExportToFile, ExportToStream		데이터를 프로젝트로 기록합니다.
ExportToString	string	MSPDI 텍스트로 된 데이터.

이벤트

이벤트	형식	설명
OnImportWarning	TsgcPlannerMSProjectWarningEvent	근사적으로 읽은 파일의 일부: aLine과 aMessage.

예제: Microsoft Project 파일 열고 저장하기

관련 항목

- TsgcVCLGantt
- TsgcPlannerICalendar
- TsgcPlannerData
- MS Project XML
- 기준선

TsgcPlannerRenderer

유닛: sgcPlanner_Renderer
TsgcPlannerRenderer는 TsgcPlannerData의 Scheduler, Gantt 또는 Kanban 뷰를 창 없이 임의의 TsgcUICanvas, PNG 또는 SVG로 그리므로, 서버, 보고서 또는 콘솔 애플리케이션이 폼이 보여 주는 것과 같은 그림을 만듭니다. 컨트롤이 만드는 것과 같은 클래스인 각 종류의 뷰 모델을 하나씩 소유하므로 출력은 컨트롤과 픽셀 단위로 일치합니다. 프레임워크에 독립적이며 모든 에디션에서 Windows와 Linux64에서 실행됩니다.

속성

속성	형식	설명
Data	TsgcPlannerData	그릴 데이터. 선택은 데이터의 SelectedId입니다.
View	TsgcPlannerRendererViewKind	prvScheduler(기본값), prvGantt 또는 prvKanban.
Date	TDateTime	Scheduler의 날짜. 0(기본값)이면 시계를 따릅니다.
Width, Height	Integer	DIP 단위의 크기. 기본값 800 x 600.
Scale	Double	HiDPI 배율. 출력은 Round(Width * Scale) x Round(Height * Scale) 픽셀입니다. 기본값 1.
Theme, Colors	TsgcUITheme, TsgcPlannerViewColors	컨트롤과 같습니다.
FontFile, FontFileBold	string	텍스트를 측정하고 그릴 TrueType 파일. 비어 있으면(기본값) 내장 글꼴을 사용합니다. Kanban은 FontFile로 카드 제목을 줄바꿈합니다.
SchedulerOptions	TsgcPlannerRendererSchedulerOptions	TsgcVCLScheduler와 같은 Scheduler의 View, Options, TimeScale.
GanttOptions	TsgcPlannerRendererGanttOptions	TsgcVCLGantt와 같은 Columns, TreeWidth, RowHeight, TimeScale, DurationUnit, BarColorMode, ShowProgress, ShowToday, ShowDependencies, ShowCriticalPath, ShowBaseline.
KanbanOptions	TsgcPlannerRendererKanbanOptions	TsgcVCLKanban와 같은 ColumnSource, Columns, ShowUnknownColumn, SwimlaneBy, WIPMode, CardOptions.
SchedulerView, GanttView, KanbanView		Public, 읽기 전용. 옵션이 다루지 않는 항목(예: ExpandAll, Collapse, ScrollToTime)을 위한 뷰 모델. 크기는 Width와 Height를 따르므로 렌더링 전에 설정한 스크롤은 유지됩니다.
ActiveView	TsgcPlannerView	Public, 읽기 전용. View의 뷰 모델.

각 옵션 클래스는 뷰 모델의 옵션도 복사하므로 `GanttOptions.Assign(VCLGantt1.GanttView)`는 해당 컨트롤이 보여 주는 것을 렌더링합니다.

메서드

메서드	반환값	설명
RenderTo(aCanvas)		DeviceWidth x DeviceHeight 픽셀의 TsgcUICanvas에 뷰를 그립니다.
SaveToPNG(aFileName 또는 aStream)		PNG로 된 뷰.
SaveToSVG(aFileName 또는 aStream)		SVG로 된 뷰.
DeviceWidth, DeviceHeight	Integer	출력의 크기(픽셀).

이벤트

TsgcPlannerRenderer에는 이벤트가 없습니다. 데이터가 바뀌면 데이터의 OnItemChanging과 OnChange 이벤트가 평소처럼 발생합니다.

예제: SVG로 된 Gantt와 HiDPI PNG로 된 어두운 월 보기

관련 항목

- 아키텍처
- TsgcPlannerData
- TsgcVCLGantt
- 헤드리스 렌더링
- 테마

공유 데이터

모든 sgcPlanner 뷰는 TsgcPlannerData를 보여 줍니다. 데이터 객체가 Planner의 모든 것, 즉 항목, 리소스, 상태, 종속성, 근무 달력, ProjectStart, 선택을 보관합니다. 뷰는 그 복사본을 갖지 않고 읽기만 하므로 Gantt, Kanban 보드, Scheduler가 같은 항목을 동시에 보여 줄 수 있으며, 그중 하나에서 또는 코드에서 한 변경이 모두에 전달됩니다.

데이터가 보관하는 것

부분	내용
Items	TsgcPlannerItem: Id, Title, Start, Finish, AllDay, Status, Progress, Color, ParentId, Tags, Resources, Rank, Notes, Milestone, Manual, Constraint, ConstraintDate, 기준선 날짜, 반복. 항목은 Gantt에서는 작업, 보드에서는 카드, Scheduler에서는 이벤트입니다.
Resources	사람 또는 회의실: Id, Name, Color, Initials. 항목은 쉼표로 구분한 Resources 목록에서 Id로 리소스를 지정합니다.
Statuses	워크플로: Id, Caption, Color. 보드의 열이며 막대와 이벤트에 색을 입힐 수 있습니다.
Dependencies	항목 사이의 링크: FromId, ToId, Kind, Lag. 종속성을 참조하십시오.
Calendar	근무일, 근무 시간, 휴일. 근무 달력을 참조하십시오.
ProjectStart, SelectedId	일정 계산 엔진을 위한 프로젝트 시작, 그리고 SyncSelection이 있는 모든 뷰의 선택된 항목.

항목은 FindItem과 IndexOfItem으로 Id를 통해 찾습니다. ParentId가 작업 트리를 만듭니다: ChildrenOf('')는 루트를 반환하고, 요약 항목은 UpdateSummaries로 잎에서 날짜와 진행률을 가져옵니다. DeleteItem은 항목의 링크와 그 수정된 발생 항목을 제거하고 자식을 부모로 이동합니다. MoveItem은 카드의 상태와 순위를 한 번의 호출로 바꿉니다.

변경 알림

모든 setter가 알립니다. BeginUpdate와 EndUpdate 사이에는 변경이 수집되며, 가장 바깥쪽 EndUpdate 이후 모든 리스너가 TsgcPlannerChange 하나를 받습니다:

필드	의미
Kinds	변경의 집합: pckItems(항목의 필드), pckStructure(항목 추가 또는 삭제, Id, ParentId 또는 순서 변경), pckResources, pckStatuses, pckDependencies, pckCalendar, pckSelection, pckSchedule(ProjectStart 또는 엔진의 TotalSlack과 Critical 결과).
ItemId	정확히 한 항목이 바뀌었으면 그 항목의 Id, 아니면 빈 값.
Version	선택만 바뀌는 경우를 제외한 모든 변경 때마다 증가하는 숫자. 두 버전이 같으면 같은 데이터입니다.

뷰는 리스너입니다. 선택만 바뀌면 오버레이만 다시 그리고, 다른 변경은 다음 프레임에 레이아웃을 다시 계산합니다. 사용자 코드는 OnChange 이벤트 또는 핸들러를 여러 개 받는 AddListener로 수신합니다. 핸들러를 소유한 객체를 해제하기 전에 RemoveListener를 호출하십시오. 데이터를 다시 변경하는 핸들러는 알림 라운드를 한 번 더 시작하며, 8라운드 후에는 EsgcPlannerError가 발생하므로 서로를 계속 바꾸는 두 핸들러가 애플리케이션을 멈추게 하지 않습니다.

예제

관련 항목

- 동기화된 뷰
- TsgcPlannerData
- 아키텍처

동기화된 뷰

둘 이상의 뷰를 같은 TsgcPlannerData에 바인딩하면 보조를 맞춥니다. 보드에서 Done으로 끈 카드는 Gantt와 Scheduler에서 즉시 Done 색으로 그려지고, Gantt에서 이동한 막대는 Scheduler의 이벤트를 이동하며, Scheduler에서 만든 이벤트는 카드가 됩니다. 각 편집은 한 번의 업데이트로 커밋되므로 놓을 때마다 모든 뷰가 한 번만 다시 그려집니다.

공유 선택

기본값이 True인 SyncSelection을 사용하면 뷰의 선택은 데이터의 SelectedId입니다. 막대를 클릭하면 보드와 Scheduler에서도 같은 항목이 선택됩니다. SyncSelection이 False이면 뷰는 자체 선택을 유지합니다. 코드에서 SelectedId를 설정하면 동기화된 모든 뷰에서 항목이 선택됩니다. OnSelectionChange는 컨트롤에서, 또는 SyncSelection이 있으면 같은 데이터의 다른 뷰에서 한 선택에 대해 발생하며, ScrollToItem으로 다른 뷰를 그 항목으로 스크롤하기에 알맞은 위치입니다.

여러 폼의 뷰

TsgcPlannerData를 데이터 모듈에 두고 모든 폼의 뷰에 할당하십시오. 보이지 않는 폼의 뷰도 업데이트되며, 표시될 때 현재 상태를 그립니다.

폼 밖에서의 같은 그림

뷰의 옵션은 Assign으로 TsgcPlannerRenderer에 복사할 수 있습니다. 예를 들어 `Renderer.GanttOptions.Assign(VCLGantt1.GanttView)`이며, 보고서는 폼이 보여 주는 것을 보여 줍니다. 컨트롤은 PNG와 SVG로 스스로 저장하기도 합니다. 헤드리스 렌더링을 참조하십시오.

예제

관련 항목

- 공유 데이터
- 끌어서 놓기
- VCL 빠른 시작

근무 달력

TsgcPlannerData.Calendar는 Planner의 근무 달력입니다: 근무하는 요일, 하루의 근무 시간, 휴일. 기간, 지연, Gantt의 이동, 일정 계산 엔진은 여기서 근무 분을 계산하므로, 금요일에 시작하는 이틀짜리 근무 작업은 월요일 저녁에 끝납니다.

속성

속성	설명
WorkDays	근무하는 요일, pwdMonday에서 pwdSunday까지의 집합. 기본값은 월요일부터 금요일. 빈 집합은 매일이 근무일임을 뜻합니다.
WorkTime	근무일의 근무 시간, 쉼표로 구분한 최대 4개의 `hh:nn-hh:nn` 구간. 기본값 `08:00-12:00,13:00-17:00`. 빈 텍스트는 하루 전체를 뜻합니다. 잘못된 텍스트는 EsgcPlannerError를 발생시키고, 올바른 텍스트는 표준 형식으로 저장됩니다.
Holidays	근무하지 않는 날짜, 각각 Name이 있습니다. AddHoliday(Date, Name)로 추가합니다. 휴무일에 있는 휴일은 무시됩니다.
FirstDayOfWeek	Scheduler의 한 주와 시간 눈금의 주 눈금에서 주의 첫 요일. 기본값 pwdMonday.

근무 시간 연산

메서드는 모델의 정확한 Int64 분 수인 TsgcPlannerTime을 받고 반환합니다. sgcPlannerTimeOf와 sgcPlannerDateTimeOf는 TDateTime과 서로 변환합니다.

메서드	반환값
IsWorkingDay(aTime), IsWorkingTime(aTime)	해당 날 또는 해당 분이 근무인지 여부.
WorkingMinutesPerDay	하루 근무일의 근무 분, 기본 WorkTime에서는 480.
NextWorkingTime(aTime), PrevWorkingTime(aTime)	aTime 또는 그 이후의 첫 번째 근무 분, 또는 그 이전의 마지막 근무 분.
AddWorkingMinutes(aTime, aMinutes)	aMinutes 근무 분 뒤의 시간. 양수는 마지막 근무 분의 끝을 반환하므로 월요일 08:00에 480을 더하면 화요일 08:00이 아니라 월요일 17:00입니다. 음수는 거꾸로 가며 마지막 분의 시작을 반환합니다.
WorkingMinutesBetween(aStart, aFinish)	[aStart, aFinish)의 근무 분. aFinish가 aStart보다 앞이면 음수.
GetNonWorkingIntervals(aStart, aFinish, aList)	시간 눈금이 그리는 기간의 밤, 휴무일, 휴일.

달력이 사용되는 곳

- Gantt는 기간을 근무 시간으로 표시하고(DurationUnit), 이동한 막대의 근무 기간을 유지하며, TimeScale.ShowNonWorking이 True이면 비근무 시간에 음영을 입힙니다.
- 종속성의 Lag는 작업 분 단위이며, 일정 계산 엔진은 모든 작업을 근무 시간에 배치합니다. 자동 일정 계산을 참조하십시오.
- UpdateSummaries는 각 잎의 진행률을 그 근무 기간으로 가중합니다.
- Scheduler의 주중 뷰는 WorkDays의 요일을 보여 줍니다.

달력이 바뀌면 pckCalendar를 통지하므로 뷰가 다시 그려지며, JSON과 MS Project XML로 저장됩니다.

예제

관련 항목

- 자동 일정 계산
- 종속성
- 시간 눈금과 확대/축소
- TsgcPlannerData

반복

에디션: Registered. Community 에디션에서는 반복 항목이 자신의 Start에 한 번만 표시되며, 그 규칙은 유지되고 저장됩니다. 에디션별 기능을 참조하십시오.
RecurrenceRule이 있는 항목은 시리즈입니다: iCalendar(RFC 5545) RRULE의 날짜마다 반복됩니다. Scheduler는 모든 발생 항목을 보여 주고, 데이터의 GetOccurrences는 임의 기간의 발생 항목을 반환하며, iCalendar 파일로 가져오고 내보냅니다.

규칙

RecurrenceRule은 규칙의 텍스트를 담으며 `RRULE:` 접두사는 있어도 없어도 됩니다. 항목의 Start와 Finish가 첫 번째 발생 항목이며, 모든 발생 항목은 같은 길이를 갖습니다.

부분	지원
FREQ	DAILY, WEEKLY, MONTHLY 또는 YEARLY.
INTERVAL	n 기간마다. 예를 들어 격주.
COUNT, UNTIL	발생 횟수 또는 마지막 날짜. UNTIL은 날짜, 로컬 날짜 시간 또는 UTC 날짜 시간을 받습니다.
BYDAY	요일. 예: MO,WE,FR. MONTHLY와 YEARLY 규칙에서는 요일에 -5에서 5까지의 서수를 붙일 수 있으므로 2TU는 둘째 화요일, -1FR은 마지막 금요일입니다. DAILY와 함께 쓰면 날을 걸러냅니다.
BYMONTH, BYMONTHDAY	한 해의 월, 그리고 한 달의 하루(MONTHLY와 YEARLY). BYDAY가 있는 YEARLY 규칙에는 BYMONTH가 필요합니다.
WKST	규칙의 한 주의 첫 요일.

첫 번째 발생 항목은 항상 항목의 Start이며 COUNT에 포함됩니다. 잘못된 규칙은 항목을 한 번만 표시하게 합니다. 사용자가 입력한 규칙을 확인하려면 TsgcPlannerRecurrence의 TryParse를 호출하십시오. 이 함수는 False와 그 이유를 반환합니다.

예외와 수정된 발생 항목

- **RecurrenceExceptions**는 ISO 날짜 또는 날짜 시간을 쉼표로 구분한 목록입니다. 날짜는 그 날 전체의 발생 항목을 제거하고, 날짜 시간은 그 분에 시작하는 발생 항목을 제거합니다. 제거된 발생 항목도 COUNT에는 계속 포함됩니다.
- **수정된 발생 항목**은 RecurrenceParentId가 시리즈의 Id로, RecurrenceDate가 대체하는 발생 항목의 시작으로 설정된 독립된 항목입니다. 다른 시간, 다른 제목, 다른 리소스를 가질 수 있으며, 그 날짜에 생성된 발생 항목은 더 이상 표시되지 않습니다.
- 시리즈의 DeleteItem은 그 수정된 발생 항목도 삭제합니다.

Scheduler에서 편집하기

생성된 발생 항목 위에 놓으면 시리즈, 발생 항목, 범위와 함께 OnRecurrenceEdit이 발생합니다: prsOccurrence(기본값)는 수정된 발생 항목을 만들고, prsSeries는 전체 시리즈를 같은 양만큼 이동하며, prsCancel은 놓기를 취소합니다. 발생 항목에서의 Delete 키는 같은 범위로 OnRecurrenceDelete를 발생시킵니다: prsOccurrence(기본값)는 그 발생 항목만 제거하며, 그 날짜를 시리즈의 예외에 추가하고 수정된 발생 항목이면 삭제하고, prsSeries는 전체 시리즈를 삭제합니다.
일정 계산 엔진은 시리즈와 그 수정된 발생 항목의 날짜를 유지합니다. Kanban 보드는 수정된 발생 항목을 카드로 표시하지 않습니다.

예제

관련 항목

- TsgcVCLScheduler
- iCalendar
- TsgcPlannerData
- 근무 달력

시간 눈금과 확대/축소

Gantt 와 Scheduler 의 리소스 Timeline 은 컨트롤의 TimeScale 속성(렌더러에서는 GanttOptions.TimeScale 과 SchedulerOptions.TimeScale)인 TsgcPlannerTimeScale로 기술되는 가로 축에 시간을 그립니다. 시간을 픽셀로 매핑하고, 레이블이 있는 두 단의 눈금을 그리며, 비근무 시간에 음영을 입히고, 끌기를 맞춥니다.

확대/축소 수준

Zoom	작은 눈금	큰 눈금	기본 너비	레이블(작은, 큰)
pz5Minutes	5분	시간	20	`nn` , `ddd d mmm t`
pz15Minutes	15분	시간	30	`nn` , `ddd d mmm t`
pzHours	시간	일	40	`hh` , `dddd d mmmm yyyy`
pzDays (default)	일	주	32	`d` , `d mmm yyyy`
pzWeeks	주	월	56	`d mmm` , `mmmm yyyy`
pzMonths	월	년	60	`mmm` , `yyyy`
pzQuarters	분기	년	60	`"Q"q` , `yyyy`
pzYears	년	10년	60	`yyyy` , `yyyy`

축은 시간에 대해 선형입니다. 거친 확대/축소에서는 한 달을 30일, 한 분기를 91일, 한 해를 365일로 계산하므로, 길이가 다른 달의 눈금은 너비가 약간 다릅니다.

속성

속성	설명
Zoom	확대/축소 수준. 기본값 pzDays.
TickWidth	작은 눈금의 너비(DIP). 0(기본값)이면 위 표의 확대/축소 너비입니다.
Origin	축의 X = 0 위치의 시간.
Snap	이동이나 크기 변경의 단계: psTick(기본값, pzDays까지는 작은 눈금, 더 거친 확대/축소에서는 하루 단위), psNone, ps5Min, ps15Min, psHour 또는 psDay.
ShowNonWorking	근무 달력의 밤, 휴무일, 휴일에 음영을 입힙니다. 기본값 True.
MajorFormat, MinorFormat	두 단의 레이블. 비어 있으면 위 표의 형식을 사용합니다.

형식 토큰은 대소문자를 구분하지 않습니다: 연도는 `yyyy`와 `yy`, 분기는 `q`, 월은 `m`에서 `mmmm`, 일은 `d`에서 `dddd`, ISO 8601 주는 `w`와 `ww`, 0에서 23까지의 시는 `h`와 `hh`, 분은 `n`과 `nn`, 로캘의 시계로 표시한 시간은 `t`입니다. 작은따옴표나 큰따옴표 안의 텍스트는 그대로 복사됩니다. 이름은 sgcPlannerLocale에서 옵니다.

확대/축소

- Gantt의 ZoomIn과 ZoomOut은 한 수준 더 세밀하게 또는 더 거칠게 이동하며 범위의 끝에서 False를 반환합니다. ZoomToFit은 차트 너비에 프로젝트 전체가 보이는 확대/축소를 고릅니다.
- Ctrl과 마우스 휠은 Gantt와 Timeline을 마우스 위치에서 확대/축소하므로 커서 아래의 시간이 제자리에 유지됩니다. Scheduler의 일, 주중, 주 뷰에서는 대신 슬롯 높이를 바꿉니다.
- ScrollToTime(aTime)은 축을 한 시간으로 스크롤합니다.

리소스 Timeline에서 기간은 확대/축소를 따릅니다: pzHours까지는 하루, pzDays에서는 4주, pzWeeks에서는 3개월, pzMonths에서는 1년, pzQuarters에서는 3년, pzYears에서는 10년.

예제

관련 항목

- TsgcVCLGantt
- TsgcVCLScheduler
- 근무 달력
- 끌어서 놓기

끌어서 놓기

모든 뷰는 마우스, 펜 또는 손가락으로 데이터를 편집합니다. 끌기는 포인터가 4 DIP 이동한 뒤에 시작되고, 진행되는 동안 미리 보기를 보여 주며, 가장자리에 닿으면 뷰를 스크롤하고, 버튼을 놓으면 편집을 커밋합니다. Escape 또는 두 번째 버튼으로 취소합니다.

뷰별 끌기

View	끌기
Scheduler	빈 슬롯 위로 끌어 이벤트를 만듭니다(OnItemCreate가 제목을 주거나 거부합니다). 이벤트를 다른 시간, 다른 날 또는 다른 리소스 열로 이동하고, 위쪽이나 아래쪽 가장자리에서 크기를 바꿉니다. 이동과 크기 변경은 Options.SlotMinutes에 맞춰집니다. 종일 영역과 월 보기에서는 하루 전체가 만들어지고 이동됩니다. Timeline에서는 막대가 다른 시간이나 리소스 행으로 이동하고 눈금에 맞춰 크기가 바뀝니다.
Gantt	막대를 이동하고, 시작이나 종료의 크기를 바꾸고, 막대의 아래쪽 절반에서 진행률을 끌고, 막대 끝 바로 바깥에서 다른 막대로 링크를 그리고(Registered), 작업 트리와 차트 사이의 분할선을 끕니다. 이동하면 작업의 근무 기간이 유지됩니다.
Kanban	카드를 다른 위치, 다른 열 또는 다른 레인으로 이동하거나 열의 카드 순서를 바꿉니다.

허용과 거부

설정	효과
ReadOnly	컨트롤의 모든 끌기와 Delete 키를 끕니다.
DragOptions	컨트롤이 허용하는 끌기: pdoCreate, pdoMove, pdoResize, pdoProgress, pdoLink, pdoReorder. 기본값은 모두입니다. 보드에서 pdoMove 없이 pdoReorder만 있으면 모든 카드가 자기 열에 머뭅니다.
OnItemChanging(데이터)	놓을 때마다 한 번, 어떤 뷰에서든, 변경될 항목과 필드와 함께 발생합니다. aAccept를 False로 설정하면 편집이 거부됩니다. 모든 뷰에서 지켜져야 하는 규칙을 둘 단 하나의 위치입니다.
OnCardMoving(Kanban)	OnItemChanging 전에 대상 상태와 레인과 함께 발생합니다.
OnItemCreate(Scheduler)	새 이벤트에 대해 시작, 종료, 리소스와 함께 발생합니다.
OnLinkCreating(Gantt)	새 링크에 대해 발생합니다. 종류를 바꾸거나 링크를 거부할 수 있습니다.
OnRecurrenceEdit(Scheduler)	시리즈의 발생 항목 위에 놓을 때 발생합니다. 반복을 참조하십시오.
WIPMode pwmBlock(Kanban)	카드가 WIP 제한에 도달한 열로 들어가지 못하게 합니다. 스윔레인과 WIP 제한을 참조하십시오.

놓기가 하는 일

놓기는 데이터에 대한 BeginUpdate와 EndUpdate 한 쌍이므로 모든 뷰는 한 번만 다시 그려집니다. Gantt에서 이동하거나 시작 크기를 바꾸면 작업이 Manual이 아닌 한 작업의 제약도 새 날짜의 pcSNET으로 설정되어 이후 일정 계산이 그 자리에 유지하며, AutoSchedule이면 놓은 뒤에 엔진이 실행됩니다. 항목에 부모가 있으면 그 요약이 업데이트됩니다.

키보드와 터치

Delete 키는 선택된 항목에 대해 OnItemDelete를 발생시키고 aAccept가 True로 남으면 삭제합니다. 보드에서는 SelectNextCard가 화살표 키, Home, End로 선택을 이동합니다. 터치 입력은 마우스 입력으로 전달되므로 손가락은 마우스처럼 끕니다.

예제

관련 항목

- 동기화된 뷰
- TsgcVCLGantt
- TsgcVCLKanban
- TsgcVCLScheduler

Dependencies

종속성은 두 항목을 연결합니다: 후행 작업(ToId)은 선행 작업(FromId)을 기다립니다. 링크는 모든 에디션에서 TsgcPlannerData의 Dependencies에 보관되고 JSON과 MS Project XML로 저장됩니다. Registered 에디션에서 Gantt는 이를 화살표로 그리고, 사용자가 마우스로 그리도록 하며, 작업의 일정을 계산하는 데 사용합니다.

종류와 지연

종류	의미
pdkFS(기본값)	종료-시작: 선행 작업이 끝나면 후행 작업이 시작됩니다.
pdkSS	시작-시작: 선행 작업이 시작되면 후행 작업이 시작됩니다.
pdkFF	종료-종료: 선행 작업이 끝나면 후행 작업이 끝납니다.
pdkSF	시작-종료: 선행 작업이 시작되면 후행 작업이 끝납니다.

Lag는 근무 달력의 작업 분 단위이며 경계에 더해집니다. 음수 Lag는 선행(리드)이므로 후행 작업이 선행 작업과 겹칩니다. AddDependency(FromId, ToId, Kind, Lag)는 한 번의 호출로 링크를 추가합니다.

Gantt에서

- ShowDependencies는 보이는 막대 사이에 각 링크를 꺾인 화살표로 그립니다. 크리티컬 패스의 링크는 Critical 색을 사용합니다.
- 막대의 시작이나 끝 바로 바깥에서 끌어 다른 막대에 놓으면 링크가 만들어집니다. 고른 끝이 종류를 결정합니다. OnLinkCreating은 종류를 바꾸거나 링크를 거부할 수 있습니다. 항목 자신으로의 링크, 항목과 그 조상 또는 자손 사이의 링크, 이미 있는 링크는 거부됩니다.
- 요약 작업에 있는 링크는 계획의 일정을 계산할 때 그 아래의 모든 잎 작업에 적용됩니다.

링크의 일관성 유지

DeleteItem은 삭제된 항목의 링크를 제거합니다. Id를 바꿔도 그것을 사용하는 링크의 이름은 바뀌지 않으므로, 같은 업데이트 안에서 그 링크의 FromId와 ToId를 변경하십시오. 알 수 없는 FromId 또는 ToId를 가진 링크는 유지되지만 엔진이 무시하며 그려지지 않습니다. 링크의 순환은 엔진이 보고합니다. 자동 일정 계산을 참조하십시오.
DataSet 소스는 자체 테이블(DependencyDataSet)이나 작업 테이블의 선행 작업 쉼표 구분 목록(DependsOnField)에서 링크를 읽습니다. DataSet 바인딩을 참조하십시오.

예제

관련 항목

- 자동 일정 계산
- 크리티컬 패스
- TsgcVCLGantt
- MS Project XML

자동 일정 계산

에디션: Registered. Community 에디션에서는 Gantt의 Schedule과 SetBaseline이 EsgcPlannerError를 발생시키고, AutoSchedule은 효과가 없으며, 엔진 유닛은 포함되지 않습니다. 에디션별 기능을 참조하십시오.
일정 계산 엔진인 sgcPlanner_AutoSchedule의 TsgcPlannerScheduleEngine은 작업을 근무 달력에서 선행 작업 뒤에 가능한 한 빨리 배치하고, 모든 작업의 여유 시간을 계산하고, 크리티컬 패스를 표시합니다. Gantt는 Schedule로, 또는 AutoSchedule이 True일 때 편집할 때마다 이를 실행하며, 코드에서는 뷰 없이 TsgcPlannerData에 대해 실행할 수도 있습니다(예: 서비스에서).

규칙

- 잎 작업은 근무 기간, 즉 Start와 Finish 사이의 근무 분을 유지하며, 프로젝트 시작, 제약 날짜, 종속성 경계 중 가장 늦은 것 이후의 첫 근무 분에 시작합니다.
- 프로젝트 시작은 데이터의 ProjectStart이며, ProjectStart가 설정되지 않았으면 잎 작업의 가장 이른 Start입니다.
- ConstraintDate가 있는 pcSNET 제약은 그 날짜보다 일찍 시작하지 않음을 뜻합니다. 기본값인 pcASAP에는 날짜가 없습니다. Gantt에서 막대를 이동하면 새 날짜에 pcSNET이 설정되므로 다음 일정 계산은 작업을 놓은 자리에 유지합니다.
- Manual 작업, 반복 항목, 종속성 순환에 속한 모든 작업은 날짜를 유지하며 후행 작업을 계속 결정합니다.
- 근무 분이 없는 작업은 마일스톤이며 경계에 정확히 위치합니다.
- 요약 작업은 잎의 가장 이른 시작과 가장 늦은 종료를 취합니다. 요약에 있는 링크는 그 잎 각각에 적용됩니다.

실행하기

멤버	설명
Gantt.Schedule	Gantt의 데이터에 엔진을 실행합니다. 순환이 발견되면 False를 반환합니다.
Gantt.AutoSchedule	Gantt에서 편집할 때마다(이동, 크기 변경, 새 링크) 엔진을 실행합니다. 코드나 다른 뷰에서 한 변경은 엔진을 실행하지 않으므로 그 뒤에 Schedule을 호출하십시오.
Gantt.OnScheduleError, Engine.OnScheduleError	링크가 순환을 이룰 때 업데이트 후에 발생하며, 메시지와 순환에 속한 작업의 Id를 전달합니다. 나머지 작업은 그대로 일정에 반영됩니다.
Engine.Schedule(aData), Engine.CycleIds	뷰 없이 같은 처리를 하고, 마지막으로 발견된 순환의 Id.

모든 날짜와 결과는 한 번의 업데이트로 기록됩니다: 날짜는 pckItems, 여유 시간과 크리티컬 플래그는 pckSchedule. 아무것도 바꾸지 않는 처리는 아무것도 통지하지 않습니다. 개발 머신에서 링크 15,000개와 휴일 30개를 가진 작업 10,000개는 약 11 ms(Win64)에 일정이 계산됩니다.

예제

관련 항목

- 종속성
- 크리티컬 패스
- 기준선
- 근무 달력

크리티컬 패스

에디션: Registered. Community 에디션에서는 ShowCriticalPath가 효과가 없고 여유 시간을 계산하는 엔진이 포함되지 않습니다. 에디션별 기능을 참조하십시오.
크리티컬 패스는 프로젝트의 종료를 결정하는 작업의 사슬입니다: 그중 하나가 지연되면 종료가 지연됩니다. 일정 계산 엔진은 처리할 때마다 이를 찾습니다.

여유 시간

- 정방향 처리 뒤에, 모든 잎 작업의 가장 늦은 종료인 프로젝트 종료에서 역방향 처리를 하면 **TotalSlack**이 나옵니다: 프로젝트 종료를 움직이지 않고 작업이 지연될 수 있는 근무 분입니다. 시작 여유와 종료 여유 중 작은 값입니다.
- **Critical**은 TotalSlack이 0 이하이면 True입니다. 날짜를 유지하는 작업(Manual 또는 반복)이 종속성을 깨뜨리면 여유 시간이 음수가 됩니다.
- 요약 작업은 잎의 가장 작은 여유 시간을 취하며, 그중 하나가 크리티컬이면 크리티컬입니다. 순환에 속한 작업은 여유 시간이 없으며 크리티컬이 되지 않습니다.

TotalSlack과 Critical은 읽기 전용 결과입니다. JSON에 저장되지 않으므로 프로젝트를 로드한 뒤 Schedule을 실행하십시오.

표시하기

Gantt의 ShowCriticalPath를 True로 설정합니다: 크리티컬 작업의 막대와 링크가 테마의 Critical 색으로 그려지며, Colors.Critical로 재정의할 수 있습니다. 결과가 최신이 되도록 Schedule을 실행하거나 AutoSchedule을 설정하십시오.

예제

관련 항목

- 자동 일정 계산
- 종속성
- 기준선
- TsgcVCLGantt

기준선

에디션: Registered. Community 에디션에서는 BaselineStart와 BaselineFinish가 저장되고 보관되며, ShowBaseline은 효과가 없고, Gantt의 SetBaseline은 EsgcPlannerError를 발생시킵니다. 에디션별 기능을 참조하십시오.
기준선은 계획의 저장된 복사본입니다. 각 항목에는 Start와 Finish 옆에 BaselineStart와 BaselineFinish가 있으며, Gantt는 기준선을 현재 막대 아래에 얇은 막대로 그리므로 지연된 작업을 한눈에 볼 수 있습니다.

기준선 설정

- Gantt의 SetBaseline은 모든 항목의 Start와 Finish를 기준선 날짜로 한 번의 업데이트로 복사합니다.
- 뷰 없이는 TsgcPlannerScheduleEngine의 SetBaseline과 ClearBaseline이 TsgcPlannerData에 대해 같은 일을 합니다. ClearBaseline은 기준선 날짜를 0, 즉 설정 안 됨으로 되돌립니다.
- 기준선 날짜는 일반 속성이므로 코드에서 작업 하나의 기준선을 설정하거나, 자체 기준선을 여러 개 보관하고 그중 하나를 복사해 넣을 수 있습니다.

표시하기

Gantt의 ShowBaseline을 True로 설정합니다. 기준선이 있는 항목의 행은 현재 막대를 행의 위쪽에, 기준선을 그 아래에 그립니다. 코드에서 달력의 WorkingMinutesBetween으로 날짜를 비교하면 지연을 근무 시간 단위로 보고할 수 있습니다.

파일

JSON은 기준선 날짜를 저장합니다. MS Project XML은 기준선 번호 0을 읽고 쓰며, iCalendar는 이를 X-SGC-BASELINE-START와 X-SGC-BASELINE-FINISH에 보관합니다.

예제

관련 항목

- 자동 일정 계산
- 크리티컬 패스
- TsgcVCLGantt
- MS Project XML

스윔레인과 WIP 제한

에디션: Registered. Community 에디션에서는 SwimlaneBy가 pslNone처럼 동작하고 WIPLimit는 배지를 표시하지 않으며 카드를 막지 않습니다. 열, 너비, 접기는 모든 에디션에서 동작합니다. 에디션별 기능을 참조하십시오.
Kanban 보드는 항목을 열 안의 카드로 보여 줍니다. 스윔레인은 보드를 가로 띠로 나누고, WIP 제한은 열의 진행 중인 작업을 제한합니다.

Columns

속성	설명
ColumnSource	pkcStatuses(기본값): 데이터의 상태마다 열 하나, 상태의 순서대로. pkcList: Columns의 열만.
Columns	StatusId, Caption, Color, Width, WIPLimit, Collapsed를 가진 항목. pkcStatuses에서는 상태의 StatusId를 가진 항목이 그 열에 설정을 추가합니다. AddColumn(StatusId, Caption, WIPLimit)으로 하나를 추가합니다.
ShowUnknownColumn	상태가 어떤 열과도 일치하지 않는 카드를 마지막 열에 보여 줍니다. 카드를 여기에 놓을 수는 없습니다.

요약 항목과 수정된 발생 항목은 카드로 표시되지 않습니다. 열의 카드는 Rank 순으로, 그다음 데이터에서의 순서로 정렬됩니다. CollapseColumn과 ExpandColumn, 또는 열 머리글의 화살표는 열을 좁은 레일로 접으며, OnColumnCollapse가 이를 알립니다.

스윔레인

SwimlaneBy	레인	다른 레인에 놓기
pslNone (default)	레인 없음.	불가능.
pslResource	리소스마다 레인 하나. 카드는 첫 번째 리소스의 레인으로 갑니다.	카드의 첫 번째 리소스를 대체합니다.
pslParent	카드의 부모마다 레인 하나, 데이터의 순서대로.	카드의 ParentId를 설정합니다.
pslTag	카드의 첫 번째 태그마다 레인 하나, 정렬됨.	불가능.
pslEvent	OnGetSwimlane이 키와 캡션으로 반환하는 레인, 나타나는 순서대로. 핸들러가 없으면 pslNone처럼 동작합니다.	불가능.

레인이 없는 카드는 끝의 레인으로 갑니다.

WIP 제한

- 열 항목의 WIPLimit은 모든 레인을 합산하여 열이 담아야 하는 최대 카드 수입니다. 0은 제한 없음을 뜻합니다.
- 머리글 배지는 개수를 표시하고, 제한 미만이면 개수와 제한을, 제한에 도달하면 주황색 배지를, 초과하면 빨간색 배지를 표시합니다.
- WIPMode pwmWarn(기본값)은 열에 표시만 합니다. pwmBlock은 제한에 도달한 열로 들어오려는 카드를 거부하며, 이미 열에 있는 카드의 순서는 계속 바꿀 수 있습니다.

예제

관련 항목

- TsgcVCLKanban
- 끌어서 놓기
- 테마
- TsgcFMXKanban

DataSet 바인딩

TsgcPlannerSource_DataSet는 TsgcPlannerData를 TDataSet에 바인딩합니다. FireDAC, TClientDataSet, dbExpress, 서드파티 데이터셋 등 DB 유닛의 모든 데이터셋이 가능합니다. 항목, 리소스, 종속성을 로드하고, 데이터셋의 변경을 따라가며, 뷰에서 한 편집을 되써 줍니다. DB 유닛만 사용하므로 모든 에디션에서 VCL, FireMonkey, 콘솔 애플리케이션에서 동작합니다.

데이터셋과 필드 매핑

데이터셋	필드 매핑	기본 필드 이름
DataSet	ItemFields	Id, Title, Start, Finish, AllDay, Status, Progress, Color, ParentId, Tags, Resources, Rank, Notes, RecurrenceRule, DependsOn.
ResourceDataSet	ResourceFields	Id, Name, Color, Initials.
DependencyDataSet	DependencyFields	FromId, ToId, Kind, Lag.

비어 있거나 데이터셋에서 찾을 수 없는 필드 이름은 무시되므로 테이블은 자신이 가진 필드만 매핑합니다. 날짜는 날짜 및 시간 필드, 숫자 필드 또는 ISO 8601 텍스트에서, 불리언은 불리언 필드, 숫자 또는 Y나 True 같은 텍스트에서, 색은 ARGB 숫자나 #RRGGBB 같은 텍스트에서 읽습니다. null인 Finish는 Finish = Start가 되고, 종일 항목이면 다음 자정이 됩니다. 종속성의 Kind는 FS, SS, FF 또는 SF이며 Lag는 작업 분 단위입니다. DependencyDataSet이 없으면 DependsOn 필드가 작업의 선행 작업 Id를 쉼표로 구분해 담을 수 있으며, 종료-시작 링크로 읽습니다. OnLoadItem은 매핑된 필드가 복사된 뒤 각 레코드마다 발생하여 항목의 다른 속성을 채울 수 있습니다.

로드와 실시간 업데이트

Active를 True로 하면 열려 있는 모든 데이터셋을 한 번의 업데이트로 로드하므로 바인딩된 뷰는 한 번만 다시 그려집니다. 10,000행은 Win64에서 약 26 ms에 로드됩니다. LiveUpdates를 사용하면 게시된 편집이나 삽입이 해당 항목 하나를 업데이트하거나 추가하며, 레코드 수의 변경(삭제, 필터, 새로 고침)은 Id로 항목을 동기화합니다. 스크롤과 게시되지 않은 편집은 아무것도 바꾸지 않습니다. Load와 Refresh는 수동으로 다시 로드합니다.

되쓰기

기본적으로 True인 WriteBack을 사용하면 뷰나 코드에서 한 데이터 편집이 Id 필드에서 Locate로 찾은 레코드에 게시됩니다: 변경된 매핑 필드만 할당됩니다. 새 항목은 레코드를 추가하고, 삭제된 항목은 해당 레코드를 삭제하며, 새로 만들거나 제거된 링크는 DependencyDataSet 또는 DependsOn 목록에 기록됩니다. 데이터셋이 편집 또는 삽입 상태인 동안에는 아무것도 기록되지 않으며, 보류 중인 변경은 찾아보기 상태로 돌아오면 기록됩니다. 리소스, 계산 필드, 조회 필드, 읽기 전용 필드는 기록되지 않습니다. 기록이 새 로드로 되돌아오는 일은 없습니다.

예제

DataSetPlanner 데모는 자체 필드 이름을 가진 TClientDataSet 테이블 세 개를 바인딩합니다. 데모를 참조하십시오.

관련 항목

- TsgcPlannerSource_DataSet
- 공유 데이터
- JSON 저장
- 종속성

JSON 저장

TsgcPlannerData는 모든 에디션과 모든 플랫폼에서 JSON으로 저장하고 불러옵니다: 문자열에는 SaveToJSON과 LoadFromJSON, 파일에는 SaveToFile과 LoadFromFile, 스트림(sgcPlannerJSONSaveToStream과 sgcPlannerJSONLoadFromStream)과 파일에는 sgcPlanner_JSON의 함수를 사용합니다.

형식

문서에는 `"format": "sgcPlanner"`와 버전이 있고, 이어서 프로젝트 시작, 달력, 리소스, 상태, 항목, 종속성이 있습니다. 날짜는 ISO 8601 로컬 시간이고 색은 ARGB 숫자입니다. 줄인 프로젝트:

```
{
  "format": "sgcPlanner",
  "version": 1,
  "projectStart": "2026-10-05T08:00:00",
  "calendar": {"workDays":["mon","tue","wed","thu","fri"],"workTime":"08:00-12:00,13:00-17:00","firstDayOfWeek":"mon","holi
  "resources": [
    {"id":"ann","name":"Ann Lee","color":4280973526}
  ],
  "statuses": [
    {"id":"todo","caption":"To do"},
    {"id":"done","caption":"Done","color":4282557941}
  ],
  "items": [
    {"id":"t1","title":"Design","start":"2026-10-05T08:00:00","finish":"2026-10-07T17:00:00","status":"done","progress":100
    {"id":"m1","title":"Weekly review","start":"2026-10-05T09:00:00","finish":"2026-10-05T09:30:00","color":4285887861,"not
  ],
  "dependencies": [
    {"from":"t1","to":"t2","kind":"FS"}
  ]
}
```

규칙

- 파일과 스트림은 바이트 순서 표시가 없는 UTF-8입니다. 불러올 때는 바이트 순서 표시도 허용됩니다.
- 기본값을 가진 항목 멤버(빈 텍스트, false, 0 또는 설정되지 않은 날짜)는 기록되지 않으며, id, title, start, finish는 예외로 항상 기록됩니다. 이렇게 하면 파일이 작게 유지됩니다.
- 계산된 값(TotalSlack, Critical, IsSummary), 선택, Version은 저장되지 않습니다.
- 불러오기는 달력과 ProjectStart를 포함한 전체 데이터를 한 번의 업데이트로 교체하므로 바인딩된 모든 뷰가 한 번만 다시 그려집니다. 없는 멤버는 기본값으로 로드됩니다.
- 알 수 없는 멤버, 다른 JSON 형식의 멤버, 구문 분석되지 않는 값은 무시되므로 이후 릴리스가 쓴 파일도 계속 로드됩니다. 버전이 없으면 1로 읽습니다.
- 텍스트가 올바른 JSON이 아니거나, 루트가 객체가 아니거나, 형식이 다르거나, 버전이 지원되지 않으면 EsgcPlannerError가 발생하고 데이터는 그대로 유지됩니다.
- Delphi 2009 이전에는 텍스트가 시스템의 ANSI 코드 페이지를 거칩니다.

개발 머신에서 링크 15,000개를 가진 항목 10,000개는 약 48 ms에 저장되고 약 56 ms에 로드됩니다(Win64).

예제

관련 항목

- TsgcPlannerData
- DataSet 바인딩
- iCalendar
- MS Project XML

iCalendar

에디션: Registered. Community 에디션에는 TsgcPlannerICalendar가 없으며, 그 에디션의 파일 형식은 JSON입니다. 에디션별 기능을 참조하십시오.
TsgcPlannerICalendar는 Google Calendar, Outlook, Apple Calendar 및 대부분의 캘린더 서버가 쓰는 형식인 RFC 5545 달력(.ics)을 가져오고 내보냅니다. VEVENT 하나가 항목 하나입니다.

가져오기

- ImportFromString, ImportFromStream, ImportFromFile은 바이트 순서 표시(BOM) 유무와 관계없이 UTF-8을 읽습니다. 잘못된 UTF-8은 경고와 함께 Latin-1로 읽습니다.
- ImportMode pimReplace(기본값)는 먼저 항목과 종속성을 삭제하고 리소스, 상태, 달력, ProjectStart는 유지합니다. pimMerge는 같은 Id의 항목을 업데이트하고 나머지를 추가합니다.
- UTC 시간과 TZID가 있는 시간은 파일의 VTIMEZONE을 사용해 컴퓨터의 로컬 시간으로 변환됩니다. UTC, GMT, Etc/UTC에는 VTIMEZONE이 필요 없습니다. 알 수 없는 TZID는 경고 하나를 내고 로컬 시간으로 읽습니다.
- 항목의 Id는 UID입니다. RECURRENCE-ID가 있는 VEVENT는 해당 UID를 가진 시리즈의 수정된 발생 항목이 됩니다.
- 지원되지 않는 RRULE 부분이나 RDATE처럼 가져오기를 멈추지 않는 문제는 줄 번호와 함께 OnImportWarning으로 보고되고 WarningCount에 집계됩니다. BEGIN:VCALENDAR가 없는 텍스트는 EsgcPlannerError를 발생시키고 데이터를 변경하지 않습니다.

내보내기

- ExportToString, ExportToStream, ExportToFile은 모든 항목을 VEVENT로 기록하며, ProductId를 PRODID로 사용합니다.
- ExportUTC(기본값 True)를 사용하면 반복되지 않는 항목의 시간이 UTC로 기록됩니다. 시리즈와 수정된 발생 항목은 항상 부동 로컬 시간을 사용하므로 발생 항목이 일광 절약 시간 변경 후에도 같은 시각에 유지됩니다.
- 시작과 종료가 자정인 종일 항목은 DATE 값으로 기록됩니다.
- 종속성은 내보내지 않습니다.

매핑

iCalendar	항목
UID, SUMMARY, DESCRIPTION	Id, Title, Notes.
DTSTART, DTEND	Start, Finish.
RRULE, EXDATE, RECURRENCE-ID	RecurrenceRule, RecurrenceExceptions, RecurrenceParentId, RecurrenceDate.
STATUS	소문자의 Status: tentative, confirmed 또는 cancelled. 다른 상태는 X-SGC-STATUS에 보관됩니다.
RESOURCES	리소스의 이름. 다르면 X-SGC-RESOURCES가 Id를 보관하며, 가져오기는 없는 리소스를 생성합니다.
CATEGORIES	Tags.
RELATED-TO;RELTYPE=PARENT	ParentId.
X-SGC-PROGRESS, X-SGC-COLOR, X-SGC-RANK, X-SGC-MILESTONE, X-SGC-MANUAL, X-SGC-CONSTRAINT, X-SGC-CONSTRAINT-	iCalendar에 대응하는 필드가 없는 Planner 속성이므로, Planner 파일이 왕복 변환됩니다.

DATE, X-SGC-BASELINE-START, X-SGC-BASELINE-FINISH, X-SGC-ALLDAY	

개발 머신에서 시간대가 있는 VEVENT 5,000개를 약 84 ms(Win64)에 가져옵니다. ICalendarSync 데모는 Google 및 Outlook 달력을 가져오고 시리즈를 내보냅니다. 데모를 참조하십시오.

관련 항목

- TsgcPlannerICalendar
- 반복
- MS Project XML
- JSON 저장

MS Project XML

에디션: Registered. Community 에디션에는 TsgcPlannerMSProject가 없으며, 그 에디션의 파일 형식은 JSON입니다. 에디션별 기능을 참조하십시오.
TsgcPlannerMSProject는 Microsoft Project가 열고 저장하는 XML 형식인 Microsoft Project XML(MSPDI)을 가져오고 내보냅니다. 작업, 링크, 달력, 리소스, 할당, 기준선을 읽습니다.

가져오기

- ImportFromString, ImportFromStream, ImportFromFile은 바이트 순서 표시 유무와 관계없이 UTF-8을 읽습니다. 형식이 올바르지 않은 문서는 줄 번호와 함께 EsgcPlannerError를 발생시킵니다.
- ImportMode pmiReplace(기본값)는 항목과 링크를 삭제하고 달력과 프로젝트 시작(StartDate)을 파일에서 가져옵니다. 리소스와 상태는 유지되며 파일의 리소스가 추가되거나 이름이 바뀝니다.
- pmiMerge는 달력과 ProjectStart를 유지하고, Id로 항목을 업데이트하며, 두 끝으로 링크를 추가하거나 업데이트합니다. 항목의 리소스는 파일에 Assignments가 있을 때만 파일을 따릅니다.
- OnImportWarning은 다른 방식으로 읽은 내용을 보고합니다. 예를 들어 sgcPlanner에 없는 제약 유형입니다.

매핑

MSPDI	sgcPlanner
Task UID, Name, Start, Finish, PercentComplete, Milestone, Notes	Id, Title, Start, Finish, Progress, Milestone, Notes.
OutlineLevel과 작업 순서	ParentId, 작업 트리.
PredecessorLink Type 0, 1, 2, 3	pdkFF, pdkFS, pdkSF, pdkSS. LinkLag는 작업 분의 10분의 1 단위이며, 경과 시간 및 백분율 지연 형식은 변환됩니다.
ConstraintType 0과 4	pcASAP와 ConstraintDate가 있는 pcSNET. 유형 2(지정일에 시작)는 pcSNET으로, 나머지 유형은 pcASAP로 읽으며 둘 다 경고가 표시됩니다.
Baseline Number 0	BaselineStart와 BaselineFinish.
Calendar WeekDays, WorkingTimes, Exceptions	근무 달력: 근무일, 근무 시간, 휴일.
Resource Code 또는 Name, Assignments	리소스와 각 항목의 Resources 목록.

내보내기

ExportToString, ExportToStream, ExportToFile은 SaveVersion 14의 스키마 순서로 기록합니다: Standard 달력 하나(휴일마다 예외 하나), 개요 순서의 작업, 그 링크, 그 할당. 숫자 Id는 자신의 UID를 유지하고 다른 Id는 다음 빈 번호를 받습니다. MSPDI는 작업 쌍마다 링크 하나를 유지합니다. 상태, 태그, 색, 순위, 종일, 반복은 기록되지 않으므로 필요하면 JSON 파일을 함께 보관하십시오.
개발 머신에서 링크 15,000개를 가진 작업 10,000개는 약 264 ms(Win64)에 가져옵니다. ProjectGantt 데모는 MSPDI 파일을 열고 저장합니다. 데모를 참조하십시오.

관련 항목

- TsgcPlannerMSProject
- 종속성
- 기준선

- JSON 저장

헤드리스 렌더링

모든 sgcPlanner 뷰는 창 없이 그릴 수 있습니다. 뷰 모델은 sgcUICore의 TsgcUICanvas에 그려지며, 폼의 컨트롤, TsgcPlannerRenderer, 컨트롤의 SaveToPNG와 SaveToSVG 메서드가 같은 페인터를 사용하므로 이미지는 화면과 같은 픽셀입니다.

TsgcPlannerRenderer

렌더러는 VCL이나 FireMonkey 유닛이 없는 비주얼이 아닌 컴포넌트이므로 서비스, 웹 서버, 콘솔 애플리케이션, Linux64에서 실행됩니다. 모든 에디션에서 동작합니다.

멤버	설명
Data, View	데이터와 그릴 뷰: prvScheduler, prvGantt 또는 prvKanban.
Width, Height, Scale	DIP 단위의 크기(기본값 800 x 600)와 출력의 배율, HiDPI 이미지는 2.
Theme, Colors	컨트롤과 같은 테마 및 색 재정의. 테마를 참조하십시오.
SchedulerOptions, GanttOptions, KanbanOptions	각 뷰의 게시된 옵션. Assign은 뷰 모델도 받으므로 `Renderer.GanttOptions.Assign(VCLGantt1.GanttView)` 는 폼이 보여 주는 것을 복사합니다.
Date	Scheduler가 보여 주는 날짜, 0이면 오늘.
SchedulerView, GanttView, KanbanView	렌더러가 소유한 뷰 모델. 렌더링 전에 ExpandAll, ZoomToFit, ScrollToTime 또는 CollapseColumn을 호출할 때 사용합니다.
RenderTo(aCanvas), SaveToPNG, SaveToSVG	캔버스에 그리거나, PNG 또는 SVG를 파일이나 스트림으로 저장합니다.
FontFile, FontFileBold	텍스트용 TrueType 파일. 비어 있으면 sgcUICore에 내장된 글꼴을 사용하므로 컴퓨터에 설치된 글꼴이 필요 없습니다.

컨트롤의 이미지

여섯 컨트롤 모두 SaveToPNG와 SaveToSVG를 가지며, 파일 이름 또는 스트림과 선택적 너비, 높이, 배율을 받습니다. 크기는 DIP 단위이며 0은 클라이언트 크기이고, 배율 0은 컨트롤의 배율입니다. 출력은 컨트롤의 스크롤 위치를 유지하며, 같은 데이터, 옵션, 크기, 배율을 가진 렌더러와 같은 이미지입니다. 스크롤 막대, 끌기 미리 보기, 놓기 표시는 제외됩니다.

서버에서

- 이미지의 선택은 데이터의 SelectedId입니다. 선택된 항목이 없는 보고서를 만들려면 이를 지우십시오.
- 요일과 월 이름, 날짜 형식은 컴퓨터에서 초기화되는 sgcPlannerLocale에서 옵니다. 서버가 데스크톱과 같은 레이블을 만들어야 한다면 시작할 때 설정하십시오.
- SVG 출력은 텍스트를 텍스트로 유지하므로 어떤 확대/축소에서도 선명하며 검색할 수 있습니다.

예제

HeadlessReport 데모는 각 뷰를 PNG와 SVG로 기록하는 Windows 및 Linux64용 콘솔 애플리케이션입니다. 데모를 참조하십시오.

관련 항목

- TsgcPlannerRenderer
- 테마
- 플랫폼
- 동기화된 뷰

테마

Planner 컨트롤과 렌더러는 sgcUICore의 테마로 그려지므로, Planner는 같은 폼의 다른 sgcUI 제품, VCL, FireMonkey, 헤드리스 출력과 어울립니다.

테마와 색

- **Theme**은 utLight(기본값), utDark 또는 utCustom입니다. utCustom은 밝은 색에서 시작합니다.
- **Colors**는 테마의 개별 색을 재정의합니다: Background, Surface, Border, Grid, HeaderBackground, HeaderText, Text, TextMuted, Accent, Selection, NonWorking, NowLine, Critical, Ghost. 색은 ARGB TsgcUIColor이며, 예를 들어 sgcUIARGB($FF, $00, $96, $88)입니다. 0이면 테마의 색을 유지합니다.

항목 색

항목은 설정된 첫 번째 색으로 그려집니다: 항목의 Color, 상태의 Color, 첫 번째 리소스의 Color, 그리고 없으면 Id로 고른 테마 팔레트의 색이므로, 같은 항목은 모든 뷰와 모든 실행에서 같은 색을 유지합니다.

설정	효과
Gantt BarColorMode	pbcItem(위의 항목 색), pbcStatus(상태 색) 또는 pbcResource(첫 번째 리소스의 색).
Kanban Columns Color	열 머리글의 색 막대. pkcStatuses에서는 설정되지 않으면 상태 색이 사용됩니다.
Tags	카드의 태그 알약은 태그 텍스트로 고른 테마 팔레트의 색을 사용합니다.
Resources	카드와 레인의 이니셜 원은 리소스 색을 사용합니다.

글꼴과 크기

텍스트는 sgcUICore에 내장된 글꼴을 사용하므로 출력은 모든 컴퓨터에서 같습니다. 렌더러는 FontFile과 FontFileBold로 TrueType 파일을 받습니다. 크기는 DIP 단위이며 모니터, 폼 또는 렌더러의 Scale에 맞춰 조정됩니다.

예제

관련 항목

- 헤드리스 렌더링
- 스윔레인과 WIP 제한
- TsgcVCLGantt
- TsgcVCLKanban

에디션별 기능

sgcPlanner는 Registered, Trial, Community로 제공됩니다. Trial에는 Registered의 모든 기능이 있습니다. 이 표는 각 기능과 그에 필요한 에디션을 나열합니다. 에디션 자체는 에디션을 참조하십시오.

기능	Registered 및 Trial	Community
공유 데이터, 동기화된 뷰	예	예
근무 달력	예	예
반복	예	아니요. 반복 항목은 한 번만 표시됩니다.
시간 눈금과 확대/축소	예	예
끌어서 놓기	예	예. 단, Gantt에서 끌어서 링크 만들기는 제외입니다.
종속성	예	링크는 저장되고 보관되지만 그려지지 않습니다.
자동 일정 계산	예	아니요. Schedule이 EsgcPlannerError를 발생시킵니다.
크리티컬 패스	예	No.
기준선	예	날짜는 저장되고 보관되지만 그려지지 않습니다. SetBaseline이 EsgcPlannerError를 발생시킵니다.
Scheduler의 일, 주중, 주, 월, 안건 뷰	예	예
Scheduler 리소스 Timeline	예	아니요. 뷰에 안내 메시지가 표시됩니다.
Kanban 열, 접기, 카드 옵션	예	예
스윔레인과 WIP 제한	예	아니요. SwimlaneBy는 pslNone처럼 동작합니다.
DataSet 바인딩	예	예
JSON 저장	예	예
iCalendar, MS Project XML	예	아니요. 컴포넌트가 존재하지 않습니다.
헤드리스 렌더링, 테마	예	예

모든 에디션에서 컴파일되는 코드

고급 기능은 SGC_PLANNER_ADVANCED 아래에서 컴파일되며, sgcVer.inc는 이를 Registered와 Trial 에디션에서 정의하고 Community에서는 끕니다. 두 에디션 모두에서 빌드되어야 하는 유닛에는 sgcVer.inc를 포함하고 고급 기능의 코드를 `{$IFDEF SGC_PLANNER_ADVANCED}` 안에 두십시오. 뷰의 속성은 모든 에디션에서 게시되므로 ShowCriticalPath나 SwimlaneBy를 설정하는 폼은 Community에서 로드되지만, TsgcPlannerICalendar나 TsgcPlannerMSProject를 포함한 폼은 로드되지 않습니다.

예제

관련 항목

- 에디션
- 설치
- 문제 해결

Google 캘린더 동기화

TsgcPlannerGoogleSync는 TsgcPlannerData와 Google 캘린더를 양방향으로 동기화된 상태로 유지합니다. 이 항목은 구성 요소에 필요한 것, 동기화가 실행되는 방식, 필드 매핑, 충돌 규칙, 상태 파일을 설명합니다.

요구 사항

- sgcPlanner의 Registered 에디션과 sgcPlannerWS 브리지 패키지(유닛 sgcPlannerWS_Classes, sgcPlannerWS_Google, sgcPlannerWS_ICal).
- Standard 에디션 이상의 sgcWebSockets. Google 캘린더 클라이언트 TsgcHTTPGoogleCloud_Calendar_Client가 포함되어 있습니다. sgcPlanner, sgcUICore, sgcWebSockets 소스 또는 sgcWebSockets 라이브러리 폴더를 검색 경로에 추가하십시오.
- sgcWebSockets가 HTTPS에 사용하는 OpenSSL 라이브러리를 실행 파일 옆에 둡니다.

Google Cloud 프로젝트와 OAuth2 클라이언트

1. Google Cloud 콘솔을 열어 프로젝트를 만들거나 선택한 다음 Google Calendar API를 사용 설정합니다.
2. OAuth 동의 화면을 구성합니다. 앱이 테스트 중인 동안에는 로그인할 Google 계정을 테스트 사용자로 추가합니다.
3. 유형이 데스크톱 앱인 OAuth 클라이언트 ID를 만들고 클라이언트 ID와 클라이언트 보안 비밀을 복사합니다.
4. 클라이언트를 설정합니다. GoogleCloudOptions.Authentication을 gcaOAuth2로, 클라이언트 ID와 보안 비밀을 GoogleCloudOptions.OAuth2에, 브라우저 리디렉션용 LocalIP와 LocalPort(예: 127.0.0.1과 8080)를, 그리고 Scopes.EventsReadWrite와 Scopes.CalendarsReadWrite를 설정합니다. 동기화는 시간대를 알아내기 위해 캘린더를 한 번 읽습니다.
5. 첫 요청에서 브라우저가 Google 동의 페이지를 엽니다. 사용자가 접근을 허용하면 동기화가 계속됩니다.

클라이언트 보안 비밀을 폼 파일이나 배포하는 소스 코드에 저장하지 마십시오. 데모는 실행 시 이를 입력받아 메모리에만 보관합니다.

동기화가 실행되는 방식

1. 작업자 스레드: CalendarId의 이벤트 목록을 가져옵니다. 마지막 동기화의 동기화 토큰이 있으면 그 이후의 변경만(삭제된 이벤트 포함), 첫 동기화에서는 모든 이벤트를 nextPageToken을 따라 가져옵니다. 시리즈는 각각 하나의 마스터 이벤트로 옵니다.
2. 메인 스레드: 이벤트는 한 번의 BeginUpdate와 EndUpdate 안에서 Data에 적용됩니다. 취소된 이벤트는 해당 항목을 삭제하고, 취소된 발생은 해당 시리즈에 예외를 추가합니다. 충돌이 해결되며, OnFilterItem 이후에 새로 생기거나 변경되거나 삭제된 항목으로 전송 목록이 만들어집니다.
3. 작업자 스레드: Google에서의 삽입, 업데이트, 삭제.
4. 메인 스레드: 새 이벤트 ID, etag, 업데이트 시간, 지문, 새 동기화 토큰이 저장되고, 상태가 저장되며, OnSyncDone이 발생합니다.

Google에서 읽은 항목은 g:에 이벤트 ID를 붙인 ID를 받습니다. Planner에서 만든 항목은 자신의 ID를 유지하며, 이 ID는 이벤트의 비공개 확장 속성(sgcPlannerId)에 담겨 전달되므로 같은 ID로 돌아옵니다.

필드 매핑

Planner 항목	Google 이벤트	비고
Title	summary	
비고	description	
Start, Finish	start.dateTime, end.dateTime	오프셋과 함께 읽혀 로컬 벽시계 시간으로 변환됩니다. UTC로 기록되지만, 반복 이벤트와 그 인스턴스는 예외로 오프셋이 있는 로컬 벽시계 시간과 timeZone(TimeZone 또는 캘린더의 시간

		대)으로 기록됩니다. 시간의 정밀도는 분이며 초는 버려집니다.
AllDay, Start, Finish	start.date, end.date	종료는 양쪽 모두 배타적입니다. 10월 1일의 종일 항목은 10월 1일 00:00부터 10월 2일 00:00까지이며, 해당 이벤트는 10월 2일에 끝납니다. 기록할 때 자정이 아닌 Finish는 다음 날로 올림됩니다.
RecurrenceRule	the RRULE line of recurrence	양방향으로 변경 없이 전달됩니다.
RecurrenceExceptions	the EXDATE lines of recurrence	UTC 시간으로, 종일 시리즈에서는 날짜로 기록되므로 시간대 데이터베이스가 필요하지 않습니다.
RecurrenceParentId, RecurrenceDate	recurringEventId, originalStartTime	시리즈의 수정된 발생.
Id	extendedProperties.private.sgcPlannerId	항목의 Planner ID.

RDATE나 EXRULE처럼 이벤트의 다른 반복 줄은 상태에 보관되고 변경 없이 다시 기록됩니다. sgcPlanner의 반복 엔진이 펼치지 않는 규칙(예: BYSETPOS나 HOURLY가 있는 규칙)도 보관되어 변경 없이 다시 기록됩니다. 항목은 첫 번째 발생을 표시하고 OnError가 규칙마다 한 번 이유를 보고하므로 Google이 규칙을 잃는 일은 없습니다.

동기화되지 않는 것

Status, Progress, Color, Tags, Resources와 항목의 종속성은 Google로 전송되지 않으며, 이벤트의 참석자, 알림, 색상, 회의 데이터는 읽지 않습니다. Planner에서 편집한 항목은 Calendar API의 patch 메서드로 기록됩니다. 이 메서드는 Planner가 소유한 필드만 전송하므로 이벤트의 참석자, 알림, 회의 데이터 및 기타 속성은 유지됩니다.

충돌

마지막 동기화 이후 Planner와 Google 양쪽에서 변경된 항목은 충돌입니다. 구성 요소는 항목이 마지막으로 동기화된 지문과 처음 달라진 시간을 기록하며, 이 시간을 이벤트의 `updated` 시간과 UTC로 비교합니다.

- 나중의 변경이 우선합니다. 동시이거나 읽을 수 있는 업데이트 시간이 없는 이벤트는 Google이 우선합니다.
- OnConflict는 항목과 이벤트의 JSON을 받으며 aResolution을 gcrKeepLocal, gcrKeepRemote 또는 gcrSkip(이번 실행에서는 양쪽을 그대로 둠)으로 설정할 수 있습니다.
- 삭제에 대한 편집은 양쪽에서 편집을 유지합니다. 삭제된 이벤트는 다시 만들어지고 삭제된 항목은 복원됩니다. 충돌로 집계되지만 OnConflict는 발생하지 않습니다.
- 내용이 이미 항목과 일치하는 이벤트는 기록만 되고 충돌로 집계되지 않습니다. 구성 요소 자체의 전송에서 돌아오는 이벤트는 etag로 식별됩니다.
- Direction이 gsdPullOnly이면 로컬 변경은 집계되지 않으므로 항상 Google이 우선합니다.

이 처리기는 두 제목을 모두 기록하고, 태그가 `office`인 항목은 로컬 버전을 유지하며, 태그가 `private`인 항목은 Google로 보내지 않습니다.

상태 파일

상태는 JSON(UTF-8)입니다. 형식 이름과 버전, 캘린더 ID, 동기화 토큰, 시간대, 그리고 동기화된 각 항목에 대해 로컬 ID, 이벤트 ID, etag, `updated` 시간, 동기화된 필드의 지문, 항목이 변경된 시간, 추가 반복 줄이 들어 있습니다. StateFileName은 첫 동기화 전에 이를 불러오고 매 동기화 후에 임시 파일을 거쳐 저장합니다(그 후 이름이 변경됨). SaveStateToStream과 LoadStateFromStream으로 다른 곳에 보관할 수 있습니다. CalendarId가 바뀌면 다른 캘린더의 상태는 지워집니다.
상태는 그것이 설명하는 항목과 함께 보관하십시오. Data에 없는 동기화된 항목은 로컬에서 삭제된 것으로 간주되며, 다음 동기화가 Google에서 해당 이벤트를 삭제합니다. 항목을 저장하지 않는 경우 StateFileName을 비워 두십시오. 그러면 시작할 때마다 빈 Planner로의 전체 동기화로 시작합니다. ResetState는 아무것도 삭제하지 않고 전체 동기화를 강제합니다.

오류와 전체 동기화

- 목록을 가져오는 중의 오류는 아무것도 변경하지 않습니다. 전송 중의 오류는 처음 실패한 호출에서 멈춥니다. 상태는 성공한 것과 이전 동기화 토큰을 유지하므로, 다음 동기화는 같은 원격 변경을 다시 읽고 나머지를 재시도합니다. OnError가 이유를 알려 줍니다.
- 만료된 동기화 토큰에 Google이 410 Gone으로 응답하면 토큰이 지워지고 실행이 전체 목록으로 반복됩니다. 항목의 매핑은 유지되므로 아무것도 두 번 삽입되지 않습니다. 전체 목록에 없는 매핑된 이벤트는 Google에서 삭제된 것으로 간주되며, 편집이 삭제보다 우선한다는 규칙이 적용됩니다.
- 404 또는 410으로 응답받은 업데이트는 삽입으로 다시 전송됩니다. 404 또는 410으로 응답받은 삭제는 완료된 것으로 간주됩니다.

Google 동기화와 iCalendar 구독을 함께 사용

TsgcPlannerICalSubscription은 읽기 전용 피드를 같은 데이터에 병합할 수 있습니다. 해당 항목은 IdPrefix `ics:`를 받는데, 이는 Google 동기화의 기본 ExcludeIdPrefix이므로 피드 이벤트는 표시되지만 Google로 전송되지 않습니다. 한쪽 접두사를 변경하면 다른 쪽도 변경하십시오.

데모

Delphi 및 C++Builder 데모 GoogleCalendarSync와 ICalSubscription은 TsgcVCLScheduler에서 두 구성 요소를 보여 줍니다. `--test`로 실행하면 Google 계정 없이 테스트용 데이터로 오프라인 자체 테스트를 수행합니다. 데모를 참조하십시오.

관련 항목

- TsgcPlannerGoogleSync
- TsgcPlannerICalSubscription
- .NET의 Google 캘린더 동기화와 iCalendar
- 반복
- iCalendar
- TsgcPlannerData

TsgcPlannerGoogleSync

유닛: sgcPlannerWS_Google
TsgcPlannerGoogleSync는 TsgcPlannerData의 항목을 하나의 Google 캘린더와 양방향으로 동기화된 상태로 유지합니다. sgcPlannerWS 브리지 패키지(Registered 에디션)의 일부이며 sgcWebSockets의 **TsgcHTTPGoogleCloud_Calendar_Client**를 통해 Google Calendar API를 호출합니다. 클라이언트는 인증된 전송 수단으로만 사용됩니다. 구성 요소는 클라이언트의 API 수준 메서드(Event_List, Event_Insert, Event_Patch, Event_Delete, Calendar_Get)를 호출하고 이벤트 JSON을 직접 읽고 쓰므로, 캘린더 ID `primary`가 동작하며 클라이언트의 OnError는 애플리케이션에 맡겨집니다. Google Cloud 설정, 필드 매핑, 충돌 규칙, 상태 파일은 Google 캘린더 동기화를 참조하십시오.

- 동기화는 변경된 이벤트를 나열하고(저장된 동기화 토큰이 있으면 그것을 사용하고 없으면 모두, nextPageToken을 따라), 메인 스레드에서 Data에 적용한 다음, 로컬 변경(삽입, 업데이트, 삭제)을 전송하고 새 이벤트 ID, etag, 지문, 동기화 토큰을 상태에 저장합니다.
- 로컬 변경은 지문으로 찾습니다. 동기화된 필드가 마지막 동기화와 다른 항목이 전송됩니다. 항목별 수정 시간은 필요하지 않습니다.
- 양쪽에서 변경된 항목은 충돌입니다. 나중의 변경이 우선하고, 동시이면 Google이 우선하며, OnConflict로 이를 재정의할 수 있습니다. 편집은 항상 삭제보다 우선합니다.
- HTTP 또는 권한 부여 오류는 실행을 중지하고, 이전 동기화 토큰을 유지하며(다음 동기화에서 재시도), OnError를 발생시킵니다.

속성

속성	형식	설명
Active	Boolean	기본값은 True입니다. Active가 False인 동안 Sync와 SyncAsync는 아무 일도 하지 않고 False를 반환합니다. AutoSyncInterval의 자동 동기화는 Active인 동안에만 실행됩니다.
AutoSyncInterval	Integer	두 자동 동기화 사이의 초 단위 간격이며, 0(기본값)이면 끕니다. Active인 동안 0보다 큰 값은 작업자 스레드에서 즉시 동기화한 다음 AutoSyncInterval초마다 동기화합니다.
CalendarId	string	동기화할 캘린더. 기본값은 `primary` 이며 계정의 기본 캘린더입니다. 다른 캘린더의 ID(Google 캘린더의 해당 캘린더 설정에 표시됨)도 사용할 수 있습니다.
Client	TsgcHTTPGoogleCloud_Calendar_Client	인증하고 요청을 보내는 sgcWebSockets의 Google 캘린더 클라이언트. Busy인 동안에는 애플리케이션이 이를 사용해서는 안 됩니다.
Data	TsgcPlannerData	동기화할 Planner 데이터.
Direction	TsgcPlannerGoogleDirection	gsdTwoWay(기본값)는 읽고 씁니다. gsdPullOnly는 Google에 쓰지 않으며 로컬 변경은 집계되지 않으므로 항상 Google이 우선합니다. gsdPushOnly는 로컬 변경만 전송하고 Google에서 이루어진 변경은 읽지 않습니다.
ExcludeIdPrefix	string	Id가 이것으로 시작하는 항목은 전송되지 않습니다. 기본값은 `ics:` 이며 TsgcPlannerICalSubscription의 기본 IdPrefix와 같으므로 구독의 이벤트는 Google로 가지 않습니다.

StateFileName	string	동기화 상태의 JSON 파일. 첫 동기화 전에 불러오고 매 동기화 후에 저장합니다. 비어 있으면(기본값) 상태를 메모리에만 보관합니다.
TimeZone	string	반복 이벤트와 그 인스턴스와 함께 기록되는 IANA 시간대 이름(예: `Europe/Madrid`)이며 Google이 요구합니다. 비어 있으면(기본값) 한 번 읽은 캘린더의 시간대를 사용합니다.
Busy	Boolean	Public, 읽기 전용. 동기화가 실행되는 동안 True입니다.
LastSyncTime	TDateTime	Public, 읽기 전용. 마지막으로 성공한 동기화가 끝난 시점의 로컬 시간.
State	TsgcPlannerGoogleSyncState	Public, 읽기 전용. 동기화 상태로, 동기화 토큰, 캘린더 ID, 시간대, 매핑된 항목당 하나의 항목이 들어 있습니다. Busy가 아닐 때만 변경하십시오.

메서드

메서드	반환 값	설명
Sync	Boolean	호출한(메인) 스레드에서 동기화를 실행하고 끝나면 반환합니다. Active가 아닐 때, Busy일 때(OnError가 알려 줌), 또는 실행이 실패했을 때(OnError가 이유를 알려 줌)는 False입니다.
SyncAsync	Boolean	작업자 스레드에서 동기화를 시작하고 즉시 반환합니다. Data가 변경되며 모든 이벤트는 Synchronize를 통해 메인 스레드에서 발생합니다. Active가 아니거나 Busy이면 False입니다.
ResetState		동기화 토큰과 매핑된 항목을 잊어서 다음 동기화가 전체 동기화가 되게 합니다. Busy인 동안에는 무시됩니다.
SaveStateToStream		상태를 JSON(UTF-8)으로 기록합니다. StateFileName 대신 자체 데이터와 함께 상태를 보관하는 애플리케이션용입니다.
LoadStateFromStream	Boolean	SaveStateToStream이 기록했거나 StateFileName에 저장된 상태를 읽습니다. 텍스트가 이 형식과 버전의 상태가 아니면 상태를 변경하지 않고 False를 반환합니다.

이벤트

이벤트	형식	설명
OnConflict	TsgcPlannerGoogleConflictEvent	마지막 동기화 이후 여기와 Google에서 변경된 항목입니다. aItem은 로컬 항목이고 aEventJSON은 Google이 반환한 이벤트입니다. aResolution은 gcrLastModified(나중의 변경이 우선하고 동시이면 Google이 우선)로 시작합니다. 이번 실행에서 양쪽을 그대로 두려면 gcrKeepLocal, gcrKeepRemote 또는 gcrSkip을 설정합니다(항목은 변경된 상태로 남음). 삭제에 대한 편집은 여기에서 보고되지 않으며 편집이 우선합니다.
OnError	TsgcPlannerWSErrorEvent	aError는 동기화가 실패한 이유를 알려 줍니다. HTTP 상태, 권한 부여 오류, 또는 이미 실행 중인 동기화입니다. 또

		한 Planner가 펼 수 없는 반복 규칙을 규칙마다 한 번 보고하며, 그 규칙은 변경 없이 계속 동기화됩니다.
OnFilterItem	TsgcPlannerGoogleFilterItemEvent	각 로컬 항목이 전송되기 전에 호출됩니다. 항목을 Google로 보내지 않으려면 aSync를 False로 설정합니다.
OnSyncDone	TsgcPlannerGoogleSyncDoneEvent	동기화가 오류 없이 끝났습니다. aPulled는 여기에 적용된 이벤트 수, aPushed는 Google에 기록된 항목 수, aDeleted는 Google에서 삭제된 이벤트 수, aConflicts는 발견된 충돌 수입니다.

스레딩

- SyncAsync와 자동 동기화는 작업자 스레드에서 Google과 통신하고 메인 스레드에서 한 번의 BeginUpdate와 EndUpdate 안에서 Data를 변경하므로 뷰는 한 번만 다시 그려집니다. 모든 이벤트는 메인 스레드에서 발생합니다.
- Busy인 동안에는 실행 중인 동기화만 Client를 사용합니다. 그동안에는 애플리케이션에서 클라이언트를 호출하지 마십시오.
- 구성 요소를 소멸시키면 자동 동기화가 중지되고 실행 중인 동기화가 끝나기를 기다립니다. 구성 요소를 자체 이벤트 처리기 안에서 해제하지 마십시오.

protected 메서드

전송은 protected 가상 메서드의 집합입니다. 각 메서드는 원시 응답과 HTTP 상태를 반환하며 작업자 스레드(Sync에서는 메인 스레드)에서 실행됩니다. 기본 구현은 CalendarId에 대해 클라이언트를 호출합니다. 하위 클래스에서 이를 재정의하면 데모가 `--test` 스위치로 하는 것처럼 네트워크 없이 동기화를 테스트할 수 있습니다.

메서드	설명
DoListEvents(const aPageToken, aSyncToken: string; out aStatus: Integer): string	이벤트 목록의 한 페이지.
DoInsertEvent(const aJSON: string; out aStatus: Integer): string	이벤트를 삽입합니다.
DoUpdateEvent(const aEventId, aJSON: string; out aStatus: Integer): string	이벤트를 패치합니다. aJSON에는 Planner가 소유한 필드만 들어 있으므로 이벤트의 나머지 필드는 유지됩니다.
DoDeleteEvent(const aEventId: string; out aStatus: Integer): string	이벤트를 삭제합니다. 404와 410은 완료된 것으로 간주됩니다.
DoGetCalendar(out aStatus: Integer): string	이벤트 목록이 시간대를 제공하지 않았을 때 시간대를 얻기 위해 캘린더를 읽습니다.
DoError(const aError: string)	OnError를 발생시킵니다.

예제

폼이나 서비스를 위한 동기화 개체. 데스크톱 앱 클라이언트를 사용한 OAuth2, 실행 파일 옆에 저장되는 상태, 5분마다의 자동 동기화, 로그입니다.

관련 항목

- Google 캘린더 동기화
- TsgcPlannerICalSubscription
- .NET의 Google 캘린더 동기화와 iCalendar

- TsgcPlannerData
- 반복
- 에디션
- 데모

TsgcPlannerICalSubscription

유닛: sgcPlannerWS_ICal
TsgcPlannerICalSubscription은 TsgcPlannerData를 URL에 게시된 읽기 전용 iCalendar 피드(.ics)에 구독시킵니다. 예를 들어 Google, Outlook, Apple 캘린더의 공개 또는 비공개 주소, 팀 캘린더, 공휴일 목록이 있습니다. sgcWebSockets의 HTTP 클라이언트로 피드를 다운로드하고 TsgcPlannerICalendar로 읽어서 이벤트를 데이터에 병합합니다. sgcPlannerWS 브리지 패키지(Registered 에디션, SGC_PLANNER_ADVANCED)의 일부입니다.

다운로드

- `webcal://` 또는 `webcals://`로 시작하는 URL은 `https://`로 가져옵니다.
- 구성 요소는 최대 5번의 리디렉션(301, 302, 303, 307, 308)을 직접 따라가며, 상대 Location 값은 현재 URL을 기준으로 해석합니다.
- 마지막으로 가져온 응답의 ETag와 Last-Modified를 사용해 If-None-Match와 If-Modified-Since를 보냅니다. 304 Not Modified이면 가져오기를 건너뜁니다. URL 또는 IdPrefix가 변경되면 이 검증 값은 잊혀집니다.
- MaxSize보다 큰 본문, 그 밖의 상태, 전송 오류, BEGIN:VCALENDAR가 없는 피드는 OnError를 발생시키고 항목과 검증 값을 그대로 두므로 다음 새로 고침에서 재시도합니다.

병합

- 피드의 각 항목은 IdPrefix에 해당 ID를 붙인 ID를 받습니다. ID는 이벤트의 UID이며, 수정된 발생은 UID#RECURRENCE-ID입니다. ParentId와 RecurrenceParentId도 같은 접두사를 받습니다.
- 새 ID는 추가되고, 필드가 다른 기존 ID는 통째로 교체되며, 더 이상 피드에 없는 구독의 항목은 삭제됩니다. 이 모두가 Data의 한 번의 BeginUpdate와 EndUpdate 안에서 이루어집니다.
- 항목의 Id가 IdPrefix로 시작하면 그 항목은 구독에 속합니다. IdPrefix가 비어 있으면 마지막 새로 고침의 항목이 속합니다(메모리에만 보관). Data의 다른 항목은 절대 건드리지 않으므로 로컬 항목과, 각각 자체 IdPrefix를 가진 여러 구독이 하나의 데이터 개체를 공유합니다.
- 피드의 RESOURCES 이름으로 Data의 리소스를 찾고, 새 리소스는 Data에 추가됩니다.
- 구독 항목에 대한 로컬 편집은 가져오기를 수행하는 다음 새로 고침에서 덮어써집니다. 예제처럼 Data의 OnItemChanging에서 IsSubscriptionItem으로 거부하십시오.

속성

속성	형식	설명
Active	Boolean	기본값은 False입니다. True이면 작업자 스레드가 시작되어 즉시 새로 고치고 그 후 RefreshInterval분마다 새로 고칩니다(RefreshInterval이 0이면 한 번만).
Data	TsgcPlannerData	피드가 병합되는 Planner 데이터.
IdPrefix	string	피드 항목의 ID 접두사. 기본값은 `ics:` 이며 TsgcPlannerGoogleSync의 기본 ExcludeIdPrefix이기도 하므로 피드 이벤트는 Google로 전송되지 않습니다.
MaxSize	Integer	허용되는 가장 큰 본문 크기(바이트). 기본값은 10485760(10 MB)이며, 0 이하이면 제한이 없습니다.
RefreshInterval	Integer	Active인 동안 두 새로 고침 사이의 분 단위 간격이며, 0(기본값)이면 한 번만 새로 고칩니다.
TLSOptions	TsgcHTTPTLS_Options	HTTPS 피드를 위한, 내부 HTTP 클라이언트의 TLS 옵션.
URL	string	피드의 주소: https://, http://, webcal:// 또는 webcals://.

Busy	Boolean	Public, 읽기 전용. 새로 고침이 실행되는 동안 True입니다.
HTTPClient	TsgcHTTP1Client	Public, 읽기 전용. 프록시나 기타 설정을 위한 sgcWebSockets의 내부 HTTP 클라이언트. Busy인 동안에는 실행 중인 새로 고침만 이를 사용합니다.
LastRefreshTime	TDateTime	Public, 읽기 전용. 마지막으로 성공한 새로 고침의 로컬 시간이며, 없으면 0입니다.

메서드

메서드	반환 값	설명
Refresh	Boolean	호출한(메인) 스레드에서 피드를 다운로드하고 가져옵니다. Busy이거나 오류가 있으면 False입니다.
RefreshAsync	Boolean	작업자 스레드에서 새로 고침을 시작하고 즉시 반환합니다. 가져오기와 모든 이벤트는 메인 스레드에서 실행됩니다. Busy이면 False입니다.
IsSubscriptionItem(aItem)	Boolean	aItem이 이 구독에 속하면 True입니다.

이벤트

이벤트	형식	설명
OnBeforeRefresh	TNotifyEvent	새로 고침이 시작됩니다. 다운로드 전입니다.
OnError	TsgcPlannerWSErrorEvent	새로 고침이 실패했습니다. aError는 HTTP 상태, 크기 제한, 너무 많은 리디렉션, 전송 오류, 또는 구문 분석되지 않는 피드를 알려 줍니다. 항목은 그대로 남습니다.
OnImportWarning	TsgcPlannerICalWarningEvent	TsgcPlannerICalendar가 건너뛰었거나 근사적으로 읽은 피드의 일부이며 aLine과 aMessage가 함께 전달됩니다.
OnRefreshed	TsgcPlannerICalRefreshedEvent	새로 고침이 성공했습니다. aAdded, aUpdated, aDeleted는 항목 수입니다. 304 Not Modified 응답이면 0, 0, 0이 됩니다.

스레딩

Refresh는 호출한 스레드를 차단합니다. RefreshAsync와 Active 루프는 작업자 스레드에서 다운로드하고 Synchronize를 통해 메인 스레드에서 가져오므로 모든 이벤트가 메인 스레드에서 발생합니다. 구성 요소를 소멸시키면 작업자가 중지됩니다. 구성 요소를 자체 이벤트 처리기 안에서 해제하지 마십시오.

protected 메서드

메서드	설명
DoFetch(const aURL: string; aRequestHeaders, aResponseHeaders: TStrings; aBody: TStream): Integer	리디렉션을 따라가지 않는 aURL에 대한 한 번의 GET. aRequestHeaders는 요청 헤더 줄(이름: 값)을 담고, aResponseHeaders는 응답 헤더 줄을, aBody는 본문을 받습니다. HTTP 상태를 반환하며 전송 오류 시 예외를 발생시킵니다. 새로 고침의 스레드에서 호출되며, 테스트와 데모는 로컬 파일을 읽기 위해 이를 재정의합니다.

DoGetRefreshIntervalMs: Cardinal	Active 루프의 밀리초 단위 간격.
DoError, DoRefreshed	OnError와 OnRefreshed를 발생시킵니다.

예제

로컬 항목 옆에 표시되는 팀 캘린더로, 한 시간마다 새로 고쳐지며 뷰에서는 피드 이벤트가 읽기 전용입니다.

관련 항목

- TsgcPlannerICalendar
- TsgcPlannerGoogleSync
- Google 캘린더 동기화
- iCalendar
- .NET의 Google 캘린더 동기화와 iCalendar
- TsgcPlannerData
- 반복
- 에디션
- 데모

.NET 개요

sgcPlanner .NET은 Delphi sgcPlanner 라이브러리를 감싼 래퍼가 아니라 C#으로 이식한 관리형 포트입니다. C# 클래스는 **esegece.sgcPlanner** 네임스페이스에 있고 Delphi 이름을 유지하므로, **TsgcPlannerData**, **TsgcPlannerItem**, **TsgcPlannerCalendar**, **TsgcPlannerRenderer**와 열거형은 두 언어에서 똑같이 읽힙니다. 세 가지 뷰는 모든 호스트에서 Delphi 명명 규칙을 따릅니다. **TsgcVCLGantt**는 **TsgcWinFormsGantt**, **TsgcWpfGantt**, **TsgcAvaloniaGantt**이며 Scheduler와 Kanban 보드도 마찬가지입니다. 뷰는 **sgcUICore**의 .NET 포트로 그려지므로, eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 밝은 테마와 어두운 테마를 사용합니다.
Delphi와 마찬가지로 하나의 **TsgcPlannerData**가 항목, 리소스, 상태, 종속성, 근무 달력을 보관하고 원하는 수만큼의 뷰가 이를 보여 줍니다. 한 뷰에서 한 편집은 데이터를 거치며, 데이터는 바인딩된 모든 뷰에 한 번만 알립니다. .NET 데이터와 일정 계산을 참조하십시오.

패키지

포트는 다섯 개의 NuGet 패키지로 나뉘어 있으므로, 프로젝트는 사용하는 호스트만 가져옵니다.

PackageId	내용	대상 프레임워크
esegece.sgcPlanner	데이터, 근무 달력, 시간 눈금, 반복, 일정 계산 엔진, Scheduler, 리소스 Timeline, Gantt, Kanban 보드의 뷰 모델과 페인터, **System.Data.DataTable** 소스, JSON, iCalendar, MS Project XML, 뷰의 SVG 출력입니다. 네이티브 종속성이 없습니다. esegece.sgcUICore에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcPlanner.Rendering	헤드리스 PNG 출력입니다. **TsgcPlannerRenderer**와 **sgcPlanner_Raster**의 확장 메서드 **SaveToPNG**, **ExportPNG**, **RenderToRaster**가 있으며, Windows, Linux, macOS에서 esegece.sgcUICore.Rendering의 SkiaSharp 캔버스로 그립니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcPlanner.WinForms	**TsgcPlannerWinFormsView** 기반의 **TsgcWinFormsScheduler**, **TsgcWinFormsGantt**, **TsgcWinFormsKanban**입니다.	net472, net8.0-windows, net9.0-windows
esegece.sgcPlanner.Wpf	**TsgcPlannerWpfView** 기반의 **TsgcWpfScheduler**, **TsgcWpfGantt**, **TsgcWpfKanban**입니다.	net472, net8.0-windows, net9.0-windows
esegece.sgcPlanner.Avalonia	**TsgcPlannerAvaloniaView** 기반의 **TsgcAvaloniaScheduler**, **TsgcAvaloniaGantt**, **TsgcAvaloniaKanban**이며, Avalonia 11로 Windows, Linux, macOS에서 동작합니다.	net8.0, net9.0

세 호스트 패키지는 esegece.sgcPlanner에 종속된 esegece.sgcPlanner.Rendering에 종속되므로, `PackageReference` 한 줄로 호스트에 필요한 모든 것이 설치됩니다. NuGet으로 설치을 참조하십시오. sgcWebSockets 브리지의 Google 캘린더 동기화와 iCalendar 구독은 별도 패키지인 **esegece.sgcPlanner.sgcWebSockets**에 있으며, Registered 에디션에서만 제공됩니다.

에디션

에디션	패키지 ID	포함 내용
Registered	esegece.sgcPlanner, .Rendering, .WinForms, .Wpf, .Avalonia	모든 것이 들어 있습니다. 세 호스트의 세 가지 뷰, 끌어서 놓기, 요약, 마일스톤, 진행률, 테마, DataTable 소스, JSON, 헤드

		리스 PNG 및 SVG 출력, 그리고 고급 기능인 반복, 종속성 화살표를 갖춘 일정 계산 엔진, 크리티컬 패스와 기준선, 리소스 Timeline, 보드의 스윔레인과 WIP 제한, iCalendar와 MS Project XML이 포함됩니다.
Community	같은 ID에 .Community 접미사가 붙습니다. 예를 들어 esegece.sgcPlanner.WinForms.Community입니다.	고급 기능을 제외한 모든 것입니다. 속성은 그대로 남아 있으므로 같은 폼 코드가 두 에디션에서 컴파일됩니다. 반복 항목은 시작 위치에 한 번만 표시되고, **ShowDependencies**, **AutoSchedule**, **ShowCriticalPath**, **ShowBaseline**은 아무 동작도 하지 않으며, Gantt의 **Schedule**과 **SetBaseline**은 **EsgcPlannerError**를 발생시키고, **psvTimeline**은 안내 문구를 그리며, **SwimlaneBy**는 **psINone**처럼 동작합니다. 고급 유닛의 클래스는 Community 패키지에 없습니다. **TsgcPlannerScheduleEngine**, **TsgcPlannerICalendar**, **TsgcPlannerMSProject**, **TsgcPlannerSchedulerTimelineView**와 반복 클래스가 그렇습니다.

이는 Delphi 에디션과 같은 구분입니다. 에디션별 기능을 참조하십시오.

호스트

호스트	Delphi 대응 항목	플랫폼
WinForms	VCL(TsgcVCL 컨트롤)	Windows, .NET Framework 4.7.2, .NET 8 및 9.
WPF	VCL과 FMX	Windows, .NET Framework 4.7.2, .NET 8 및 9. 컨트롤을 XAML에서 선언할 수 있습니다.
Avalonia	FMX(TsgcFMX 컨트롤)	Windows, Linux, macOS, .NET 8 및 9.
헤드리스	TsgcPlannerRenderer와 뷰 모델	대상 프레임워크의 모든 플랫폼입니다. SVG는 esegece.sgcPlanner로, PNG는 esegece.sgcPlanner.Rendering으로 만듭니다.

모든 호스트의 API는 같습니다. Delphi 컨트롤과 이름이 같은 속성, 메서드, 이벤트에 더해 기본 클래스에 **Data**, **Theme**, **Colors**, **ReadOnly**, **DragOptions**, **SyncSelection**, **ItemAt**, **ScrollToItem**, **SaveToPNG**, **SaveToSVG**가 있습니다. 클래스 이름은 .NET 클래스를, Delphi에서 달라지는 점은 Delphi와 .NET의 차이를 참조하십시오.

관련 항목

- NuGet으로 설치
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 클래스
- Delphi와 .NET의 차이
- .NET 데이터와 일정 계산
- .NET 데이터 바인딩
- .NET JSON, iCalendar, MS Project
- .NET 헤드리스 렌더링
- .NET 데모
- 플랫폼

NuGet으로 설치

패키지는 nuget.org에 있습니다. 호스트 패키지를 추가하면 코어 패키지와 렌더링 패키지가 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcPlanner.WinForms
dotnet add package esegece.sgcPlanner.Wpf
dotnet add package esegece.sgcPlanner.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcPlanner.WinForms.Community
dotnet add package esegece.sgcPlanner.Wpf.Community
dotnet add package esegece.sgcPlanner.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가하십시오.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcPlanner.Wpf" Version="2026.10.0" />
</ItemGroup>
```

프로젝트에서는 Registered 패키지와 Community 패키지 중 하나만 참조하고 둘 다 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있기 때문입니다.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`가 있는 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트입니다.
- WPF: `<UseWPF>true</UseWPF>`가 있는 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트입니다. XAML에서는 `xmlns:sgc="clr-namespace:esegece.sgcPlanner;assembly=esegece.sgcPlanner.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: 다른 Avalonia 데스크톱 애플리케이션과 마찬가지로 `net8.0` 또는 `net9.0` 프로젝트에 Avalonia 11.3의 **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 패키지를 추가합니다. 뷰의 스크롤 막대는 Fluent 테마에서 모양을 가져옵니다. XAML에서는 `xmlns:sgc="using:esegece.sgcPlanner"`로 네임스페이스를 매핑합니다.
- 헤드리스: 데이터, 파일 형식, SVG에는 esegece.sgcPlanner만으로 충분합니다. PNG에는 esegece.sgcPlanner.Rendering을 추가하십시오. SkiaSharp와 Windows 및 Linux용 네이티브 자산이 함께 설치됩니다.
- Google 캘린더 동기화와 iCalendar 구독: esegece.sgcPlanner와 esegece.sgcWebSockets에 종속된 **esegece.sgcPlanner.sgcWebSockets**(Registered 에디션)를 추가하십시오.

C#에서는 모든 호스트에 `using esegece.sgcPlanner;`가 필요하고, `using esegece.sgcUICore;`는 **TsgcUITheme**, **TsgcUINotifyEvent** 등 sgcUICore의 다른 형식에 필요합니다.

관련 항목

- .NET 개요
- WinForms 빠른 시작
- WPF 빠른 시작
- Avalonia 빠른 시작
- .NET 데모

WinForms 빠른 시작

이 빠른 시작은 하나의 **TsgcPlannerData**에 바인딩된 세 가지 WinForms 뷰로 폼을 코드로 만듭니다. 위쪽에 Gantt 차트, 왼쪽에 Kanban 보드, 오른쪽에 주중 보기가 있습니다. 카드를 다른 열로 끌면 Gantt의 상태 열이 따라갑니다.

단계

1. `net8.0-windows`용 Windows Forms 프로젝트를 만들고 **esegece.sgcPlanner.WinForms** 패키지(Community 에디션에서는 **esegece.sgcPlanner.WinForms.Community**)를 추가합니다.
2. `using esegece.sgcPlanner;`와 `using esegece.sgcUICore;`를 추가합니다.
3. **TsgcPlannerData**를 만들고 채웁니다. 상태, 리소스, 항목입니다.
4. 컨트롤을 만들고 **Data**를 대입한 다음 폼에 추가합니다. 일반 WinForms 컨트롤이므로 Dock, Anchor, TabIndex, Enabled가 평소처럼 동작합니다.
5. 이벤트를 대입합니다. Delphi 컨트롤의 이벤트이며 이름이 같습니다.

예제

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcPlanner;
using esegece.sgcUICore;

public class PlannerForm : Form
{
    public readonly TsgcPlannerData Data = new TsgcPlannerData();
    public readonly TsgcWinFormsGantt Gantt;
    public readonly TsgcWinFormsKanban Kanban;
    public readonly TsgcWinFormsScheduler Scheduler;
    private readonly Label lblStatus;

    public PlannerForm()
    {
        Text = "sgcPlanner";
        ClientSize = new Size(1200, 760);

        // the statuses are the columns of the board
        Data.Statuses.AddStatus("todo", "To do", 0xFF94A3B8);
        Data.Statuses.AddStatus("doing", "In progress", 0xFF3B82F6);
        Data.Statuses.AddStatus("done", "Done", 0xFF22C55E);
        Data.Resources.AddResource("ann", "Ann Lee");
        TsgcPlannerItem oItem = Data.AddItem("t1");
        oItem.Title = "Write the specification";
        oItem.Start = new DateTime(2026, 10, 5, 9, 0, 0);
        oItem.Finish = new DateTime(2026, 10, 6, 18, 0, 0);
        oItem.Status = "doing";
        oItem.Resources = "ann";

        lblStatus = new Label { Dock = DockStyle.Bottom, Height = 24 };

        // three views of the same data: an edit in one is seen by the others
        Gantt = new TsgcWinFormsGantt { Dock = DockStyle.Top, Height = 260, Data = Data };
        Gantt.Columns.Clear();
        Gantt.Columns.AddColumn(TsgcPlannerGanttField.pgcTitle, 200);
        Gantt.Columns.AddColumn(TsgcPlannerGanttField.pgcStatus, 90);
        Gantt.TreeWidth = 290;
        Gantt.BarColorMode = TsgcPlannerBarColorMode.pbcStatus;

        Kanban = new TsgcWinFormsKanban { Dock = DockStyle.Left, Width = 600, Data = Data };
        Kanban.OnCardMoved = (Sender, aItem) => lblStatus.Text = aItem.Title + " is now " + aItem.Status;
```

```
        Scheduler = new TsgcWinFormsScheduler { Dock = DockStyle.Fill, Data = Data };
        Scheduler.View = TsgcPlannerSchedulerViewKind.psvWorkWeek;
        Scheduler.Date = new DateTime(2026, 10, 5);
        Scheduler.Options.DayStartHour = 8;
        Scheduler.Options.DayEndHour = 19;
        Scheduler.OnItemDblClick = (Sender, aItem) => MessageBox.Show(aItem.Title);

        // Fill first, then the docked edges
        Controls.Add(Scheduler);
        Controls.Add(Kanban);
        Controls.Add(Gantt);
        Controls.Add(lblStatus);
    }

    public void SetDark(bool aDark)
    {
        TsgcUITheme vTheme = aDark ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        Gantt.Theme = vTheme;
        Kanban.Theme = vTheme;
        Scheduler.Theme = vTheme;
    }
}
```

참고

- Kanban 보드는 **Data.Statuses**의 상태마다 열을 하나씩 만듭니다(**ColumnSource**는 **pkcStatuses**). 열과 WIP 제한을 직접 고르려면 **ColumnSource**를 **pkcList**로 설정하고 **Columns**를 채우십시오.
- Scheduler의 **View** 속성은 Delphi와 마찬가지로 뷰의 종류입니다(**psvDay**, **psvWorkWeek**, **psvWeek**, **psvMonth**, **psvAgenda**, **psvTimeline**). 뷰 모델은 **SchedulerView**이며, Gantt와 보드는 각각 **GanttView**와 **KanbanView**로 노출합니다.
- **Theme**은 컨트롤마다 설정합니다. 스크롤 막대는 Windows 기본 막대이며, **BorderStyle**로 가장자리를 선택합니다.
- **SaveToPNG**와 **SaveToSVG**는 컨트롤이 보여 주는 내용을 컨트롤의 크기 또는 다른 크기와 배율로 기록합니다.

관련 항목

- .NET 개요
- NuGet으로 설치
- .NET 클래스
- .NET 데이터와 일정 계산
- VCL 빠른 시작

WPF 빠른 시작

WPF 컨트롤은 다른 컨트롤과 마찬가지로 XAML에서 선언할 수 있습니다. 이 빠른 시작은 Gantt, Kanban 보드, Scheduler를 그리드에 선언하고 코드 숨김에서 하나의 **TsgcPlannerData**에 바인딩합니다.

단계

1. `net8.0-windows`용 WPF 프로젝트를 만들고 **esegece.sgcPlanner.Wpf** 패키지(Community 에디션에서는 **esegece.sgcPlanner.Wpf.Community**)를 추가합니다.
2. 창에서 네임스페이스를 매핑합니다. `xmlns:sgc="clr-namespace:esegece.sgcPlanner;assembly=esegece.sgcPlanner.Wpf"`.
3. 컨트롤을 선언합니다. 단순 속성은 특성으로 설정할 수 있고, 열거형은 `View="psvWorkWeek"`처럼 값 이름을 받습니다. 늘어나지 않는 패널 안에서 Width와 Height가 없는 컨트롤은 400 x 300을 요청합니다.
4. 코드 숨김에서 데이터를 만들고 각 컨트롤의 **Data**에 대입합니다.

예제

창:

```
<Window x:Class="QuickStart.QuickStartWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcPlanner;assembly=esegece.sgcPlanner.Wpf"
        Title="sgcPlanner" Width="1100" Height="700">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="260" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="560" />
      <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <sgc:TsgcWpfGantt x:Name="ganTasks" Grid.ColumnSpan="2" TreeWidth="260" ShowToday="False" />
    <sgc:TsgcWpfKanban x:Name="kbnBoard" Grid.Row="1" />
    <sgc:TsgcWpfScheduler x:Name="schWeek" Grid.Row="1" Grid.Column="1" View="psvWorkWeek" />
  </Grid>
</Window>
```

코드 숨김:

```
using System;
using System.Windows;
using esegece.sgcPlanner;

namespace QuickStart
{
    public partial class QuickStartWindow : Window
    {
        public readonly TsgcPlannerData Data = new TsgcPlannerData();

        public QuickStartWindow()
        {
            InitializeComponent();

            Data.Statuses.AddStatus("todo", "To do");
            Data.Statuses.AddStatus("doing", "In progress");
            Data.Statuses.AddStatus("done", "Done");
            TsgcPlannerItem oItem = Data.AddItem("t1");
            oItem.Title = "Prepare the release";
            oItem.Start = new DateTime(2026, 10, 6, 10, 0, 0);
```

```
            oItem.Finish = new DateTime(2026, 10, 6, 12, 30, 0);
            oItem.Status = "todo";

            // the data is set in code: the three controls share it
            ganTasks.Data = Data;
            kbnBoard.Data = Data;
            schWeek.Data = Data;
            schWeek.Date = new DateTime(2026, 10, 5);
            schWeek.Options.DayStartHour = 8;
            schWeek.Options.DayEndHour = 18;

            kbnBoard.OnCardMoved = (Sender, aItem) => Title = aItem.Title + ": " + aItem.Status;
            Closed += (s, e) =>
            {
                // stop the timers and detach the data listeners
                ganTasks.Dispose();
                kbnBoard.Dispose();
                schWeek.Dispose();
            };
        }
    }
}
```

참고

- 속성은 종속성 속성이 아닌 일반 CLR 속성이므로 XAML에서 리터럴 값은 받지만 `{Binding}`은 받지 않습니다. 계획의 데이터는 **TsgcPlannerData**를 통해 바인딩합니다. .NET 데이터 바인딩을 참조하십시오.
- 스크롤 막대는 컨트롤의 WPF `ScrollBar` 자식 요소입니다. WinForms의 **TabStop** 대신 **IsTabStop**을 사용하며 **BorderStyle**은 없습니다. BorderBrush와 BorderThickness 또는 Border를 사용하십시오.
- 컨트롤은 로드된 동안 현재 시간선을 위한 타이머를 실행합니다. **Dispose**는 타이머를 멈추고 데이터 리스너를 분리하므로, 창이 닫힐 때 호출하십시오.

관련 항목

- .NET 개요
- NuGet으로 설치
- .NET 클래스
- .NET 데이터 바인딩
- TsgcVCLGantt

Avalonia 빠른 시작

Avalonia 컨트롤은 Windows, Linux, macOS에서 실행됩니다. 이 빠른 시작은 코드로 작성한 완전한 Avalonia 애플리케이션입니다. 리소스별로 그룹화한 주 보기 옆에 Kanban 보드를 두며, 둘 다 어두운 테마이고 하나의 **TsgcPlannerData**에 바인딩됩니다.

단계

1. `net8.0` 콘솔 프로젝트를 만들고 `<OutputType>WinExe</OutputType>`를 설정한 다음, **Avalonia.Desktop**, **Avalonia.Themes.Fluent** 11.3과 **esegece.sgcPlanner.Avalonia** 패키지(Community 에디션에서는 **esegece.sgcPlanner.Avalonia.Community**)를 추가합니다.
2. 애플리케이션에 Fluent 테마를 추가합니다.
3. 데이터와 컨트롤을 만들고 컨트롤을 패널에 추가합니다. XAML에서는 `xmlns:sgc="using:esegece.sgcPlanner"`로 네임스페이스를 매핑합니다.

예제

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;
using esegece.sgcPlanner;
using esegece.sgcUICore;

public class App : Application
{
    public override void Initialize()
    {
        Styles.Add(new FluentTheme());
    }

    public override void OnFrameworkInitializationCompleted()
    {
        if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime vDesktop)
            vDesktop.MainWindow = new PlannerWindow();
        base.OnFrameworkInitializationCompleted();
    }
}

public class PlannerWindow : Window
{
    public readonly TsgcPlannerData Data = new TsgcPlannerData();
    public readonly TsgcAvaloniaScheduler Scheduler;
    public readonly TsgcAvaloniaKanban Kanban;

    public PlannerWindow()
    {
        Title = "sgcPlanner";
        Width = 1100;
        Height = 640;

        Data.Statuses.AddStatus("todo", "To do");
        Data.Statuses.AddStatus("done", "Done");
        Data.Resources.AddResource("ann", "Ann Lee");
        Data.Resources.AddResource("bob", "Bob Stone");
        TsgcPlannerItem oItem = Data.AddItem("m1");
        oItem.Title = "Planning meeting";
        oItem.Start = new DateTime(2026, 10, 7, 10, 0, 0);
        oItem.Finish = new DateTime(2026, 10, 7, 11, 0, 0);
```

```
        oItem.Resources = "ann,bob";
        oItem.Status = "todo";

        // a week grouped by resource: one column per person and day
        Scheduler = new TsgcAvaloniaScheduler { Data = Data };
        Scheduler.View = TsgcPlannerSchedulerViewKind.psvWeek;
        Scheduler.Date = new DateTime(2026, 10, 5);
        Scheduler.Options.GroupByResource = true;
        Scheduler.Theme = TsgcUITheme.utDark;

        Kanban = new TsgcAvaloniaKanban { Data = Data, Width = 420 };
        Kanban.Theme = TsgcUITheme.utDark;

        DockPanel pnlMain = new DockPanel();
        DockPanel.SetDock(Kanban, Dock.Left);
        pnlMain.Children.Add(Kanban);
        pnlMain.Children.Add(Scheduler);
        Content = pnlMain;

        // Avalonia does not dispose controls: release them with the window
        Closed += (s, e) =>
        {
            Scheduler.Dispose();
            Kanban.Dispose();
        };
    }
}

public static class Program
{
    [STAThread]
    public static void Start(string[] args)
    {
        AppBuilder.Configure<App>().UsePlatformDetect().StartWithClassicDesktopLifetime(args);
    }
}
```

참고

- 컨트롤의 **Theme**은 sgcUICore 테마입니다. Avalonia의 **StyledElement.Theme**을 가리므로, 예제처럼 컨트롤 형식의 변수를 통해 설정하십시오.
- **GroupByResource**는 모든 에디션에서 일, 주중, 주 보기의 사람마다 열 그룹을 만들어 줍니다. 리소스 Timeline(**psvTimeline**)은 Registered 기능입니다.
- 속성은 스타일 속성이 아닌 일반 CLR 속성이므로 코드에서 설정하거나 XAML 특성으로 설정합니다.
- macOS에서는 Command 키가 Ctrl로 취급되므로, Command와 휠로 Gantt를 확대/축소합니다.

관련 항목

- .NET 개요
- NuGet으로 설치
- .NET 클래스
- .NET 데이터와 일정 계산
- FireMonkey 빠른 시작

.NET 클래스

모든 Delphi 클래스는 .NET에서도 같은 이름을 유지합니다. VCL과 FireMonkey 컨트롤은 호스트 접두사가 붙은 WinForms, WPF, Avalonia 컨트롤이 되며, 프레임워크 중립 클래스는 모든 호스트에서 동일합니다.

호스트 클래스

Delphi	WinForms	WPF	Avalonia	모델(호스
TsgcVCLScheduler, TsgcFMXScheduler	TsgcWinFormsScheduler	TsgcWpfScheduler	TsgcAvaloniaScheduler	TsgcPlan (Schedul 에서는 TsgcPlan Commun TsgcPlan
TsgcVCLGantt, TsgcFMXGantt	TsgcWinFormsGantt	TsgcWpfGantt	TsgcAvaloniaGantt	TsgcPlan (GanttVie
TsgcVCLKanban, TsgcFMXKanban	TsgcWinFormsKanban	TsgcWpfKanban	TsgcAvaloniaKanban	TsgcPlan (KanbanV
TsgcPlannerVCLView, TsgcPlannerFMXView	TsgcPlannerWinFormsView	TsgcPlannerWpfView	TsgcPlannerAvaloniaView	TsgcPlan

기본 클래스는 sgcUICore의 **TsgcUIWinFormsControl**, **TsgcUIWpfControl**, **TsgcUIAvaloniaControl**에서 파생됩니다.

모든 컨트롤의 멤버

멤버	설명
Data	컨트롤이 보여 주는 **TsgcPlannerData**입니다. 여러 컨트롤이 하나를 공유할 수 있습니다.
View	뷰 모델이며 **TsgcPlannerView**입니다. 형식이 지정된 속성 SchedulerView, GanttView, KanbanView는 같은 개체를 반환합니다. Scheduler에서는 Delphi와 마찬가지로 **View**가 뷰의 종류로 다시 선언됩니다.
Theme, Colors	모양입니다. **TsgcUITheme**(utLight, utDark)과 테마 색을 덮어쓰는 **TsgcPlannerViewColors**가 있습니다.
ReadOnly, DragOptions	편집을 끄거나 사용자가 할 수 있는 끌기 동작(만들기, 이동, 크기 조정, 진행률, 연결, 순서 변경)을 `[Flags]` **TsgcPlannerDragOptions**로 선택합니다.
SyncSelection	선택은 데이터의 **SelectedId**를 따르므로, 같은 데이터에 바인딩된 뷰는 같은 항목을 선택합니다.
ItemAt, ScrollToItem	컨트롤의 한 지점에 있는 항목(WinForms에서는 클라이언트 픽셀, WPF와 Avalonia에서는 DIP)과 항목이 보이도록 스크롤하는 기능입니다.
UpdateScrollBars, CursorKind	모델에서 변경한 뒤 스크롤 막대를 갱신하며, 컨트롤이 표시하는 커서도 다룹니다.
OnItemClick, OnItemDblClick,	Delphi 기본 뷰의 이벤트입니다. OnItemDelete는 삭제를 거부할 수 있습니다.

OnItemDelete, OnSelectionChange	
SaveToPNG, SaveToSVG	현재 모습 그대로의 뷰를 스트림이나 파일에 그립니다. 컨트롤의 크기나 다른 크기, 배율을 지정할 수 있습니다.

프레임워크 중립 형식

형식	설명
TsgcPlannerData	항목, 리소스, 상태, 종속성, 근무 달력, 변경 알림, JSON입니다. TsgcPlannerData를 참조하십시오.
TsgcPlannerItem, TsgcPlannerResource, TsgcPlannerStatus, TsgcPlannerDependency	**TsgcPlannerItems**, **TsgcPlannerResources**, **TsgcPlannerStatuses**, **TsgcPlannerDependencies**에 들어 있는 컬렉션 항목입니다.
TsgcPlannerCalendar, TsgcPlannerHolidays	근무일, 근무 시간, 휴일입니다. 근무 달력를 참조하십시오.
TsgcPlannerTimeScale	Gantt와 Timeline의 확대/축소와 스냅입니다. 시간 눈금과 확대/축소를 참조하십시오.
TsgcPlannerSchedulerView, TsgcPlannerGanttView, TsgcPlannerKanbanView	렌더링 서브클래스를 갖춘 뷰 모델입니다. 레이아웃, 히트 테스트, 키보드, 마우스와 끌기, 그리고 SVG용 **ExportImage**가 있으며 창이 필요 없습니다.
TsgcPlannerScheduleEngine	일정 계산 엔진입니다(Registered). 자동 일정 계산를 참조하십시오.
TsgcPlannerSource_DataSet	데이터를 **System.Data.DataTable** 테이블에 바인딩합니다. .NET 데이터 바인딩을 참조하십시오.
TsgcPlannerICalendar, TsgcPlannerMSProject	iCalendar와 MS Project XML 가져오기 및 내보내기입니다(Registered). .NET JSON, iCalendar, MS Project를 참조하십시오.
TsgcPlannerRenderer	esegece.sgcPlanner.Rendering에 있으며, 세 가지 뷰를 창 없이 PNG와 SVG로 그립니다. .NET 헤드리스 렌더링을 참조하십시오.
TsgcPlannerLocale	헤더 레이블의 요일과 월 이름, 시간 형식, 소수 구분 기호이며 `CultureInfo.CurrentCulture`에서 가져옵니다.
sgcPlanner_DateUtils, sgcPlanner_JSON, sgcPlanner_Raster	Delphi 자유 함수를 담은 정적 클래스입니다. 시간 변환과 시계, JSON 함수, 그리고 esegece.sgcPlanner.Rendering에 있는 뷰의 PNG 확장 메서드가 있습니다.

열거형

열거형은 Delphi의 이름과 값 접두사를 유지합니다. 예를 들면 **TsgcPlannerSchedulerViewKind**(psv), **TsgcPlannerGanttField**(pgc), **TsgcPlannerBarColorMode**(pbc), **TsgcPlannerDurationUnit**(pdu), **TsgcPlannerDependencyKind**(pdk), **TsgcPlannerConstraint**(pc), **TsgcPlannerKanbanColumnSource**(pkc), **TsgcPlannerSwimlaneBy**(psl), **TsgcPlannerWIPMode**(pwm), **TsgcPlannerRecurrenceScope**(prs), **TsgcPlannerRendererViewKind**(prv)입니다. Delphi 집합은 `|`로 결합하는 `[Flags]` 열거형입니다. **TsgcPlannerWeekDays**, **TsgcPlannerChangeKinds**, **TsgcPlannerItemFields**, **TsgcPlannerDragOptions**가 그렇습니다. Gantt 열에 쓰는 Delphi RTL의 **TAlignment**는 esegece.sgcPlanner에 선언되어 있습니다.

관련 항목

- .NET 개요

- Delphi와 .NET의 차이
- 아키텍처

Delphi와 .NET의 차이

.NET 포트는 Delphi의 이름, 기본값, 키보드와 마우스 동작, 레이아웃 규칙을 그대로 유지합니다. 뷰의 SVG 출력은 Delphi의 출력과 바이트 단위로 같고, PNG 출력은 SkiaSharp 래스터라이저의 허용 오차 안에서 일치합니다. 아래의 차이는 호스트 프레임워크와 .NET 클래스 라이브러리에서 비롯됩니다.

시간과 값

- 날짜 속성은 Kind가 Unspecified인 `DateTime` 값입니다. **Start**, **Finish**, **ConstraintDate**, **BaselineStart**, **BaselineFinish**, **RecurrenceDate**, **ProjectStart**가 그렇습니다. 분 단위 정수인 Delphi의 TsgcPlannerTime은 `long`이며, Time 접미사가 붙은 속성(**StartTime**, **FinishTime** 등)으로 접근합니다. **Duration**은 분 단위입니다.
- 시계는 `Func<DateTime>`인 `sgcPlanner_DateUtils.sgcPlannerNow`입니다. 테스트에서 시간을 고정하려면 이것을 교체하고, 끝나면 저장해 둔 델리게이트를 복원하십시오.
- TsgcUIColor는 `uint`이며 $AARRGGBB이고, 0은 테마 또는 상태 색을 뜻합니다.
- TFormatSettings와 RTL 전역 값은 `CultureInfo.CurrentCulture`로 대체되며, 이를 **TsgcPlannerLocale**이 가져옵니다. 인덱스가 있는 요일 및 월 이름 속성은 **GetShortDayName**, **SetShortDayName**, **GetLongMonthName** 등의 메서드입니다. 파일은 항상 고정 형식으로 기록됩니다.

이벤트, 메서드, 목록

- 이벤트는 Delphi와 같은 이름의 델리게이트 속성입니다(**OnChange**, **OnItemCreate**, **OnCardMoving**...). 핸들러 하나는 `=`로 대입합니다. Delphi의 `var` 매개변수는 `ref` 매개변수이며, `ref` 매개변수가 있는 람다는 모든 매개변수의 형식을 선언합니다. **TNotifyEvent**는 **TsgcUINotifyEvent**입니다.
- 데이터는 여러 리스너를 받을 수 있습니다. 뷰와 마찬가지로 **AddListener**와 **RemoveListener**는 **TsgcPlannerChangeEvent**를 받습니다.
- Delphi의 인덱스 속성은 메서드입니다. `Rows(i)`, `ColumnInfo(i)`, `Cards(i)`, `Entries(i)`, `Calendar.Intervals(i)`가 그렇습니다. 매개변수가 없는 Delphi 함수는 `RowCount()`나 `WorkingMinutesPerDay()`처럼 메서드로 남습니다.
- Delphi의 `var` 배열은 `ref` 배열이며, 메서드가 이를 정확한 길이의 배열로 교체합니다. `ChildrenOf(id, ref aList)`, `GetOccurrences(start, finish, ref aList)`가 그렇습니다.
- Delphi 집합은 `|`로 결합하는 `[Flags]` 열거형입니다. 컬렉션은 TCollection의 형태를 유지합니다. Add, Delete, Clear, Assign, BeginUpdate, EndUpdate, 인덱서, `foreach`가 있습니다.

```
using System;
using esegece.sgcPlanner;

public static class DifferencesSample
{
    public static void Setup(TsgcWinFormsScheduler aScheduler, TsgcWinFormsKanban aKanban)
    {
        // a Delphi var parameter is a ref parameter: the lambda declares every parameter type
        aScheduler.OnItemCreate = (object Sender, DateTime aStart, DateTime aFinish, string aResourceId,
            ref string aTitle, ref bool aAccept) =>
        {
            aTitle = "Meeting";
            aAccept = aFinish - aStart <= TimeSpan.FromHours(4);
        };

        // a card can not leave the Done column
        aKanban.OnCardMoving = (object Sender, TsgcPlannerItem aItem, string aToStatus, string aToLane,
            ref bool aAccept) =>
        {
            aAccept = aItem.Status != "done";
        };

        // TNotifyEvent is TsgcUINotifyEvent
        aScheduler.OnDateChange = Sender => Console.WriteLine(aScheduler.Date.ToShortDateString());

        // DateTime properties and the exact minutes
        TsgcPlannerItem oItem = aScheduler.Data.FindItem("t1");
        if (oItem != null)
```

```
        {
            DateTime vStart = oItem.Start;
            long vMinutes = oItem.StartTime;
            oItem.FinishTime = vMinutes + 90;
            Console.WriteLine(vStart + " lasts " + oItem.Duration + " minutes");
        }

        // a Delphi indexed property is a method
        TsgcPlannerKanbanView oBoard = aKanban.KanbanView;
        for (int i = 0; i < oBoard.ColumnCount(); i++)
            Console.WriteLine(oBoard.ColumnInfo(i).Caption);
    }
}
```

컴포넌트 스트리밍과 해제 알림 없음

- .NET에는 DFM이 없습니다. 컨트롤은 코드 또는 XAML에서 만들고, 데이터, 열, 옵션은 코드에서 설정합니다.
- .NET에는 해제 알림이 없습니다. 뷰는 **Data**를 null로 설정하거나 컨트롤이 해제될 때까지 데이터를 유지합니다. WPF와 Avalonia 컨트롤과 뷰 모델은 `IDisposable`입니다. **Dispose**는 타이머를 멈추고 데이터 리스너를 분리합니다. Avalonia는 컨트롤을 해제하지 않으므로 창이 닫힐 때 직접 해제하십시오.
- 데이터에는 로딩 상태가 없습니다. 변경 사항은 **BeginUpdate**와 **EndUpdate** 사이에 모였다가 한 번만 알려집니다.

호스트

- WinForms는 기본 스크롤 막대와 WinForms의 **BorderStyle**을 사용하고, WPF와 Avalonia는 `ScrollBar` 자식 요소를 가지며 BorderStyle이 없습니다.
- WPF와 Avalonia에는 **TabStop** 대신 **IsTabStop**이 있습니다. Avalonia에서는 Command 키가 Ctrl로 취급되며, **Theme**은 Avalonia의 **StyledElement.Theme**을 가립니다.
- 컨트롤의 속성은 모든 호스트에서 종속성 속성이나 스타일 속성이 아닌 일반 CLR 속성입니다.
- 터치는 마우스 입력으로 전달되며, Delphi의 터치 끌기에는 별도의 코드 경로가 없습니다.
- 분할 커서(**pcrSplit**)는 분할 커서가 없는 WPF와 Avalonia에서 가로 크기 조정 커서입니다.

데이터 소스와 파일

- **TsgcPlannerSource_DataSet**은 TDataSet 대신 `System.Data.DataTable` 테이블에 바인딩됩니다. 차이점은 .NET 데이터 바인딩을 참조하십시오.
- MS Project XML은 `System.Xml`로 읽습니다. Delphi의 sgcPlanner_XML 유닛은 이식되지 않았으며 필요하지도 않습니다.
- 뷰 모델의 **ExportImage**는 SVG만 기록합니다. PNG에는 esegece.sgcPlanner.Rendering이 필요합니다. .NET 헤드리스 렌더링을 참조하십시오.

관련 항목

- .NET 개요
- .NET 클래스
- .NET 데이터와 일정 계산
- .NET 데이터 바인딩

.NET 데이터와 일정 계산

TsgcPlannerData는 계획의 유일한 기준입니다. 항목(작업, 이벤트, 마일스톤), 리소스, 상태, 종속성, 근무 달력을 보관합니다. 뷰는 항목 필드를 직접 쓰지 않습니다. 모든 드롭은 **BeginUpdate**와 **EndUpdate** 안에서 데이터를 호출하고, 데이터는 모든 리스너에 한 번만 알리므로, 같은 데이터에 바인딩된 Gantt, 보드, Scheduler는 항상 일치합니다.

계획 만들기

이 예제는 상태 3개, 사람 2명, 근무조 2개와 휴일이 있는 달력, 완료-시작으로 연결된 작업 4개와 마일스톤 하나로 작은 프로젝트를 만든 다음, 변경 사항을 수신하고, Kanban 드롭처럼 카드를 이동하고, 일정 계산 엔진을 실행합니다.

```
using System;
using esegece.sgcPlanner;

public static class DataModelSample
{
    // a small project: three statuses, two people, a working calendar and four linked tasks
    public static TsgcPlannerData CreateProject()
    {
        TsgcPlannerData oData = new TsgcPlannerData();
        // one notification for the whole set-up
        oData.BeginUpdate();
        try
        {
            oData.Statuses.AddStatus("todo", "To do", 0xFF94A3B8);
            oData.Statuses.AddStatus("doing", "In progress", 0xFF3B82F6);
            oData.Statuses.AddStatus("done", "Done", 0xFF22C55E);
            oData.Resources.AddResource("ann", "Ann Lee").Color = 0xFF0EA5E9;
            oData.Resources.AddResource("bob", "Bob Stone").Color = 0xFFF97316;

            // Monday to Friday, two shifts, one holiday
            oData.Calendar.WorkDays = TsgcPlannerCalendar.CS_DEFAULT_WORKDAYS;
            oData.Calendar.WorkTime = "09:00-13:00,14:00-18:00";
            oData.Calendar.Holidays.AddHoliday(new DateTime(2026, 10, 12), "Public holiday");
            oData.ProjectStart = new DateTime(2026, 10, 5, 9, 0, 0);

            TsgcPlannerItem oItem = oData.AddItem("design");
            oItem.Title = "Design";
            oItem.Start = new DateTime(2026, 10, 5, 9, 0, 0);
            oItem.Finish = new DateTime(2026, 10, 7, 18, 0, 0);
            oItem.Status = "doing";
            oItem.Progress = 40;
            oItem.Resources = "ann";

            oItem = oData.AddItem("build");
            oItem.Title = "Build";
            oItem.Start = new DateTime(2026, 10, 8, 9, 0, 0);
            oItem.Finish = new DateTime(2026, 10, 14, 18, 0, 0);
            oItem.Status = "todo";
            oItem.Resources = "ann,bob";
            oItem.Tags = "dev";

            oItem = oData.AddItem("test");
            oItem.Title = "Test";
            oItem.Start = new DateTime(2026, 10, 15, 9, 0, 0);
            oItem.Finish = new DateTime(2026, 10, 16, 18, 0, 0);
            oItem.Status = "todo";
            oItem.Resources = "bob";

            // zero length: a milestone
            oItem = oData.AddItem("release");
            oItem.Title = "Release";
```

```
            oItem.Start = new DateTime(2026, 10, 19, 9, 0, 0);
            oItem.Finish = oItem.Start;
            oItem.Status = "todo";

            oData.Dependencies.AddDependency("design", "build");
            oData.Dependencies.AddDependency("build", "test");
            // finish to start with a lag of one working hour
            oData.Dependencies.AddDependency("test", "release", TsgcPlannerDependencyKind.pdkFS, 60);
        }
        finally
        {
            oData.EndUpdate();
        }
        return oData;
    }

    // every view bound to the data hears the same notifications
    public static void Watch(TsgcPlannerData aData, Action<string> aLog)
    {
        aData.AddListener((object Sender, TsgcPlannerChange aChange) =>
        {
            if ((aChange.Kinds & TsgcPlannerChangeKinds.pckItems) != 0)
                aLog("items changed: " + aChange.ItemId + " version " + aChange.Version);
        });
    }

    // a card moved to another column, the same call a Kanban drop makes
    public static void Finish(TsgcPlannerData aData, string aId)
    {
        aData.MoveItem(aId, "done", 0);
        aData.FindItem(aId).Progress = 100;
    }

    // place the tasks on the working calendar after their links (Registered edition)
    public static bool Schedule(TsgcPlannerData aData)
    {
#if !SGC_COMMUNITY
        TsgcPlannerScheduleEngine oEngine = new TsgcPlannerScheduleEngine();
        oEngine.OnScheduleError = (Sender, aError, aCycleIds) => Console.WriteLine(aError);
        return oEngine.Schedule(aData);
#else
        return false;
#endif
    }
}
```

참고

- **Finish**는 종일 항목에서도 배타적입니다. 10월 1일의 종일 항목은 10월 1일 00:00부터 10월 2일 00:00까지입니다. **Start**와 **Finish**가 같은 항목은 마일스톤입니다.
- **Resources**와 **Tags**는 쉼표로 구분한 목록이며, **ResourceAt**, **ResourceCount**, **TagAt**, **TagCount**로 하나씩 읽습니다. **ParentId**는 작업 트리를 만들고 **UpdateSummaries**는 요약 작업을 다시 계산합니다.
- **MoveItem**은 카드의 상태와 순위를 바꾸고, **DeleteItem**은 항목과 그 시리즈의 예외 항목을 삭제하며, **SelectedId**는 뷰가 공유하는 선택입니다.
- **OnItemChanging**은 **TsgcPlannerItemFields** 집합에 있는 필드에 대해 편집이 이루어지기 전에 거부할 수 있습니다.
- 엔진은 **Manual**이 아닌 모든 작업을 링크와 **Constraint**에 따라 근무 달력 위에서 가능한 한 빨리 배치하고 **TotalSlack**과 **Critical**을 계산합니다. 링크가 순환하면 순환에 속한 id와 함께 **OnScheduleError**가 발생합니다. **AutoSchedule**이 켜진 Gantt는 편집할 때마다 엔진을 실행하고, Gantt의 **Schedule**은 필요할 때 실행하며, **SetBaseline**은 현재 날짜를 기준선으로 저장합니다. 엔진은 Registered 기능입니다.

관련 항목

- .NET 개요
- .NET 클래스
- .NET JSON, iCalendar, MS Project
- 공유 데이터
- 자동 일정 계산
- 근무 달력

.NET 데이터 바인딩

Delphi의 **TsgcPlannerSource_DataSet**은 TDataSet 테이블에서 계획을 불러오고 편집 내용을 다시 기록합니다. .NET에서는 같은 클래스가 `System.Data.DataTable` 테이블, 즉 모든 ADO.NET 공급자의 `DataAdapter`가 채우는 테이블에 바인딩됩니다. 뷰는 **TsgcPlannerData**에 바인딩된 상태로 남으므로, 같은 소스가 Gantt, 보드, Scheduler를 한꺼번에 처리합니다.

속성	설명
Data	소스가 채우는 **TsgcPlannerData**입니다.
DataSet, ResourceDataSet, DependencyDataSet	항목, 리소스, 링크의 테이블입니다. 항목 테이블만 필수입니다.
ItemFields, ResourceFields, DependencyFields	열 이름입니다. 기본값은 항목(Id, Title, Start, Finish, AllDay, Status, Progress, Color, ParentId, Tags, Resources, Rank, Notes, RecurrenceRule, DependsOn), 리소스(Id, Name, Color, Initials), 링크(FromId, ToId, Kind, Lag)의 속성 이름입니다. 존재하지 않는 열은 매핑되지 않습니다.
Active	테이블을 데이터로 불러오고, true인 동안 동기화 상태를 유지합니다.
LiveUpdates	테이블에서 추가, 변경, 삭제된 행이 즉시 데이터에 반영됩니다(기본값 true).
WriteBack	뷰나 코드에서 한 편집이 행에 다시 기록됩니다(기본값 true).
OnLoadItem	불러온 항목마다 테이블과 `DataRow` 와 함께 발생하며, 맵이 다루지 않는 필드를 채우는 데 씁니다.

예제

```
using System;
using System.Data;
using esegece.sgcPlanner;

public static class DataBindingSample
{
    // a DataTable with the default column names of the source
    public static DataTable CreateTasks()
    {
        DataTable oTable = new DataTable("Tasks");
        oTable.Columns.Add("Id", typeof(string));
        oTable.Columns.Add("Title", typeof(string));
        oTable.Columns.Add("Start", typeof(DateTime));
        oTable.Columns.Add("Finish", typeof(DateTime));
        oTable.Columns.Add("Status", typeof(string));
        oTable.Columns.Add("Progress", typeof(double));
        oTable.Rows.Add("t1", "Kick-off", new DateTime(2026, 10, 5, 9, 0, 0), new DateTime(2026, 10, 5, 12, 0, 0), "done",
        oTable.Rows.Add("t2", "Prototype", new DateTime(2026, 10, 6, 9, 0, 0), new DateTime(2026, 10, 9, 18, 0, 0), "doing"
        return oTable;
    }

    public static TsgcPlannerSource_DataSet Bind(TsgcPlannerData aData, DataTable aTasks)
    {
        TsgcPlannerSource_DataSet oSource = new TsgcPlannerSource_DataSet();
        oSource.Data = aData;
        oSource.DataSet = aTasks;
        // a column with another name is mapped by its field property
        oSource.ItemFields.TitleField = "Title";
        // rows added, changed or deleted in the table reach the data at once
```

```
        oSource.LiveUpdates = true;
        // and edits made in the views are written back to the rows
        oSource.WriteBack = true;
        oSource.OnLoadItem = (Sender, aDataSet, aRow, aItem) =>
        {
            if (aItem.Status == "done")
                aItem.Color = 0xFF22C55E;
        };
        oSource.Active = true;
        return oSource;
    }
}
```

TDataSet과의 차이

- `DataTable`에는 커서가 없습니다. Delphi의 Edit와 Post는 `DataRow.BeginEdit`와 `EndEdit` 또는 열에 직접 값을 대입하는 것에 해당하고, Cancel은 `CancelEdit`입니다. 편집 중인 행은 편집이 끝날 때 기록됩니다.
- 소스는 테이블 이벤트를 수신합니다. 추가되거나 변경된 행은 게시된 레코드로 처리되고, 삭제된 행이나 비워진 테이블은 항목을 제거하며, 거부된 변경은 레코드를 다시 불러오고, `AcceptChanges`는 무시됩니다. 삭제된 행과 분리된 행은 건너뜁니다.
- 쓸 수 있는 열은 ReadOnly, AutoIncrement, 계산 열이 아닌 열입니다. 숫자 날짜 열에는 OLE Automation 날짜가 들어가며, 텍스트 열이 아닌 열에 빈 텍스트를 대입하면 `DBNull`이 설정됩니다.
- 현재 레코드가 없으므로 **OnLoadItem**은 `DataRow`를 전달합니다. 열 이름은 이름이 붙은 속성 외에도 인덱스로 접근하는 **GetName**과 **SetName**이 있습니다.
- `DataTable` 개체만 바인딩하십시오. `DataView`나 `IDataReader`는 소스가 될 수 없습니다. 해제된 테이블은 스스로 바인딩을 해제하며, 소스는 `IDisposable`입니다.

컨트롤 바인딩

컨트롤의 속성은 종속성 속성이나 스타일 속성이 아니라 일반 CLR 속성입니다. 계획은 하나의 값이 아니라 전체 모델이기 때문입니다. MVVM 애플리케이션에서는 **TsgcPlannerData**를 뷰 모델에 두고 컨트롤의 **Data**에 대입한 다음, **OnChange**나 데이터의 리스너, **SelectedId**, 컨트롤의 이벤트에 반응하십시오.

관련 항목

- .NET 개요
- .NET 데이터와 일정 계산
- Delphi와 .NET의 차이
- TsgcPlannerSource_DataSet
- DataSet 바인딩

.NET JSON, iCalendar, MS Project

계획은 세 가지 형식으로 저장하고 불러오며, 결과는 Delphi와 같습니다. .NET 애플리케이션이 기록한 파일을 Delphi 애플리케이션이 읽을 수 있고 그 반대도 가능합니다.

형식	클래스	에디션
JSON, 모든 필드를 담은 기본 형식	**TsgcPlannerData**의 **SaveToJSON**, **LoadFromJSON**, **SaveToFile**, **LoadFromFile**과 **sgcPlanner_JSON**의 스트림 및 파일 함수입니다.	둘 다
iCalendar(RFC 5545), Outlook, Google 캘린더, Apple 캘린더용	**TsgcPlannerICalendar**	Registered
Microsoft Project XML(MSPDI): 작업, 링크, 리소스, 할당, 달력	**TsgcPlannerMSProject**	Registered

예제

```
using System;
using System.IO;
using esegece.sgcPlanner;

public static class InterchangeSample
{
    // JSON: the native format, in every edition
    public static TsgcPlannerData CopyByJSON(TsgcPlannerData aData, string aFileName)
    {
        aData.SaveToFile(aFileName);
        TsgcPlannerData oCopy = new TsgcPlannerData();
        oCopy.LoadFromFile(aFileName);
        // or as a string, or on a stream with sgcPlanner_JSON
        string vText = aData.SaveToJSON();
        using (MemoryStream oStream = new MemoryStream())
            sgcPlanner_JSON.sgcPlannerJSONSaveToStream(aData, oStream);
        oCopy.LoadFromJSON(vText);
        return oCopy;
    }

#if !SGC_COMMUNITY
    // iCalendar (RFC 5545): Outlook, Google Calendar, Apple Calendar
    public static void ExchangeICal(TsgcPlannerData aData, string aFileName)
    {
        TsgcPlannerICalendar oICal = new TsgcPlannerICalendar();
        oICal.Data = aData;
        oICal.ExportUTC = true;
        oICal.ExportToFile(aFileName);

        // merge the events of another calendar into the data
        oICal.ImportMode = TsgcPlannerICalImportMode.pimMerge;
        oICal.OnImportWarning = (Sender, aLine, aMessage) =>
            Console.WriteLine("line " + aLine + ": " + aMessage);
        oICal.ImportFromFile(aFileName);
    }

    // Microsoft Project XML (MSPDI): tasks, links, resources, assignments and the calendar
    public static TsgcPlannerData ExchangeMSProject(TsgcPlannerData aData)
    {
```

```
        TsgcPlannerMSProject oProject = new TsgcPlannerMSProject();
        oProject.Data = aData;
        string vXml = oProject.ExportToString();

        TsgcPlannerData oCopy = new TsgcPlannerData();
        oProject.Data = oCopy;
        oProject.ImportMode = TsgcPlannerMSProjectImportMode.pmiReplace;
        oProject.ImportFromString(vXml);
        if (oProject.WarningCount > 0)
            Console.WriteLine(oProject.WarningCount + " warnings");
        return oCopy;
    }
#endif
}
```

참고

- JSON은 바이트 순서 표시 없는 UTF-8로 기록되며, 불러올 때는 표시가 있어도 허용됩니다. 유효한 UTF-8이 아닌 텍스트는 **EsgcPlannerError**를 발생시킵니다.
- **TsgcPlannerICalendar**와 **TsgcPlannerMSProject**는 일반 클래스입니다. 만든 다음 **Data**를 설정하고 **ImportFromString**, **ImportFromStream**, **ImportFromFile**이나 **ExportToString**, **ExportToStream**, **ExportToFile**을 호출하십시오. **ImportMode**는 데이터를 교체하거나 가져온 항목을 병합합니다.
- 입력에 문제가 있어도 가져오기는 중단되지 않습니다. 줄 번호와 메시지와 함께 **OnImportWarning**이 발생하고 **WarningCount**가 그 수를 셉니다. 메시지는 클래스의 공개 상수이며 `{0}` 자리 표시자가 있습니다.
- **ExportUTC**는 iCalendar 시간을 UTC로 기록합니다. **ProductId**는 내보내기의 PRODID입니다.
- MS Project XML은 DTD를 무시하고 외부 확인자 없이 `System.Xml.XmlReader`로 읽으며, Delphi 작성기를 이식한 코드로 기록하므로 내보낸 결과가 Delphi와 바이트 단위로 같습니다.

관련 항목

- .NET 개요
- .NET 데이터와 일정 계산
- JSON 저장
- iCalendar
- MS Project XML

.NET 헤드리스 렌더링

모든 뷰는 자신의 뷰 모델을 통해 그려지며, 모델에는 창이 필요 없습니다. 서비스, 테스트, 보고서는 계획을 불러와 Scheduler, Gantt, 보드를 Linux를 포함한 어떤 플랫폼에서든 PNG나 SVG로 저장할 수 있습니다.

- esegece.sgcPlanner.Rendering의 **TsgcPlannerRenderer**가 세 가지 뷰를 소유합니다. **Data**, **View**(**prvScheduler**, **prvGantt**, **prvKanban**), **Width**, **Height**, **Scale**, **Theme**, 뷰의 옵션을 설정한 다음 **SaveToPNG** 또는 **SaveToSVG**를 호출하십시오.
- **SchedulerOptions**, **GanttOptions**, **KanbanOptions**에는 컨트롤과 같은 속성이 있습니다. **Assign**은 뷰 모델에서 이를 복사하므로, 보고서가 컨트롤이 보여 주는 내용을 그대로 그릴 수 있습니다. 옵션이 다루지 않는 내용은 **SchedulerView**, **GanttView**, **KanbanView**에서 설정합니다.
- 뷰 모델만 따로 그릴 수도 있습니다. **ExportImage**는 esegece.sgcPlanner로 SVG를 기록하고, **sgcPlanner_Raster**의 확장 메서드 **SaveToPNG**, **ExportPNG**, **RenderToRaster**는 PNG를 기록합니다.

예제

```
using System;
using esegece.sgcPlanner;
using esegece.sgcUICore;

public static class HeadlessSample
{
    public static void Run(TsgcPlannerData aData)
    {
        using (TsgcPlannerRenderer oRenderer = new TsgcPlannerRenderer())
        {
            oRenderer.Data = aData;
            oRenderer.Theme = TsgcUITheme.utLight;
            oRenderer.Width = 1200;
            oRenderer.Height = 400;

            // the Gantt with a tree of three columns
            oRenderer.View = TsgcPlannerRendererViewKind.prvGantt;
            oRenderer.GanttOptions.Columns.Clear();
            oRenderer.GanttOptions.Columns.AddColumn(TsgcPlannerGanttField.pgcTitle, 180);
            oRenderer.GanttOptions.Columns.AddColumn(TsgcPlannerGanttField.pgcResources, 120);
            oRenderer.GanttOptions.Columns.AddColumn(TsgcPlannerGanttField.pgcDuration, 70, TAlignment.taRightJustify);
            oRenderer.GanttOptions.TreeWidth = 370;
            oRenderer.GanttOptions.BarColorMode = TsgcPlannerBarColorMode.pbcStatus;
            oRenderer.SaveToPNG("gantt.png");
            oRenderer.SaveToSVG("gantt.svg");

            // the board, one column per status
            oRenderer.View = TsgcPlannerRendererViewKind.prvKanban;
            oRenderer.Height = 500;
            oRenderer.SaveToPNG("board.png");

            // the work week of 5 October 2026 at twice the size
            oRenderer.View = TsgcPlannerRendererViewKind.prvScheduler;
            oRenderer.Date = new DateTime(2026, 10, 5);
            oRenderer.SchedulerOptions.View = TsgcPlannerSchedulerViewKind.psvWorkWeek;
            oRenderer.SchedulerOptions.Options.DayStartHour = 8;
            oRenderer.SchedulerOptions.Options.DayEndHour = 19;
            oRenderer.Scale = 2;
            oRenderer.SaveToPNG("week@2x.png");
        }
    }
}
```

참고

- 크기는 논리 픽셀 단위입니다. 1200 x 500 뷰에서 배율 2는 2400 x 1000 이미지를 기록합니다. **DeviceWidth**와 **DeviceHeight**는 이미지의 크기를 반환합니다.
- **FontFile**과 **FontFileBold**는 기본 글꼴이 없는 서버를 위해 TrueType 글꼴 파일을 선택합니다.
- 헤더 레이블은 공유 **TsgcPlannerLocale**(`sgcPlanner_DateUtils.sgcPlannerLocale()`)을 따르며, 처음 사용할 때 현재 문화권에서 가져옵니다. 모든 컴퓨터에서 같은 출력을 얻으려면 첫 렌더링 전에 문화권을 설정하거나 해당 로캘의 이름을 설정하십시오.
- 컨트롤에도 컨트롤 크기로 동작하는 **SaveToPNG**와 **SaveToSVG**가 있습니다.

관련 항목

- .NET 개요
- .NET 클래스
- TsgcPlannerRenderer
- 헤드리스 렌더링

.NET의 Google 캘린더 동기화와 iCalendar

패 키 지: esegece.sgcPlanner.sgcWebSockets(Registered 에 디 션), 네 임 스 페 이 스 esegece.sgcPlanner.netstandard2.0, net8.0, net9.0 을 대 상 으 로 하 며 HTTP 와 Google OAuth2 에 sgcWebSockets .NET 라 이 브 러 리 (esegece.sgcWebSockets)를 사용합니다.
.NET 패키지는 Delphi 브리지와 같은 두 구성 요소를 가지며 이름, 매핑, 충돌 규칙, 상태 파일 형식이 같습니다. 따라서 Delphi가 기록한 상태 파일을 .NET에서 불러올 수 있고 그 반대도 가능합니다.

- **TsgcPlannerGoogleSync**는 TsgcPlannerData를 TsgcHTTP_Google_Cloud_Calendar_API_Client(Client 속성)를 통해 Google 캘린더와 양방향으로 동기화된 상태로 유지합니다. TsgcPlannerGoogleSync를 참조하십시오.
- **TsgcPlannerICalSubscription**은 읽기 전용 iCalendar URL(https://, http:// 또는 webcal://)을 데이터에 병합하며, TsgcPlannerICalendar로 구문 분석합니다. TsgcPlannerICalSubscription을 참조하십시오.

Delphi와의 차이

- 두 클래스 모두 IDisposable을 구현하는 일반 클래스이며 구성 요소가 아닙니다. Dispose는 작업자 스레드를 중지하며 그 이후에는 아무것도 적용되지 않습니다.
- 이벤트는 델리게이트 속성입니다. 예를 들어 OnRefreshed, OnError, OnImportWarning, OnBeforeRefresh(EventHandler)가 있습니다. OnConflict와 OnFilterItem은 결과를 ref로 받습니다.
- SyncAsync는 Task<bool>을 반환합니다. RefreshAsync는 bool을 반환하며 새로 고침이 실행 중이면 False입니다.
- Data를 건드리고 이벤트를 발생시키는 작업은 SyncAsync나 RefreshAsync를 호출했거나 Active를 설정한 스레드의 SynchronizationContext에서 실행됩니다(WinForms, WPF, Avalonia에서는 UI 스레드). 컨텍스트가 없으면 작업자 스레드에서 실행됩니다.
- LastSyncTime과 LastRefreshTime은 DateTime이며, default(DateTime)은 한 번도 없음을 의미합니다.
- TsgcPlannerICalSubscription의 전송 연결 지점은 HTTP 상태를 반환하는 protected 가상 메서드 DoFetch(string aURL, IList<string> aRequestHeaders, IList<string> aResponseHeaders, Stream aBody)입니다. 기본 구현은 HandleRedirects를 끈 내부 HTTPClient(TsgcHTTP1Client)를 사용하며, 구성 요소는 최대 5번의 리디렉션을 직접 따라가고 마지막 가져오기의 검증 값으로 If-None-Match와 If-Modified-Since를 보냅니다.
- Free 알림은 없습니다. Data를 소멸시켜도 Data 속성은 지워지지 않습니다.

읽기 전용 피드 이벤트

새로 고침은 구독 항목에 대한 로컬 편집을 덮어씁니다. 뷰가 끌기, 크기 조정, 편집을 확정하기 전에 묻는 OnItemChanging에서 이를 거부하십시오.

```
TsgcPlannerData oData = new TsgcPlannerData();
TsgcPlannerICalSubscription oSubscription = new TsgcPlannerICalSubscription();
oSubscription.Data = oData;
oSubscription.URL = "webcal://calendar.example.com/team.ics";
oSubscription.RefreshInterval = 60; // minutes
oSubscription.OnRefreshed = (s, aAdded, aUpdated, aDeleted) =>
  Console.WriteLine("added " + aAdded + ", updated " + aUpdated + ", deleted " + aDeleted);
oSubscription.OnError = (s, aError) => Console.WriteLine(aError);
oData.OnItemChanging = (object s, TsgcPlannerItem aItem, TsgcPlannerItemFields aFields, ref bool aAccept) =>
{
  if (oSubscription.IsSubscriptionItem(aItem))
    aAccept = false;
};
oSubscription.Active = true; // refreshes at once, then every hour
```

Google 캘린더 동기화

```
TsgcHTTP_Google_Cloud_Calendar_API_Client oClient = new TsgcHTTP_Google_Cloud_Calendar_API_Client();
oClient.GoogleCloudOptions.OAuth2.ClientId = "your client id";
oClient.GoogleCloudOptions.OAuth2.ClientSecret = "your client secret";
oClient.GoogleCloudOptions.OAuth2.LocalPort = 8080;
oClient.GoogleCloudOptions.OAuth2.Scope.Add("https://www.googleapis.com/auth/calendar");

TsgcPlannerGoogleSync oSync = new TsgcPlannerGoogleSync();
```

```
oSync.Client = oClient;
oSync.Data = oData;
oSync.CalendarId = "primary";
oSync.StateFileName = "calendar_state.json";
oSync.OnSyncDone = (s, aPulled, aPushed, aDeleted, aConflicts) =>
  Console.WriteLine("pulled " + aPulled + ", pushed " + aPushed);
oSync.Active = true;
bool vOK = await oSync.SyncAsync();
```

Id가 ExcludeIdPrefix(기본값 ics:, TsgcPlannerICalSubscription의 IdPrefix)로 시작하는 항목은 Google로 전송되지 않습니다.

데모

WinForms 데모 demos\Planner\GoogleCalendarSync는 TsgcWinFormsScheduler에서 두 구성 요소를 보여 줍니다. --test로 실행하면 오프라인 자체 테스트를 수행합니다. 테스트용 Google 전송과 루프백 iCalendar 서버를 사용하므로 자격 증명이 필요하지 않습니다.

관련 항목

- TsgcPlannerGoogleSync
- TsgcPlannerICalSubscription
- Google 캘린더 동기화
- TsgcPlannerICalendar
- TsgcPlannerData
- iCalendar

.NET 데모

데모는 `.net\demos\Planner`에 있으며, 데모마다 프로젝트, `Program.cs`, README.md가 들어 있는 폴더가 하나씩 있습니다. 사용자 인터페이스는 XAML 없이 코드로 만듭니다. 모든 데모에는 화면 밖에서 자체 점검을 실행하고 PASS와 FAIL 줄을 기록하며 실패한 점검의 수를 종료 코드로 돌려주는 `--test` 스위치가 있습니다. Delphi 데모의 `/test` 스위치와 같습니다. 창이 있는 데모에는 뷰를 PNG 파일로 그리는 `--snapshot` 스위치도 있습니다.

데모	호스트	내용
ProjectPlanner	WinForms	웹사이트 재출시 프로젝트를 세 가지 방식으로 동시에 보여 줍니다. Gantt 차트, Kanban 보드, 주 보기가 모두 같은 데이터에 바인딩됩니다. 두 개의 트랙에 작업 14개, 사람 5명, 상태 4개, 완료-시작 링크가 있습니다. 보드에서 카드를 끌면 Gantt와 Scheduler가 따라갑니다.
ProjectPlannerWpf	WPF	**TsgcWpf** 컨트롤을 사용한 같은 창입니다.
ProjectPlannerAvalonia	Avalonia	**TsgcAvalonia** 컨트롤을 사용한 같은 창이며, Windows, Linux, macOS에서 동작합니다.
TeamScheduler	WinForms	5명의 팀 회의 3주치입니다. 일, 주중, 주, 월, 안건 목록, 리소스 Timeline, 반복 시리즈인 일일 스탠드업과 주간 동기화, 종일 컨퍼런스, **OnItemCreate**를 이용한 끌어서 만들기, 리소스별 그룹화를 보여 줍니다.
TeamSchedulerWpf	WPF	**TsgcWpfScheduler**를 사용한 같은 창입니다.
TeamSchedulerAvalonia	Avalonia	**TsgcAvaloniaScheduler**를 사용한 같은 창입니다.
HeadlessRender	콘솔	창 없이 동작합니다. JSON에서 프로젝트를 불러오거나(또는 만들어 project.json으로 저장하고) **TsgcPlannerRenderer**로 Scheduler, Gantt, 보드의 PNG와 SVG를 기록하며, Windows, Linux, macOS에서 동작합니다. Delphi HeadlessReport 데모의 .NET 대응 데모입니다.
GoogleCalendarSync	WinForms	Google 캘린더와 양방향으로 동기화되는 주 보기에 읽기 전용 iCalendar 구독이 더해진 데모로, esegece.sgcPlanner.sgcWebSockets 브리지의 컴포넌트를 사용합니다(Registered).

ProjectPlanner 와 TeamScheduler 의 UI 와 무관한 부분(생성된 데이터, 뷰 설정, 스냅샷, 자체 점검)은 `ProjectPlanner_Shared.cs`와 `TeamScheduler_Shared.cs`에 있으며, WinForms, WPF, Avalonia 프로젝트가 이를 컴파일하므로 세 호스트가 같은 점검을 실행합니다.
모든 데모는 두 에디션에서 빌드되고 실행됩니다. Community 에디션에서 TeamScheduler에는 Timeline 항목이 없고 각 시리즈를 한 번만 표시하며, ProjectPlanner는 종속성 화살표를 그리지 않고, GoogleCalendarSync는 브리지가 Registered 패키지임을 안내합니다.

데모 빌드

제품 트리 안에서는 데모가 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip은 기본적으로 Community 에디션의 NuGet 패키지를 사용합니다. **SgcPlannerEdition**을 설정하면 Registered 패키지로 데모를 빌드할 수 있습니다.

```
dotnet build ProjectPlanner\ProjectPlanner.csproj -p:SgcPlannerEdition=Registered
dotnet run --project HeadlessRender\HeadlessRender.csproj -- --test
```

관련 항목

- .NET 개요
- NuGet으로 설치
- WinForms 빠른 시작
- 데모

문제 해결

EsgcPlannerError: This feature requires the Registered edition of sgcPlanner

Community 빌드에서 Gantt의 Schedule과 SetBaseline이 이를 발생시킵니다. 이 호출을 `{$IFDEF SGC_PLANNER_ADVANCED}` 안에 두거나 Registered 또는 Trial 에디션을 사용하십시오. 에디션별 기능을 참조하십시오.

Community 빌드에서 폼이 로드되지 않습니다: Class TsgcPlannerICalendar not found

TsgcPlannerICalendar와 TsgcPlannerMSProject는 Community 에디션에 포함되지 않습니다. Community에서도 로드되어야 하는 폼에 놓는 대신 `{$IFDEF SGC_PLANNER_ADVANCED}` 안에서 코드로 생성하십시오.

Scheduler에 이벤트가 표시되지 않습니다

- 이벤트가 화면의 기간 안에 있는지 확인하십시오. Date 0은 시계를 따르므로 뷰는 현재 주를 보여 줍니다. Date를 설정하거나 GoToday, Next, Prior를 호출하십시오.
- 일 뷰는 Options.DayStartHour부터 Options.DayEndHour까지의 시간을 보여 줍니다. 그 밖의 이벤트는 가장자리에서 잘려 그려집니다.
- Finish가 Start보다 앞인 이벤트는 Start에서 길이가 0인 이벤트로 읽힙니다.
- Scheduler의 Data 속성이 코드가 채우는 데이터 객체인지 확인하십시오. Data가 없는 컨트롤은 자체 내부 데이터 객체를 사용합니다.

반복 이벤트가 한 번만 나타납니다

반복은 Registered 에디션에 포함됩니다. Registered에서는 규칙이 지원되지 않을 때 이유를 반환하는 TsgcPlannerRecurrence.TryParse로 규칙을 확인하고, COUNT, UNTIL, RecurrenceExceptions를 확인하십시오. 반복을 참조하십시오.

자동 일정 계산이 작업을 이동하지 않습니다

- 엔진은 Manual 작업, 반복 항목, 종속성 순환에 속한 작업의 날짜를 유지합니다. OnScheduleError는 순환에 속한 작업을 나열하며 Schedule은 False를 반환합니다.
- Constraint가 pcSNET인 작업은 ConstraintDate보다 일찍 시작하지 않습니다. Gantt에서의 이동은 이 제약을 설정합니다. 엔진이 작업을 더 일찍 당기도록 하려면 Constraint를 다시 pcASAP로 설정하십시오.
- AutoSchedule은 Gantt에서 한 편집 뒤에만 실행됩니다. 코드나 다른 뷰에서 변경한 뒤에는 Schedule을 호출하십시오.

자동 일정 계산을 참조하십시오.

Gantt가 종속성 화살표를 그리지 않습니다

화살표에는 Registered 에디션과 ShowDependencies True가 필요합니다. 링크는 두 작업이 모두 보이는 행에 있을 때만 그려지므로 접힌 요약으로 들어가는 링크는 숨겨지고, 알 수 없는 FromId 또는 ToId를 가진 링크는 그려지지 않습니다.

끌기가 아무 일도 하지 않거나 항목이 되돌아갑니다

컨트롤의 ReadOnly와 DragOptions, 그리고 편집을 거부할 수 있는 핸들러를 확인하십시오: 데이터의 OnItemChanging, OnCardMoving, OnItemCreate, OnLinkCreating, OnRecurrenceEdit. 보드에서 WIPMode pwmBlock이고 WIP 제한에

도달한 열은 새 카드를 거부하며, 알 수 없는 열은 놓을 대상이 아닙니다. Scheduler에서 요약 항목은 표시되지만 끌 수 없습니다. 끌어서 놓기를 참조하십시오.

EsgcPlannerError: OnChange kept changing the planner data for more than 8 rounds

OnChange 핸들러나 리스너가 알림마다 데이터를 변경하므로 변경이 가라앉지 않습니다. 데이터가 원하는 값과 다를 때만 변경하거나, 변경을 핸들러 밖으로 옮기십시오.

iCalendar 가져오기 또는 내보내기 후 시간이 몇 시간 어긋납니다

UTC 또는 시간대가 있는 시간은 가져올 때 컴퓨터의 로컬 시간으로 변환됩니다. 내보낼 때 ExportUTC는 반복되지 않는 항목의 시간을 UTC로 기록하며, 달력은 이를 자신의 시간대로 표시합니다. 로컬 시간으로 기록하려면 ExportUTC를 False로 설정하십시오. iCalendar를 참조하십시오.

날짜, 기간 또는 이름이 다른 컴퓨터나 Linux에서 다르게 보입니다

레이블과 기간 숫자는 컴퓨터에서 초기화되는 sgcPlannerLocale을 사용합니다. 어디서나 같은 출력을 얻으려면 시작할 때 속성(예: ShortDateFormat, Use24Hour, DecimalSeparator)을 설정하십시오.

Delphi 7 또는 Delphi 2007에서 문자가 손실됩니다

이 버전들은 Ansi 문자열을 사용하므로 JSON, iCalendar, MS Project XML 파일의 UTF-8 텍스트는 시스템 코드 페이지로 변환되며 그 밖의 문자는 손실됩니다. Delphi 2009 이상은 모든 문자를 유지합니다.

편집이 데이터셋에 되써지지 않습니다

- WriteBack이 True인지, 그리고 IdField가 Locate가 찾을 수 있는 필드를 지정하는지 확인하십시오.
- 데이터셋이 편집 또는 삽입 상태인 동안에는 아무것도 기록되지 않으며, 변경은 Post 또는 Cancel 뒤에 기록됩니다.
- 리소스, 계산 필드, 조회 필드, 읽기 전용 필드는 기록되지 않습니다.

DataSet 바인딩을 참조하십시오.

LoadFromJSON이 EsgcPlannerError를 발생시킵니다

텍스트가 올바른 JSON이 아니거나, sgcPlanner 문서가 아니거나, 더 새로운 형식 버전으로 기록되었습니다. 데이터는 그대로 유지되므로 애플리케이션이 메시지를 보고하고 계속 실행할 수 있습니다. JSON 저장을 참조하십시오.

폼이나 객체가 해제된 뒤 액세스 위반이 발생합니다

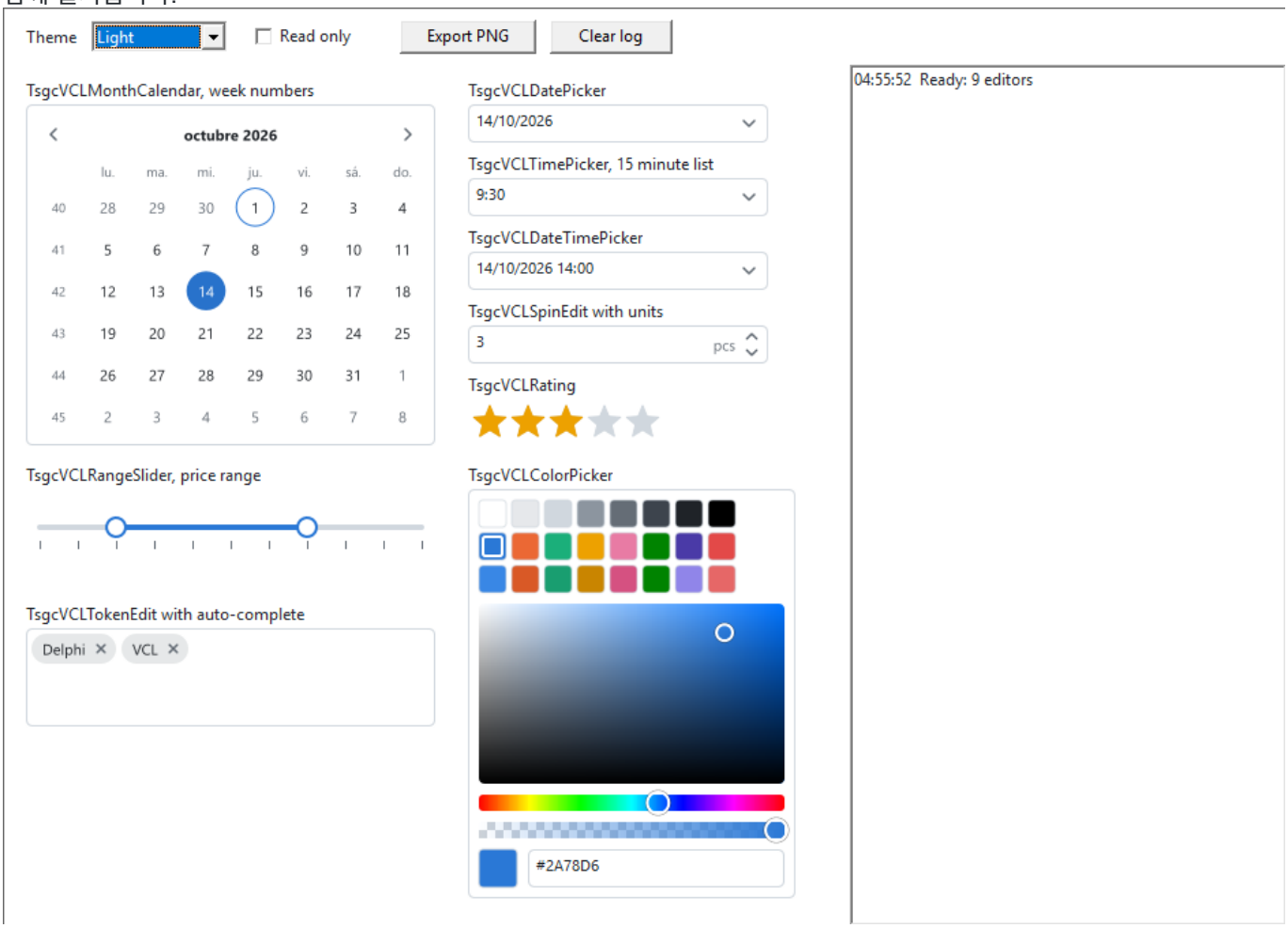
AddListener로 추가한 핸들러는 그것을 소유한 객체가 해제되기 전에 RemoveListener로 제거해야 합니다. 데이터는 그 객체가 언제 사라지는지 알지 못합니다.

관련 항목

- 에디션별 기능
- 플랫폼
- TsgcPlannerData

소개

sgcEditors는 VCL과 FireMonkey용으로 폼의 입력 컨트롤을 Delphi 및 C++Builder 애플리케이션에 추가합니다. 월 달력, 날짜 선택기, 시간 선택기, 날짜 및 시간 선택기, 범위 슬라이더, 색상 선택기, 토큰 편집기, 평점, 스핀 편집기가 포함됩니다. 모든 컨트롤은 VCL용으로는 TsgcVCL 접두사, FireMonkey용으로는 TsgcFMX 접두사로 제공됩니다. 데이터 인식 변형 5종은 날짜 선택기, 시간 선택기, 스핀 편집기, 평점, 범위 슬라이더를 TDataSet의 필드에 바인딩합니다.
텍스트는 항상 네이티브 편집 컨트롤에 입력되며 자체 캐럿에는 입력되지 않으므로, IME 입력, 클립보드, 실행 취소, 화면 읽기 프로그램이 폼의 다른 편집 컨트롤과 똑같이 동작합니다. 그 밖의 모든 것, 즉 프레임, 버튼, 달력, 칩, 별, 핸들, 색상 사각형은 **sgcUICore**가 그리므로, 컨트롤은 VCL, FireMonkey, 헤드리스 PNG 및 SVG 출력에서 모두 같은 모양입니다. 컨트롤에는 라이트 테마와 다크 테마, 강조 색상, 완전한 키보드 지원, 접근성 이름, 모니터별 DPI 배율이 있습니다.
모든 컨트롤의 뒤에는 TsgcCalendarModel이나 TsgcSpinModel 같은 프레임워크 중립 모델 클래스가 있습니다. VCL과 FireMonkey 컨트롤은 입력을 매핑하고 편집 컨트롤과 드롭다운을 호스트할 뿐이므로, 같은 모델은 창 없이 서버나 Linux64에서도 렌더링할 수 있습니다.
sgcEditors는 eSeGeCe 비주얼 제품군의 일부입니다. 캔버스, 래스터라이저, 글꼴, SVG 작성기, 테마를 갖춘 공유 런타임 패키지인 sgcUICore로 그리기 때문에, 에디터는 제품군의 다른 제품과 어울립니다. sgcEditors에는 sgcUICore가 필요하며, 함께 설치됩니다.



에디션

sgcEditors는 Registered, Trial, Community로 제공됩니다. Community는 VCL과 FireMonkey의 컨트롤 9종을 모두 유지하지만, 데이터 인식 컨트롤 5종, 색상 선택기의 HSV 사각형과 알파 막대, 토큰 편집기의 자동 완성 목록, 범위 슬라이더의 두 번째 핸들은 제외합니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 모델, 호스트, 내부 편집 컨트롤, 드롭다운, 두 개의 렌더 레이어.
- 컨트롤: 9종의 컨트롤, 데이터 인식 변형, 모델 클래스를 한눈에 보기.

- 컨트롤 페이지: 월 달력, 날짜 선택기, 시간 선택기, 날짜 및 시간 선택기, 범위 슬라이더, 색상 선택기, 토큰 편집기, 평점, 스핀 편집기, 데이터 인식 컨트롤, 모델 클래스. 속성, 메서드, 이벤트, 예제가 있습니다.
- 설치와 플랫폼: 설치 프로그램, 패키지, Delphi 및 C++Builder 버전, 플랫폼.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder, 빠른 시작 데이터 인식: 몇 분 만에 만드는 첫 폼.
- 텍스트 입력과 IME, 키보드와 접근성, 테마, 현지화, DPI, 헤드리스 출력, 에디션.
- 문제 해결: 흔한 문제와 그 원인.

아키텍처

sgcEditors는 편집 로직을 프레임워크와 분리합니다. 각 컨트롤 계열에는 RTL과 sgcUICore만 사용하는 모델 클래스가 하나 있으며, VCL과 FireMonkey 컨트롤은 그 둘레의 얇은 호스트입니다.

컨트롤의 구성 요소

1. **모델.** TsgcEditorModel의 TPersistent 하위 클래스이며, 예를 들어 TsgcCalendarModel, TsgcDatePickerModel, TsgcTokenModel이 있습니다. 값과 옵션을 담고, KeyDown, KeyPress, MouseDown, MouseMove, MouseUp, MouseWheel을 통해 논리 픽셀로 키와 마우스를 받으며, 임의의 sgcUICore 캔버스에 자신을 그립니다. 창 없이 동작하므로 헤드리스 렌더러이기도 합니다.
2. **호스트.** VCL 컨트롤은 TsgcVCLCustomEditor를, FireMonkey 컨트롤은 TsgcFMXCustomEditor를 상속합니다. 호스트는 모델을 만들고, 입력을 모델로 보내고, 두 레이어를 칠하고, OnChange를 발생시키며, 모델의 속성을 게시합니다. 컨트롤의 모델은 Model로 public이며, 각 컨트롤에는 CalendarModel, PickerModel, SpinModel 같은 형식이 지정된 속성도 있습니다.
3. **내부 편집 컨트롤.** 선택기, 스핀 편집기, 색상 선택기, 토큰 편집기는 네이티브의 테두리 없는 편집 컨트롤을 호스트합니다. VCL에서는 TEdit 자식, FireMonkey에서는 FMX TEdit입니다. 모델이 텍스트가 놓일 위치를 알려 주고, 호스트가 그 위치에 편집 컨트롤을 둡니다. 텍스트 입력과 IME를 참조하십시오.
4. **드롭다운.** 날짜 선택기의 달력, 시간 선택기의 시간 목록, 날짜 및 시간 선택기에서는 둘을 나란히 놓은 것, 토큰 편집기의 자동 완성 목록은 프레임워크마다 하나의 드롭다운으로 열립니다. VCL에서는 편집 컨트롤에서 포커스를 빼앗지 않는 WS_POPUP 도구 창으로, 컨트롤 아래에 열리고, 공간이 없으면 위로 뒤집히며, 모니터의 작업 영역 안에 머뭅니다. FireMonkey에서는 컨트롤 아래에 놓이는 TPopup입니다.

두 레이어

모델은 두 레이어를 그립니다. Render는 콘텐츠 레이어, 즉 프레임, 텍스트, 날, 견본, 칩, 눈금을 그립니다. RenderOverlay는 그 위에서 자주 바뀌는 것, 즉 호버, 포커스 링, 슬라이더 핸들, 색상 사각형의 커서, 평점 채우기와 그 호버 미리 보기를 그립니다. 드래그, 호버, 포커스 변경은 오버레이만 다시 그리므로 콘텐츠 레이어는 다시 래스터화되지 않습니다. 모델의 모든 변경은 다음 종류 중 하나로 OnModelChange를 통해 알려집니다.

종류	호스트
eckContent	콘텐츠 레이어를 다시 렌더링합니다.
eckOverlay	오버레이만 렌더링합니다.
eckValue	OnChange를 발생시키며, VCL에서는 화면 읽기 프로그램에 새 값을 알립니다.
eckLayout	내부 편집 컨트롤을 이동하고, 그 텍스트를 다시 쓰고, 콘텐츠 레이어를 렌더링합니다.

Units

유닛	내용
sgcEditors_Types	열거형, TsgcEditorModel, TsgcEditorsPainter(테마 색상, 프레임, 포커스 링, 쉐브론, 스핀 화살표), 키 코드, 숫자 서식과 구문 분석, HSV와 16진수 변환.
sgcEditors_Date	일 번호, ISO 8601 및 단순 주 번호, 월 격자, TsgcEditorLocale, 날짜 형식, 단계, 구문 분석.
sgcEditors_Calendar, _Picker, _Slider, _Spin, _Color, _Rating, _Token, _List	모델. sgcEditors_Picker에는 날짜 및 시간 드롭다운의 달력과 시간 목록인 TsgcDateTimeDropDownModel도 있습니다.

sgcEditors_VCL, _VCL_Date, _VCL_Value, _VCL_Color, _VCL_Token	VCL 호스트 기본, 내부 편집 컨트롤, 드롭다운, VCL 컨트롤. Windows 전용입니다.
sgcEditors_FMX, _FMX_Date, _FMX_Value, _FMX_Color, _FMX_Token	FireMonkey용 동일한 것. Delphi XE6부터입니다.
sgcEditors_Data, _VCL_DB, _FMX_DB	TsgcEditorFieldLink와 데이터 인식 컨트롤. 고급 에디션이며, 이 유닛들은 Community에서 비어 있습니다.
sgcEditors_Const, _License, _Reg	버전 및 에디션 상수, 체험판 검사, 디자인 타임 등록.

스레드

컨트롤은 다른 VCL 또는 FireMonkey 컨트롤처럼 메인 스레드에서 사용하며, 이벤트도 메인 스레드에서 발생합니다.

성능

QA 성능 그룹은 Win64의 Delphi 13 릴리스 빌드로 이 시나리오들을 측정합니다. 예산은 각 시나리오의 통과 기준이며, 측정값은 기준 컴퓨터에서의 실행 결과입니다. 월 달력에는 2,500 µs의 Win32 예산도 있으며, Linux64는 E1, E4, E8을 통과 기준 없이 보고합니다.

ID	시나리오	예산	측정값
E1	월 달력 콘텐츠 280 x 240: 제목, 요일 이름, 42일, 주 번호, 범위 띠	1,500 µs	395 µs
E2	달력 오버레이만: 호버 셀과 포커스 링	60 µs	24 µs
E3	비활성화된 날짜 500개, 비활성화된 요일 2개, 이벤트 할당 상태에서 10,000일에 대한 IsDateEnabled	2 ms	0.3 ms
E4	눈금 11개와 레이블이 있는 범위 슬라이더 300 x 48, 전체 렌더링. 핸들 드래그, 오버레이만	400 µs; 150 µs	88 µs; 73 µs
E5	견본 24개, 사각형, 색조 및 알파 막대가 있는 색상 선택기 240 x 300, 전체 렌더링. 사각형에서의 드래그, 오버레이만	1,500 µs; 150 µs	1,018 µs; 114 µs
E6	칩 50개가 있는 토큰 편집기 400 x 120, 레이아웃과 렌더링. 키 입력마다 10,000개 항목의 자동 완성 필터	1,500 µs; 3 ms	436 µs; 0.2 ms
E7	평점, 별 5개 160 x 32, 전체 렌더링. 스핀 편집기 크롬 120 x 28	200 µs; 120 µs	103 µs; 39 µs
E8	날짜 서식 100,000회와 구문 분석 100,000회. ISO 주 번호 1,000,000회	150 ms; 100 ms	23 ms; 27 ms
E9	9종의 컨트롤을 각각 30개씩 가진 VCL 폼의 생성과 첫 그리기. 선택기 드롭다운이 첫 프레임까지 열리는 것	350 ms; 16 ms	조용한 컴퓨터에서 약 340 ms. 5 ms
E10	FireMonkey의 E9 폼	보고만 함	

E9 시간의 대부분은 그리기가 아니라 컨트롤 270개와 그 내부 편집 컨트롤의 창을 만드는 데 쓰입니다. 수백 개의 편집기를 한 번에 표시하는 폼은 숨겨진 페이지의 컨트롤을 필요할 때 만들면 더 빨리 열립니다.

참고 항목

- 컨트롤
- TsgcEditorModel
- 헤드리스 출력
- 텍스트 입력과 IME

컨트롤

sgcEditors에는 컨트롤이 9종 있습니다. 각각 VCL용과 FireMonkey용이 있으며, 게시된 속성과 이벤트는 같습니다.

컨트롤	VCL	FireMonkey	텍스트 입력
월 달력	TsgcVCLMonthCalendar	TsgcFMXMonthCalendar	직접 그림. 월, 연도, 10년 보기, 단일 날짜 또는 범위, 제한, 비활성화된 날짜와 요일, 주 번호.
날짜 선택기	TsgcVCLDatePicker	TsgcFMXDatePicker	네이티브 편집 컨트롤과 드롭다운 달력.
시간 선택기	TsgcVCLTimePicker	TsgcFMXTimePicker	네이티브 편집 컨트롤과 ListInterval분마다의 드롭다운 시간 목록.
날짜 및 시간 선택기	TsgcVCLDateTimePicker	TsgcFMXDateTimePicker	네이티브 편집 컨트롤과 나란히 놓인 달력 및 시간 목록.
범위 슬라이더	TsgcVCLRangeSlider	TsgcFMXRangeSlider	직접 그림. 핸들 하나 또는 둘, 단계, 눈금, 레이블, 값 툴팁, 가로 또는 세로.
색상 선택기	TsgcVCLColorPicker	TsgcFMXColorPicker	견본, HSV 사각형, 색조 및 알파 막대가 그려지며, 네이티브 16진수 편집 컨트롤이 있습니다.
토큰 편집기	TsgcVCLTokenEdit	TsgcFMXTokenEdit	칩이 그려지며, 다음 토큰을 위한 네이티브 편집 컨트롤과 자동 완성 목록이 있습니다.
평점	TsgcVCLRating	TsgcFMXRating	직접 그림. 별, 하트, 원 또는 직접 만든 글리프. 정수, 반, 정확한 값을 지원합니다.
스핀 편집기	TsgcVCLSpinEdit	TsgcFMXSpinEdit	네이티브 편집 컨트롤과 위아래 버튼, 단위, 제한, 자동 반복.

데이터 인식 컨트롤

데이터 인식 컨트롤은 두 번째 열의 컨트롤을 상속하며 DataSource, DataField, public Field를 추가합니다. Registered 및 Trial 에디션에 포함됩니다. 빠른 시작 데이터 인식을 참조하십시오.

VCL	FireMonkey	기본 컨트롤	바인딩 대상
TsgcVCLDBDatePicker	TsgcFMXDBDatePicker	날짜 선택기	필드의 날짜. 시간 부분은 유지됩니다.
TsgcVCLDBTimePicker	TsgcFMXDBTimePicker	시간 선택기	필드의 시간. 날짜 부분은 유지됩니다.
TsgcVCLDBSpinEdit	TsgcFMXDBSpinEdit	스핀 편집기	숫자 필드. 정수 필드이면 스핀 편집기가 esvInteger로 전환됩니다.
TsgcVCLDBRating	TsgcFMXDBRating	평점	숫자 필드.

TsgcVCLDBRangeSlider	TsgcFMXDBRangeSlider	범위 슬라이더	Value는 DataField에, 범위 모드에서는 EndValue는 EndDataField에 바인딩합니다.

모든 컨트롤에 공통되는 것

게시: Theme, AccentColor, TextStyle, ReadOnly, AccessibleName, OnChange, 그리고 Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 프레임워크의 일반적인 속성과 이벤트. Public: Model, FormatSettings, DroppedDown, DropDown, CloseUp, SaveToPNG, SaveToSVG.

모델 클래스

모든 모델은 TsgcEditorModel을 상속합니다.

모델	사용처
TsgcCalendarModel	월 달력과 날짜 선택기의 드롭다운.
TsgcDatePickerModel	날짜, 시간, 날짜 및 시간 선택기. Kind는 edpDate, edpTime, edpDateTime입니다.
TsgcDateTimeDropDownModel	날짜 및 시간 선택기의 드롭다운: 달력과 시간 목록.
TsgcEditorListModel	시간 목록과 자동 완성 목록.
TsgcSliderModel	범위 슬라이더.
TsgcColorPickerModel	색상 선택기.
TsgcTokenModel	토큰 편집기.
TsgcRatingModel	평점.
TsgcSpinModel	스핀 편집기.

참고 항목

- 아키텍처
- 빠른 시작 VCL

설치

sgcEditors는 sgcUI 배포판의 일부입니다. 설치 프로그램 하나가 sgcUI 제품들을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 컴포넌트 페이지에서 **sgcEditors**와 원하는 다른 제품을 선택합니다. 라이선스 서버가 보유한 제품을 반환하고, 설치 프로그램은 제품마다 확인란을 하나씩 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 잠금 해제합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE용으로 의존 관계 순서에 따라 패키지를 사용자의 컴퓨터에서 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcEditors와 함께 설치되는 것

sgcUI 비주얼 제품의 공유 기반인 **sgcUICore**는 sgcEditors와 함께 제공됩니다. 자체 확인란은 없습니다. sgcEditors를 선택하면 설치 프로그램이 자동으로 설치합니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드. 이것이 sgcUI의 유일한 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지로, 제품과 IDE 버전마다 한 세트입니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛. 예를 들어 `LibD13` 입니다.
`Demos\Editors`	sgcEditors 데모: Gallery, GalleryFMX, Calendar, BookingForm, DataAware.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 자체 경로가 필요하지 않습니다.

컴포넌트 팔레트

컴포넌트는 **SGC Editors** 팔레트 페이지에 있습니다. VCL 폼에는 VCL 컨트롤이, FireMonkey 폼에는 FMX 컨트롤이 표시됩니다. Registered 및 Trial 에디션은 컨트롤 9종과 각 프레임워크의 데이터 인식 컨트롤 5종, 즉 28개의 컴포넌트를 등록합니다. Community는 데이터 인식 컨트롤을 제외한 18개의 컨트롤을 등록합니다.

체험판과 Community

- **체험판**: sgcUI 체험판 설치 프로그램은 sgcEditors를 포함한 모든 제품을 체험판 모드로 설치합니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 고급 기능이 없는 sgcEditors가 포함됩니다. 에디션을 참조하십시오.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용하는 IDE 버전의 패키지를 엽니다. `<V>`는 Delphi 버전 접미사이며, 예를 들어 `sgcEditorsD13.dpk`입니다.

패키지	내용
`sgcUICore<V>`	공유 sgcUICore 기반(런타임).
`sgcEditors<V>` / `dclsgcEditors<V>`	sgcEditors 컴포넌트(런타임 / 디자인 타임).

의존 관계 순서대로 빌드하십시오. `sgcUICore`, 그다음 `sgcEditors`, 그다음 `dclsgcEditors`입니다. 그런 다음 `dclsgcEditors` 디자인 타임 패키지를 설치하고, 사용하는 Delphi 버전의 라이브러리 경로(Tools > Options > Language > Delphi Options > Library > Library path)에 `sgcUI\Source`를 추가합니다. C++Builder에서는 사용하는 버전에 맞는 `.cbproj` 프로젝트(예: `sgcEditorsC13.cbproj`)를 열어 빌드합니다. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 만들어집니다.

참고 항목

- 플랫폼
- 빠른 시작 VCL

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE부터 XE8, 10부터 10.4, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE부터 XE8, 10부터 10.4, 11, 12, 13. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다.
VCL	Windows의 TsgcVCL 컨트롤. Win32와 Win64, 모든 Delphi 버전.
FireMonkey	Delphi XE6부터의 TsgcFMX 컨트롤(SGC_EDITORS_FMX). FireMonkey 플랫폼에서 동작합니다. FireMonkey와 sgcUICore만 사용합니다.
Linux64	Delphi 10.2부터의 프레임워크 중립 유닛(SGC_EDITORS_LINUX): 모델, 날짜 계산, 로캘, 그리고 헤드리스 PNG 및 SVG 출력. Linux용 컨트롤은 없습니다.
.NET	WinForms, WPF, Avalonia에서 동일한 API를 제공하는 C# 포트로, 9개의 컨트롤을 모든 호스트에서 사용할 수 있으며 창 없는 PNG 및 SVG 출력을 지원합니다. .NET Framework 4.7.2와 .NET 8 및 9에서 동작합니다. .NET 개요를 참조하십시오.

패키지

sgcEditors에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcEditorsD13.dpk, C++Builder 13에서는 sgcEditorsC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcEditors	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛.	rtl, dbrtl, sgcUICore
sgcEditorsVCL	런타임, VCL	sgcEditors_VCL, sgcEditors_VCL_Color, sgcEditors_VCL_DB, sgcEditors_VCL_Date, sgcEditors_VCL_Token 및 sgcEditors_VCL_Value	rtl, vcl, dbrtl, sgcUICore, sgcUICoreVCL, sgcEditors
sgcEditorsFMX	런타임, FireMonkey, XE6부터	sgcEditors_FMX, sgcEditors_FMX_Color, sgcEditors_FMX_DB, sgcEditors_FMX_Date, sgcEditors_FMX_Token 및 sgcEditors_FMX_Value	rtl, fmx, dbrtl, sgcUICore, sgcUICoreFMX, sgcEditors
dclsgcEditors	디자인 타임, VCL	VCL 편집기를 **SGC Editors** 팔레트 페이지에 등록하며, DisabledDates와 Swatches에는 표준 문자열 목록 편집기를 사용합니다.	designide, rtl, vcl, sgcEditors, sgcEditorsVCL
dclsgcEditorsFMX	디자인 타임, FireMonkey, XE6부터	FireMonkey 편집기를 **SGC Editors** 팔레트 페이지에 등록합니다.	designide, rtl, fmx, sgcEditors, sgcEditorsFMX, dclsgcEditors

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcEditorsFMX에는 dclsgcEditors가 필요하므로 dclsgcEditors를 먼저 설치하십시오.

IDE 버전과 대상 플랫폼

Delphi	sgcEditors	sgcEditorsVCL	sgcEditorsFMX	dclsgcEditors, dclsgcEditorsFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcEditors, sgcEditorsVCL	sgcEditorsFMX	dclsgcEditors, dclsgcEditorsFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcEditors만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcEditors만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcEditorsC13.cbproj, sgcEditorsVCLC13.cbproj, sgcEditorsFMXC13.cbproj, dclsgcEditorsC13.cbproj, dclsgcEditorsFMXC13.cbproj 입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMX 편집기)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcEditorsVCL은 Windows 전용입니다.

- 컨트롤을 폼에 놓거나 해당 유닛(sgcEditors_FMX, sgcEditors_FMX_Color, sgcEditors_FMX_DB, sgcEditors_FMX_Date, sgcEditors_FMX_Token 및 sgcEditors_FMX_Value)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcEditors는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 편집기 모델(값, 날짜, 색, 토큰)입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

VCL 편집기의 내부 텍스트 상자

텍스트 필드가 있는 VCL 편집기는 내부 텍스트 상자(네이티브 편집 창)를 필요할 때 만듭니다. 덕분에 편집기가 많은 폼이 빠르게 표시됩니다. 그때까지는 편집기가 텍스트를 직접 그립니다. 내부 텍스트 상자는 처음 마우스가 올라갔을 때, 편집기가 포커스를 받았을 때, 키보드나 클립보드를 처음 사용했을 때, 또는 코드가 창 핸들을 읽을 때 만들어집니다. 스크린 리더나 UI Automation 클라이언트 같은 접근성 도구는 그 시점부터만 내부 텍스트 상자를 볼 수 있습니다. 편집기 자체는 처음부터 WM_GETTEXT에 응답합니다. 비활성화된 편집기, ReadOnly 편집기, 또는 PasswordChar, 오른쪽에서 왼쪽 읽기, TextHint, 왼쪽 정렬이 아닌 Alignment를 가진 편집기는 즉시 만들며, 디자인 타임에는 항상 만들어집니다.

버전 제약

- 프레임워크 중립 유닛과 VCL 유닛은 Delphi 7에서 컴파일됩니다. 제네릭, 익명 메서드, 인라인 변수, 레코드 헬퍼를 사용하지 않습니다.
- VCL 컨트롤의 TextHint는 Delphi 2009부터 표시됩니다.
- 열거형에는 에디터 접두사(esm, efd, ewn, ecv, eor, etp, ecp, eti, emm, erp, erg, esv, esb)가 붙어 있으므로, C++Builder 헤더가 sgcUI의 다른 제품과 충돌하지 않습니다.

참고 항목

- 에디션
- 아키텍처

빠른 시작 VCL

사용하는 Delphi 버전용 sgcEditors 패키지를 설치합니다. 컴포넌트는 **SGC Editors** 팔레트 페이지에 있습니다.

디자인 타임

1. 폼에 **TsgcVCLDatePicker**를 놓습니다. 개체 검사기에서 MinDate와 Date를 설정하거나, AllowEmpty를 True로 하고 Date를 비워 둡니다.
2. **TsgcVCLSpinEdit**를 놓고 MinValue, MaxValue, Value를 설정한 다음, Units에 guests 같은 접미사를 입력합니다.
3. **TsgcVCLTokenEdit**를 놓고 TextHint를 지정합니다.
4. 세 컨트롤에 OnChange 핸들러 하나를 할당하고 애플리케이션을 실행합니다. 날짜를 입력하거나 F4 또는 Alt+Down으로 달력을 열고, 화살표 키나 마우스 휠로 스핀 편집기를 증감하고, 쉼표로 구분하여 토큰을 입력합니다.

코드에서

컨트롤은 Parent가 있으면 바로 동작합니다. 스핀 편집기의 제한은 Value보다 먼저 설정하십시오. 그렇지 않으면 값이 이전 제한 안으로 맞춰집니다.

참고 항목

- 컨트롤
- 텍스트 입력과 IME
- 키보드와 접근성

빠른 시작 FMX

FireMonkey 컨트롤에는 Delphi XE6 이상이 필요합니다. FireMonkey 폼의 **SGC Editors** 팔레트 페이지에 있으며, VCL 컨트롤과 같은 속성과 이벤트를 가집니다.

디자인 타임

1. 폼에 **TsgcFMXDatePicker**, **TsgcFMXRangeSlider**, **TsgcFMXColorPicker**를 놓습니다.
2. 슬라이더에서 RangeMode를 True로 설정한 다음 Min, Max, Step, TickInterval을 설정하고 ShowLabels를 켭니다.
3. 색상 선택기에서 표시할 Parts와 SelectedColor를 선택합니다.

FireMonkey에서 다른 점

- 드롭다운은 컨트롤 아래의 TPopup입니다. 바깥을 클릭하여 닫으면 열려 있는 동안 한 선택이 취소됩니다. Enter 또는 값을 완성하는 선택으로 닫으면 값이 유지됩니다.
- FireMonkey는 컨트롤이 키를 보기 전에 Tab으로 포커스를 이동하므로, 범위 슬라이더와 색상 선택기는 각 부분 사이를 이동하는 데 Ctrl과 화살표 키를 사용합니다. 키보드와 접근성을 참조하십시오.
- 컨트롤의 접근성 텍스트는 Hint에 들어가며, 직접 지정한 Hint도 유지됩니다. TabStop, Hint, ShowHint는 Delphi 10.1부터 게시됩니다.
- 스핀 편집기의 내부 편집 컨트롤은 모바일 플랫폼에서 NumbersAndPunctuation 가상 키보드를 요청합니다.

코드에서

참고 항목

- 컨트롤
- 플랫폼
- DPI

빠른 시작 C++Builder

사용하는 C++Builder 버전용 sgcEditors 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 인클루드 경로와 라이브러리 경로에 추가한 다음, Delphi와 똑같이 **SGC Editors** 팔레트 페이지의 컴포넌트를 사용합니다. 설치 프로그램이 두 경로를 IDE에 추가해 줍니다. sgcEditors와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있습니다.

C++에서 알아 둘 점

- 컨트롤을 선언하는 유닛의 헤더를 포함하십시오. 달력과 선택기에는 `sgcEditors_VCL_Date.hpp`, 슬라이더, 평점, 스핀 편집기에는 `sgcEditors_VCL_Value.hpp`, 그리고 `sgcEditors_VCL_Color.hpp`, `sgcEditors_VCL_Token.hpp`, `sgcEditors_VCL_DB.hpp`입니다. 열거형은 `sgcEditors_Types.hpp`에 있습니다.
- 이벤트는 클로저이므로 핸들러는 클래스의 __fastcall 멤버 함수입니다. aAccept 같은 var 매개변수는 bool 참조입니다.
- Date, MinDate, MaxDate는 TDate 속성입니다. 여기에 TDateTime을 할당하십시오. 예를 들어 `Picker->Date = Date() + 1`입니다.
- 모델 클래스에는 `new TsgcCalendarModel()`처럼 인수가 없는 생성자가 있으며, 직접 delete해야 합니다. 헤드리스 출력을 참조하십시오.

코드에서

참고 항목

- 빠른 시작 VCL
- 빠른 시작 데이터 인식
- 플랫폼

빠른 시작 데이터 인식

데이터 인식 컨트롤은 날짜 선택기, 시간 선택기, 스핀 편집기, 평점, 범위 슬라이더를 TDBEdit처럼 필드에 바인딩합니다. Registered 및 Trial 에디션에 포함되며, VCL은 sgcEditors_VCL_DB에, FireMonkey는 sgcEditors_FMX_DB에 있습니다. FireMonkey에는 TFieldDataLink가 없으므로 두 프레임워크가 sgcEditors_Data의 TDataLink인 TsgcEditorFieldLink를 공유합니다.

디자인 타임

1. 폼에 데이터셋과 **TDataSource**를 놓습니다.
2. **TsgcVCLDBDatePicker**(또는 FMX 짝)를 놓고, DataSource를 데이터 소스로 설정한 다음 DataField에서 필드를 선택합니다.
3. 범위 모드의 **TsgcVCLDBRangeSlider**에서는 아래쪽 핸들에는 DataField를, 위쪽 핸들에는 EndDataField를 설정합니다.

컨트롤이 필드를 따르는 방식

- 값은 현재 레코드를 따릅니다. null 필드는 선택기와 스핀 편집기에서는 빈 텍스트를, 평점에서는 채워진 글리프 없음을, 슬라이더에서는 핸들을 Min에(끝 핸들은 Max에) 표시합니다. 데이터 인식 선택기와 스핀 편집기에서 AllowEmpty의 기본값은 True입니다.
- 사용자의 변경은 데이터셋을 편집 모드로 만들고 링크를 수정된 것으로 표시합니다. 코드에서 Value나 Date를 설정하는 것도 사용자의 변경으로 간주됩니다. 데이터셋이 편집 모드로 들어갈 수 없으면 필드의 값이 돌아옵니다.
- Enter와 포커스 상실은 입력한 텍스트를 확정하고 필드에 기록합니다. 필드가 값을 거부하면 예외가 발생하고, TDBEdit처럼 포커스는 컨트롤에 남습니다. 게시 전에 데이터셋은 보류 중인 값을 기록합니다. Escape는 필드의 값을 복원합니다.
- 날짜 선택기는 날짜 및 시간 필드의 시간 부분을, 시간 선택기는 날짜 부분을 유지합니다. 정수 필드이면 스핀 편집기가 esvInteger로 전환됩니다.
- 컨트롤은 ReadOnly가 True인 동안, 데이터셋이 없는 동안, 데이터셋 또는 필드가 읽기 전용인 동안, 계산 필드인 경우, 그리고 범위 모드의 슬라이더에서는 끝 필드를 수정할 수 없는 동안 읽기 전용입니다.

코드에서

샘플은 TClientDataSet과 VCL 데이터 인식 컨트롤 5개를 만듭니다. 애플리케이션에 midas.dll이 필요하지 않도록 프로젝트의 uses 절에 MidasLib를 추가하십시오.

참고 항목

- 컨트롤
- DB 날짜 선택기
- TsgcEditorFieldLink
- 에디션
- 문제 해결

월 달력

클래스: TsgcVCLMonthCalendar (유닛 sgcEditors_VCL_Date), TsgcFMXMonthCalendar (유닛 sgcEditors_FMX_Date)
월 달력은 편집 컨트롤이 없는, 직접 그리는 포커스 가능한 컨트롤입니다. 요일 이름과 선택적으로 주 번호를 붙여 한 달을 표시하며, 사용자가 제목을 클릭하거나 Ctrl+Up을 누르면 1년의 12개월, 그리고 10년 단위의 연도로 축소됩니다. 날짜 하나를 선택하거나, SelectionMode가 esmRange이면 날짜 범위를 선택합니다. MinDate와 MaxDate 밖의 날, DisabledDates, DisabledWeekDays, OnIsDateDisabled가 비활성화하는 날은 취소선으로 그려지며 선택할 수 없습니다. 같은 모델인 TsgcCalendarModel이 날짜 선택기의 드롭다운도 그립니다.
VCL 클래스는 TsgcVCLCustomEditor를, FireMonkey 클래스는 TsgcFMXCustomEditor를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Date	TDate	esmSingle 모드에서 선택된 날짜이며, 선택된 날짜가 없으면 0입니다. 범위 모드는 이 값을 건드리지 않습니다.
SelectionMode	TsgcEditorSelectionMode	esmSingle(기본값)은 날짜 하나를 선택합니다. esmRange는 범위를 선택합니다. 첫 번째 클릭으로 시작하고 두 번째 클릭으로 끝나며, Shift와 클릭 또는 Shift와 화살표 키로 첫 번째 날부터 범위를 넓힙니다.
RangeStart, RangeEnd	TDate	범위의 첫 날과 마지막 날이며, 양 끝을 포함합니다.
MaxRangeDays	Integer	범위가 담을 수 있는 최대 일수이며, 양 끝을 포함합니다. 0(기본값)은 제한 없음입니다.
MinDate, MaxDate	TDate	사용자가 선택할 수 있는 첫 날과 마지막 날로, 하루 단위로 비교됩니다. 0(기본값)은 제한 없음이며, 뒤바뀐 제한은 서로 교환됩니다.
DisabledDates	TStrings	선택할 수 없는 날로, 한 줄에 ISO 날짜(yyyy-mm-dd) 하나씩입니다.
DisabledWeekDays	TsgcEditorWeekDays	선택할 수 없는 요일로, ewdMonday부터 ewdSunday까지의 집합입니다. 기본값은 []입니다.
FirstDayOfWeek	TsgcEditorFirstDayOfWeek	efdLocale(기본값)은 한 주의 첫 요일을 운영 체제에서 가져오며, 없으면 월요일입니다. efdMonday, efdSaturday, efdSunday는 이를 강제로 지정합니다.
ShowWeekNumbers	Boolean	각 행의 주 번호를 담은 열을 추가합니다. 기본값은 False입니다.
WeekNumbering	TsgcEditorWeekNumbering	ewnISO8601(기본값): 1주차는 그해의 첫 목요일을 포함합니다. ewnSimple: 1주차는 1월 1일을 포함합니다.
ShowToday	Boolean	현재 날짜를 표시합니다. 기본값은 True입니다.
ShowOtherMonthDays	Boolean	격자의 여섯 주를 채우는 이전 달과 다음 달의 날을 표시합니다. 기본값은 True입니다.

View	TsgcEditorCalendarView	ecvMonth(기본값), ecvYear(1년의 12개월), ecvDecade(10년 단위의 연도).
DisplayMonth	TDate	표시 중인 달의 첫 날이며, 항상 키보드 포커스가 있는 달입니다. 폼에는 저장되지 않습니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
FormatSettings	TFormatSettings	public. 컨트롤이 사용하는 이름, 형식, 구분 기호입니다. 사용자 로캘의 설정으로 시작합니다. 현지화를 참조하십시오.
CalendarModel	TsgcCalendarModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환값	설명
GoToDate(aDate)		aDate의 달을 제한 안으로 맞춰 표시하고 키보드 포커스를 그곳으로 옮깁니다. 날짜를 선택하지는 않습니다.
IsDateEnabled(aDate)	Boolean	날짜가 제한 안에 있고, 비활성화된 요일이 아니고, DisabledDates에 없고, OnIsDateDisabled로 비활성화되지 않았으면 True입니다.
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	선택된 날짜 또는 범위가 바뀌었습니다.
OnSelect	TNotifyEvent	esmSingle 모드에서 마우스, Enter 또는 Space로 날짜가 선택되었습니다.
OnRangeSelect	TNotifyEvent	범위가 완성되었습니다. 두 번째 클릭, 또는 Shift와 클릭, Shift와 화살표 키입니다.
OnIsDateDisabled	TsgcCalendarIsDateDisabledEvent	(Sender; const aDate: TDateTime; var aDisabled: Boolean). aDisabled를 True로 설정하면 데이터베이스의 공휴일 같은 날을 비

		활성화할 수 있습니다. 달력이 표시하는 날과 선택 중인 날에 대해 호출되므로 빠르게 처리하십시오.
OnViewChange	TNotifyEvent	보기가 바뀌었습니다. 사용자가 확대하거나 축소했습니다.
OnDisplayMonthChange	TNotifyEvent	다른 달이 표시됩니다.

키보드

화살표 키는 포커스를 하루 또는 한 주씩, PageUp과 PageDown은 한 달씩, Ctrl을 누르면 1년씩 이동합니다. Ctrl+Up은 축소합니다. Enter나 Space는 포커스된 날을 선택하거나, 연도 및 10년 보기에서는 확대합니다. Shift와 화살표 키로 범위를 넓힙니다. 비활성화된 날은 건너뜁니다.

예제: 최대 2주의 숙박

참고 항목

- 날짜 선택기
- 날짜 및 시간 선택기
- 현지화
- TsgcEditorLocale

날짜 선택기

클래스: TsgcVCLDatePicker (유닛 sgcEditors_VCL_Date), TsgcFMXDatePicker (유닛 sgcEditors_FMX_Date)
날짜 선택기는 드롭다운 달력이 있는 날짜용 네이티브 편집 컨트롤입니다. 사용자는 날짜를 입력하거나, 캐럿 위치의 일, 월, 연을 Up 과 Down 으 로 증 감 하 거 나, 달 력 에 서 선 택 합 니 다. 기 본 클 래 스 인 TsgcVCLCustomDatePicker 와 TsgcFMXCustomDatePicker, 그리고 모델인 TsgcDatePickerModel은 시간 선택기, 날짜 및 시간 선택기와 공유됩니다.
VCL 클래스는 TsgcVCLCustomDatePicker를, FireMonkey 클래스는 TsgcFMXCustomDatePicker를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Date	TDate	날짜입니다. AllowEmpty가 True이면 먼저 IsEmpty를 읽으십시오.
DisplayFormat	string	텍스트의 형식입니다(예: dd/mm/yyyy 또는 ddd d mmm yyyy). 비어 있으면(기본값) FormatSettings의 짧은 날짜 형식을 사용합니다. 텍스트 입력과 IME를 참조하십시오.
AllowEmpty	Boolean	사용자가 텍스트를 지워 값을 비울 수 있게 합니다. 기본값은 False이며, 이 경우 빈 텍스트는 마지막 값으로 되돌아갑니다.
IsEmpty	Boolean	public, 읽기 전용. 값이 비어 있는 동안 True입니다.
TextHint	string	비어 있는 편집 컨트롤에 표시되는 회색 힌트입니다. VCL에서는 Delphi 2009부터 표시되며, FireMonkey에서는 편집 컨트롤의 TextPrompt로 표시됩니다.
DroppedDown	Boolean	public. 드롭다운이 열려 있는 동안 True입니다. 설정하면 드롭다운을 열거나 닫습니다.
MinDate, MaxDate	TDate	사용자가 선택할 수 있는 첫 날과 마지막 날로, 하루 단위로 비교됩니다. 0(기본값)은 제한 없음이며, 뒤바뀐 제한은 서로 교환됩니다.
DisabledDates	TStrings	선택할 수 없는 날로, 한 줄에 ISO 날짜(yyyy-mm-dd) 하나씩입니다.
DisabledWeekDays	TsgcEditorWeekDays	선택할 수 없는 요일로, ewdMonday부터 ewdSunday까지의 집합입니다. 기본값은 []입니다.
FirstDayOfWeek, ShowWeekNumbers		월 달력과 같으며, 드롭다운 달력용입니다.
DropDownCalendar	TsgcCalendarModel	public, 읽기 전용. 드롭다운 달력의 모델입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.

AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
FormatSettings	TFormatSettings	public. 컨트롤이 사용하는 이름, 형식, 구분 기호입니다. 사용자 로캘의 설정으로 시작합니다. 현지화를 참조하십시오.
PickerModel	TsgcDatePickerModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환값	설명
Clear		AllowEmpty가 True이면 값을 비웁니다. 그렇지 않으면 아무 일도 하지 않습니다.
DropDown		버튼, F4, Alt+Down처럼 드롭다운을 엽니다.
CloseUp(aAccept)		드롭다운을 닫습니다. aAccept가 True이면 텍스트를 확정하고, False이면 열릴 때의 값으로 복원합니다.
IsDateEnabled(aDate)	Boolean	날짜가 제한 안에 있고 드롭다운 달력에서 비활성화되지 않았으면 True입니다. 확정 시 이 값이 False인 날짜는 거부됩니다.
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	새 값입니다. 입력하여 확정했거나, 드롭다운에서 선택했거나, Up과 Down으로 증감한 경우입니다.
OnDropDown, OnCloseUp	TNotifyEvent	드롭다운이 열리거나 닫힙니다.
OnParseError	TsgcDatePickerParseErrorEvent	(Sender; const aText: string; var aValue: TDateTime; var aAccept: Boolean). 확정된 텍스트를 구문 분석할 수 없습니다. aValue에는 현재 값이, aAccept에는 False가 전달됩니다. 다른 값을 사용하려면 aValue를 설정하고 aAccept를 True로 설정하십시오. 그 값은 제한 안으로 맞춰집니다. 그렇지 않으면 마지막 유효한 값으로 돌아갑니다.
OnIsDateDisabled	TsgcCalendarIsDateDisabledEvent	월 달력과 같습니다. 드롭다운과 확정 시 하루를 비활성화합니다.

입력과 드롭다운

- 텍스트는 네이티브 편집 컨트롤에 입력되므로 IME, 클립보드, 실행 취소가 동작합니다. 구문 분석은 관대하며 FormatSettings의 이름과 구분 기호를 따릅니다. 텍스트 입력과 IME를 참조하십시오.
- Up과 Down은 입력한 텍스트를 확정한 다음, 캐럿 위치의 필드를 증감하고 다시 선택합니다. 필드는 범위 안에서 순환합니다. 제한은 증감 후에 적용됩니다.
- Enter, 포커스 상실, 드롭다운에서의 선택이 텍스트를 확정합니다. 제한 밖의 값은 제한 안으로 이동되고, 비활성화된 날짜는 거부됩니다.
- F4, Alt+Down, 버튼은 현재 값으로 드롭다운을 엽니다. 열려 있는 동안 드롭다운은 입력한 텍스트를 따라가고, Up, Down, PageUp, PageDown은 드롭다운을 조작하며, Escape는 드롭다운을 닫고 열릴 때의 값과 텍스트로 복원합니다.

예제: "today"를 이해하는 마감일

참고 항목

- 월 달력
- 날짜 및 시간 선택기
- DB 날짜 선택기
- 텍스트 입력과 IME

시간 선택기

클래스: TsgcVCLTimePicker (유닛 sgcEditors_VCL_Date), TsgcFMXTimePicker (유닛 sgcEditors_FMX_Date)
시간 선택기는 시간 목록 드롭다운이 있는 하루 중 시간용 네이티브 편집 컨트롤입니다. 사용자는 시간을 입력하거나, 캐럿 위치의 시, 분, 초, AM과 PM을 Up과 Down으로 변경하거나, 목록에서 시간을 선택합니다. 목록은 00:00에서 시작하며 MinTime과 MaxTime 안에서 ListInterval분마다 시간을 담고, DisplayFormat의 시간 필드로 표기합니다.
VCL 클래스는 TsgcVCLCustomDatePicker를, FireMonkey 클래스는 TsgcFMXCustomDatePicker를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Time	TTime	하루 중 시간입니다. 시간 선택기는 00:00에서 시작합니다.
DisplayFormat	string	텍스트의 형식입니다(예: hh:nn 또는 h:nn ampm). ss가 있는 형식은 초를 표시합니다. 비어 있으면(기본값) FormatSettings의 짧은 시간 형식을 사용합니다.
MinTime, MaxTime	TTime	사용자가 설정할 수 있는 가장 이른 시간과 가장 늦은 시간. 0(기본값)이면 그쪽에는 제한이 없습니다.
ListInterval	Integer	드롭다운 목록에서 두 시간 사이의 분 간격. 기본값은 30입니다. 0이면 드롭다운과 그 버튼이 없어집니다.
AllowEmpty	Boolean	사용자가 텍스트를 지워 값을 비울 수 있게 합니다. 기본값은 False이며, 이 경우 빈 텍스트는 마지막 값으로 되돌아갑니다.
IsEmpty	Boolean	public, 읽기 전용. 값이 비어 있는 동안 True입니다.
TextHint	string	비어 있는 편집 컨트롤에 표시되는 회색 힌트입니다. VCL에서는 Delphi 2009부터 표시되며, FireMonkey에서는 편집 컨트롤의 TextPrompt로 표시됩니다.
DroppedDown	Boolean	public. 드롭다운이 열려 있는 동안 True입니다. 설정하면 드롭다운을 열거나 닫습니다.
DropDownTimeList	TsgcEditorListModel	public, 읽기 전용. 드롭다운 목록의 모델입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
FormatSettings	TFormatSettings	public. 컨트롤이 사용하는 이름, 형식, 구분 기호입니다. 사용자 로캘의 설정으로 시작합니다. 현지화를 참조하십시오.

PickerModel	TsgcDatePickerModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환 값	설명
Clear		AllowEmpty가 True이면 값을 비웁니다. 그렇지 않으면 아무 일도 하지 않습니다.
DropDown		버튼, F4, Alt+Down처럼 드롭다운을 엽니다.
CloseUp(aAccept)		드롭다운을 닫습니다. aAccept가 True이면 텍스트를 확정하고, False이면 열릴 때의 값으로 복원합니다.
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	새 값입니다. 입력하여 확정했거나, 드롭다운에서 선택했거나, Up과 Down으로 증감한 경우입니다.
OnDropDown, OnCloseUp	TNotifyEvent	드롭다운이 열리거나 닫힙니다.
OnParseError	TsgcDatePickerParseErrorEvent	(Sender; const aText: string; var aValue: TDateTime; var aAccept: Boolean). 확정된 텍스트를 구문 분석할 수 없습니다. aValue에는 현재 값이, aAccept에는 False가 전달됩니다. 다른 값을 사용하려면 aValue를 설정하고 aAccept를 True로 설정하십시오. 그 값은 제한 안으로 맞춰집니다. 그렇지 않으면 마지막 유효한 값으로 돌아갑니다.

입력과 드롭다운

- 텍스트는 네이티브 편집 컨트롤에 입력되므로 IME, 클립보드, 실행 취소가 동작합니다. 구문 분석은 관대하며 FormatSettings의 이름과 구분 기호를 따릅니다. 텍스트 입력과 IME를 참조하십시오.
- Up과 Down은 입력한 텍스트를 확정한 다음, 캐럿 위치의 필드를 증감하고 다시 선택합니다. 필드는 범위 안에서 순환합니다. 제한은 증감 후에 적용됩니다.
- Enter, 포커스 상실, 드롭다운에서의 선택이 텍스트를 확정합니다. 제한 밖의 값은 제한 안으로 이동되고, 비활성화된 날짜는 거부됩니다.
- F4, Alt+Down, 버튼은 현재 값으로 드롭다운을 엽니다. 열려 있는 동안 드롭다운은 입력한 텍스트를 따라가고, Up, Down, PageUp, PageDown은 드롭다운을 조작하며, Escape는 드롭다운을 닫고 열릴 때의 값과 텍스트로 복원합니다.

예제: 업무 시간 내의 회의

참고 항목

- 날짜 선택기
- 날짜 및 시간 선택기
- DB 시간 선택기

날짜 및 시간 선택기

클래스: TsgcVCLDateTimePicker (유닛 sgcEditors_VCL_Date), TsgcFMXDateTimePicker (유닛 sgcEditors_FMX_Date)
날짜 및 시간 선택기는 날짜와 시간을 하나의 텍스트로 편집합니다. 드롭다운에는 달력과 시간 목록이 나란히 표시됩니다. 달력에서 선택해도 드롭다운은 열린 채로 유지되어 사용자가 이어서 시간을 선택할 수 있고, 시간 목록에서 선택하면 닫힙니다. MinDate, MaxDate, 비활성화된 날이 날짜를 제한하고, MinTime과 MaxTime이 하루의 시간을 제한합니다.
VCL 클래스는 TsgcVCLCustomDatePicker를, FireMonkey 클래스는 TsgcFMXCustomDatePicker를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
DateTime	TDateTime	날짜와 시간입니다.
DisplayFormat	string	텍스트의 형식입니다(예: dd/mm/yyyy hh:nn). 비어 있으면(기본값) FormatSettings의 짧은 날짜와 짧은 시간을 공백으로 구분하여 사용합니다.
MinDate, MaxDate	TDate	사용자가 선택할 수 있는 첫 날과 마지막 날로, 하루 단위로 비교됩니다. 0(기본값)은 제한 없음이며, 뒤바뀐 제한은 서로 교환됩니다.
DisabledDates	TStrings	선택할 수 없는 날로, 한 줄에 ISO 날짜(yyyy-mm-dd) 하나씩입니다.
DisabledWeekDays	TsgcEditorWeekDays	선택할 수 없는 요일로, ewdMonday부터 ewdSunday까지의 집합입니다. 기본값은 []입니다.
MinTime, MaxTime	TTime	하루 중 가장 이른 시간과 가장 늦은 시간입니다. 0(기본값)이면 그쪽에는 제한이 없습니다.
ListInterval	Integer	시간 목록에서 두 시간 사이의 분 간격입니다. 기본값은 30입니다.
FirstDayOfWeek, ShowWeekNumbers		월 달력과 같으며, 드롭다운 달력용입니다.
AllowEmpty	Boolean	사용자가 텍스트를 지워 값을 비울 수 있게 합니다. 기본값은 False이며, 이 경우 빈 텍스트는 마지막 값으로 되돌아갑니다.
IsEmpty	Boolean	public, 읽기 전용. 값이 비어 있는 동안 True입니다.
TextHint	string	비어 있는 편집 컨트롤에 표시되는 회색 힌트입니다. VCL에서는 Delphi 2009부터 표시되며, FireMonkey에서는 편집 컨트롤의 TextPrompt로 표시됩니다.
DroppedDown	Boolean	public. 드롭다운이 열려 있는 동안 True입니다. 설정하면 드롭다운을 열거나 닫습니다.
DropDownCalendar, DropDownTimeList	TsgcCalendarModel, TsgcEditorListModel	public, 읽기 전용. 드롭다운의 두 부분의 모델입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.

AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
FormatSettings	TFormatSettings	public. 컨트롤이 사용하는 이름, 형식, 구분 기호입니다. 사용자 로캘의 설정으로 시작합니다. 현지화를 참조하십시오.
PickerModel	TsgcDatePickerModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환값	설명
Clear		AllowEmpty가 True이면 값을 비웁니다. 그렇지 않으면 아무 일도 하지 않습니다.
DropDown		버튼, F4, Alt+Down처럼 드롭다운을 엽니다.
CloseUp(aAccept)		드롭다운을 닫습니다. aAccept가 True이면 텍스트를 확정하고, False이면 열릴 때의 값으로 복원합니다.
IsDateEnabled(aDate)	Boolean	날짜가 제한 안에 있고 드롭다운 달력에서 비활성화되지 않았으면 True입니다.
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	새 값입니다. 입력하여 확정했거나, 드롭다운에서 선택했거나, Up과 Down으로 증감한 경우입니다.
OnDropDown, OnCloseUp	TNotifyEvent	드롭다운이 열리거나 닫힙니다.
OnParseError	TsgcDatePickerParseErrorEvent	(Sender; const aText: string; var aValue: TDateTime; var aAccept: Boolean). 확정된 텍스트를 구문 분석할 수 없습니다. aValue에는 현재 값이, aAccept에는 False가 전달됩니다. 다른 값을 사용하려면 aValue를 설정하고 aAccept를 True로 설정하십시오. 그 값은 제한 안으로 맞춰집니다. 그렇지 않으면 마지막 유효한 값으로 돌아갑니다.
OnIsDateDisabled	TsgcCalendarIsDateDisabledEvent	월 달력과 같습니다.

입력과 드롭다운

- 텍스트는 네이티브 편집 컨트롤에 입력되므로 IME, 클립보드, 실행 취소가 동작합니다. 구문 분석은 관대하며 FormatSettings의 이름과 구분 기호를 따릅니다. 텍스트 입력과 IME를 참조하십시오.
- Up과 Down은 입력한 텍스트를 확정한 다음, 캐럿 위치의 필드를 증감하고 다시 선택합니다. 필드는 범위 안에서 순환합니다. 제한은 증감 후에 적용됩니다.
- Enter, 포커스 상실, 드롭다운에서의 선택이 텍스트를 확정합니다. 제한 밖의 값은 제한 안으로 이동되고, 비활성화된 날짜는 거부됩니다.
- F4, Alt+Down, 버튼은 현재 값으로 드롭다운을 엽니다. 열려 있는 동안 드롭다운은 입력한 텍스트를 따라가고, Up, Down, PageUp, PageDown은 드롭다운을 조작하며, Escape는 드롭다운을 닫고 열릴 때의 값과 텍스트로 복원합니다.

날짜 및 시간 선택기의 드롭다운에서 Up, Down, PageUp, PageDown은 캐럿이 날짜 필드에 있으면 달력을, 시간 필드에 있으면 시간 목록을 조작합니다.

예제: 배송 시간대

참고 항목

- 날짜 선택기
- 시간 선택기
- 월 달력

범위 슬라이더

클래스: TsgcVCLRangeSlider (유닛 sgcEditors_VCL_Value), TsgcFMXRangeSlider (유닛 sgcEditors_FMX_Value)
범위 슬라이더는 직접 그리는 포커스 가능한 컨트롤로, 핸들 하나로 Min과 Max 사이의 숫자를, RangeMode에서는 핸들 둘로 범위를 선택합니다. 가로(Min이 왼쪽) 또는 세로(Min이 아래쪽)일 수 있습니다. 눈금, 레이블, 드래그 중인 핸들 위의 값 툴팁을 그릴 수 있습니다. 레일을 클릭하면 가장 가까운 핸들이 그곳으로 이동해 드래그되며, 핸들끼리 교차하지 않습니다.
RangeMode에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 속성은 유지되고 핸들은 하나만 그려져 편집되며, EndValue는 유지되지만 표시되지 않습니다. 에디션을 참조하십시오.
VCL 클래스는 TsgcVCLCustomEditor를, FireMonkey 클래스는 TsgcFMXCustomEditor를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Min, Max	Double	눈금의 양 끝입니다. 기본값은 0과 100입니다.
Value	Double	값, 또는 RangeMode에서 범위의 아래쪽 끝입니다. 기본값은 0입니다.
EndValue	Double	RangeMode에서 범위의 위쪽 끝입니다. 기본값은 100입니다.
RangeMode	Boolean	핸들 둘, Value와 EndValue. 기본값은 False입니다. Registered 및 Trial 에디션입니다.
MinRange	Double	두 핸들 사이의 최소 거리. 기본값은 0입니다.
Step	Double	모든 값은 Min에 Step의 배수를 더한 값에 맞춰지며, 화살표 키는 한 Step씩 이동합니다. 기본값은 1입니다. 0은 연속이며, 이 경우 화살표 키는 눈금의 1%씩 이동합니다.
LargeStep	Double	PageUp과 PageDown의 단계. 기본값은 10입니다. 0은 눈금의 10분의 1입니다.
Orientation	TsgcEditorOrientation	eorHorizontal(기본값) 또는 eorVertical.
TickInterval	Double	두 눈금 사이의 거리. 0(기본값)이면 눈금을 그리지 않습니다.
TickPlacement	TsgcEditorTickPlacement	etpBottomRight(기본값), etpTopLeft, etpBoth, etpNone.
ShowLabels	Boolean	각 눈금에 값을 씁니다. 눈금이 없으면 Min과 Max에 씁니다. 겹치는 레이블은 생략됩니다. 기본값은 False입니다.
LabelFormat	string	레이블과 값 툴팁용 FormatFloat 마스크. 비어 있으면(기본값) Step에 필요한 만큼의 소수 자릿수를 씁니다.
ShowValueTip	Boolean	드래그하는 동안 핸들 위에 값을 표시합니다. 기본값은 True입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.

ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
FormatSettings	TFormatSettings	public. 컨트롤이 사용하는 이름, 형식, 구분 기호입니다. 사용자 로캘의 설정으로 시작합니다. 현지화를 참조하십시오.
SliderModel	TsgcSliderModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환 값	설명
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	Value 또는 EndValue가 바뀌었습니다. 드래그하는 동안 새 값마다 발생합니다.
OnChangeComplete	TNotifyEvent	사용자가 변경을 마쳤습니다. 마우스 버튼 놓기, 키 놓기, 휠 한 단계입니다. 쿼리처럼 드래그 중에는 실행하지 말아야 하는 작업에 사용하십시오.
OnFormatLabel	TsgcSliderFormatLabelEvent	(Sender; aValue: Double; var aText: string). 통화나 단위처럼 레이블과 값 툴팁의 텍스트를 씁니다.

키보드

화살표 키는 활성 핸들을 한 Step, PageUp과 PageDown은 한 LargeStep 이동하고, Home과 End는 양 끝으로 이동합니다. RangeMode에서 Tab은 아래쪽 핸들에서 위쪽 핸들로, Shift+Tab은 반대로 이동하며, 그다음 Tab은 컨트롤을 벗어납니다. Ctrl+Left 또는 Ctrl+Down은 아래쪽 핸들을, Ctrl+Right 또는 Ctrl+Up은 위쪽 핸들을 활성화합니다. FireMonkey에서 Tab은 항상 다음 컨트롤로 이동하므로 Ctrl 키로 핸들을 전환합니다.

예제: 가격 필터

참고 항목

- DB 범위 슬라이더
- 스핀 편집기
- 키보드와 접근성

색상 선택기

클래스: TsgcVCLColorPicker (유닛 sgcEditors_VCL_Color), TsgcFMXColorPicker (유닛 sgcEditors_FMX_Color)
색상 선택기는 위에서부터 견본 격자, 채도와 명도 사각형, 색조 막대, 알파 막대, 미리 보기와 16진수 텍스트용 네이티브 편집 컨트롤이 있는 행을 표시합니다. Parts는 이 중 표시할 부분을 선택합니다. 16진수 편집 컨트롤은 #RGB, #RGBA, #RRGGBB, #RRGGBBAA 형식의 텍스트를 받습니다. 선택기는 색조를 색상과 별도로 유지하므로, 회색도 색조를 유지하며 사각형이 튀지 않습니다.
HSV 사각형과 알파 막대에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 그려지지 않으며, 알파는 항상 $FF이고 16진수 텍스트는 #RRGGBB입니다. 에디션을 참조하십시오.
VCL 클래스는 TsgcVCLCustomEditor를, FireMonkey 클래스는 TsgcFMXCustomEditor를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
SelectedColor	TsgcUIColor	ARGB 형식의 색상입니다(예: $FF2A78D6). FireMonkey의 TAlphaColor와 같은 레이아웃입니다.
SelectedVCLColor	TColor	VCL 전용, public. 불투명 TColor로서의 색상이며 읽고 쓸 수 있습니다.
Parts	TsgcEditorColorParts	표시할 부분: ecpSwatches, ecpHSV, ecpAlpha, ecpHex. 기본값은 네 가지 모두입니다.
Swatches	TStrings	견본 색상으로, 한 줄에 #1E88E5 같은 16진수 색상 하나씩입니다. 기본값은 sgcUICore 팔레트와 회색에서 가져온 24색입니다.
SwatchSize	Double	논리 픽셀 단위의 견본 크기. 기본값은 20입니다.
SwatchColumns	Integer	한 행의 견본 수. 기본값은 8입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
ColorModel	TsgcColorPickerModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환 값	설명

SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	색상이 바뀌었습니다. 사각형이나 막대를 드래그하는 동안 새 색상마다 발생합니다.
OnSelect	TNotifyEvent	색상이 선택되었습니다. 견본 클릭, 드래그 후 마우스 버튼 놓기, Enter, 또는 Enter나 포커스 상실로 확정된 16진수 텍스트입니다.

키보드

- VCL에서 컨트롤에는 포커스 위치가 두 곳 있습니다. Tab으로 각 부분을 이동한 다음 16진수 편집 컨트롤로 들어가고, 다음 Tab으로 벗어납니다. 읽기 전용이면 컨트롤은 한 곳입니다.
- FireMonkey에서 컨트롤은 탭 정지 하나에 내부 정지 둘입니다. Ctrl+화살표 키로 각 부분과 16진수 편집 컨트롤 사이를 이동합니다.
- 견본에서는 화살표 키, Home, End로 포커스를 옮기고 Enter나 Space로 견본을 선택합니다. 사각형에서는 왼쪽과 오른쪽 키로 채도를, 위쪽과 아래쪽 키로 명도를 1%씩 바꿉니다. 막대에서는 화살표 키로 1도 또는 1%씩 이동합니다. Shift를 누르면 열 배씩 이동합니다.
- 색상이 아닌 16진수 텍스트는 색상을 다시 표시하며, Escape로 복원합니다.

예제: 레이블의 브랜드 색상

예제: FireMonkey의 채우기 색상

참고 항목

- 토큰 편집기
- 테마
- 에디션

토큰 편집기

클래스: TsgcVCLTokenEdit (유닛 sgcEditors_VCL_Token), TsgcFMXTokenEdit (유닛 sgcEditors_FMX_Token)
토큰 편집기는 태그나 이메일 주소 같은 토큰의 목록을 담고, 삭제 버튼이 있는 칩으로 그립니다. 칩은 왼쪽에서 오른쪽으로 줄 바꿈되고, 마지막 칩 뒤의 네이티브 편집 컨트롤이 다음 토큰을 받으므로 IME, 클립보드, 실행 취소가 동작합니다. 구분 문자, Enter, Tab, 포커스 상실이 입력한 텍스트를 칩으로 바꾸며, a, b, c를 붙여넣으면 칩 세 개가 됩니다. 각 토큰은 공백이 제거되고, OnValidateToken으로 검사되고, 중복이 확인되며, MaxTokens에 대해 집계됩니다. 칩이 들어가지 않으면 컨트롤은 세로로 스크롤됩니다.
자동 완성 목록에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 속성은 유지되지만 목록이 열리지 않습니다. 에디션을 참조하십시오.
VCL 클래스는 TsgcVCLCustomEditor를, FireMonkey 클래스는 TsgcFMXCustomEditor를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Tokens	TStrings	토큰을 순서대로 담습니다. 목록을 바꾸면 칩이 바뀝니다.
Delimiters	string	토큰을 끝내는 문자. 기본값은 ,; 이며 줄 바꿈은 항상 분할합니다. 비우면 Enter와 줄 바꿈만 남습니다.
AllowDuplicates	Boolean	이미 목록에 있는 토큰을 추가합니다. 기본값은 False입니다.
CaseSensitive	Boolean	중복을 찾을 때 토큰을 대소문자를 구분하여 비교합니다. 기본값은 False입니다.
MaxTokens	Integer	목록이 받는 최대 토큰 수. 0(기본값)은 제한 없음입니다. 제한을 넘는 텍스트는 편집 컨트롤에 남습니다.
InvalidTokens	TsgcEditorInvalidTokens	etiReject(기본값)는 거부된 텍스트를 편집 컨트롤에 남깁니다. etiMark는 이를 표시된 칩으로 추가하며, 어느 것인지는 TokenModel.IsTokenInvalid(aIndex)로 알 수 있습니다.
TextHint	string	빈 편집 컨트롤의 회색 힌트.
AutoCompleteItems	TStrings	자동 완성 목록의 후보. Registered 및 Trial 에디션입니다.
AutoCompleteMatch	TsgcEditorMatchMode	emmStartsWith(기본값) 또는 emmContains.
AutoCompleteMinChars	Integer	목록이 열리기 전에 사용자가 입력하는 문자 수. 기본값은 1입니다.
AutoCompleteMaxItems	Integer	목록에 표시하는 최대 행 수. 0(기본값)은 제한 없음입니다.
DroppedDown	Boolean	public. 자동 완성 목록이 열려 있는 동안 True입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.

ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
TokenModel	TsgcTokenModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환값	설명
AddToken(aText)	Boolean	토큰 하나를 분할하지 않고, 공백 제거, OnValidateToken, 중복 검사, MaxTokens를 거쳐 추가합니다. 추가되었으면 True입니다.
RemoveToken(aIndex)	Boolean	aIndex의 토큰을 제거합니다. 제거되었으면 True입니다.
Clear		모든 토큰과 입력한 텍스트를 제거합니다.
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	토큰 목록이 바뀌었습니다.
OnValidateToken	TsgcTokenValidateEvent	(Sender; var aText: string; var aValid: Boolean). aText는 공백이 제거된 상태로 전달되며, 정규화 등을 위해 바꿀 수 있습니다. 토큰을 거부하려면 aValid를 False로 설정하거나, etiMark로 표시하십시오.
OnTokenAdded, OnTokenRemoved	TsgcTokenEvent	(Sender; aIndex: Integer; const aText: string). 칩이 추가되거나 제거되었습니다.
OnAutoComplete	TsgcTokenAutoCompleteEvent	(Sender; const aText: string; aItems: TStrings). aItems는 목록의 후보입니다. 입력한 텍스트에 맞게, 예를 들어 데이터베이스에서 바꾸면 목록이 이를 필터링합니다. Registered 및 Trial 에디션입니다.

키보드

빈 편집 컨트롤에서 Backspace 또는 Left를 누르면 마지막 칩에 포커스가 가고, Backspace를 한 번 더 누르면 제거됩니다. Left와 Right로 칩을 이동하고, 마지막 칩에서 Right를 누르면 편집 컨트롤로 돌아가며, Delete는 포커스된 칩을 제거하고, Escape 또는 입력으로 편집 컨트롤로 돌아갑니다. 자동 완성 목록이 열려 있는 동안 Up, Down, PageUp, PageDown은 목록 안에서 이동하고, Enter는 강조된 행을 선택하며, Escape는 닫습니다.

예제: 이메일 수신자

참고 항목

- 텍스트 입력과 IME
- 키보드와 접근성
- 에디션

평점

클래스: TsgcVCLRating (유닛 sgcEditors_VCL_Value), TsgcFMXRating (유닛 sgcEditors_FMX_Value)
평점은 직접 그리는 포커스 가능한 글리프의 행으로, 별, 하트, 원 또는 직접 만든 글리프를 사용할 수 있습니다. 클릭하면 마우스 아래의 값이 선택되고, 마우스를 올리면 값을 바꾸지 않고 미리 보여 줍니다. Precision은 값을 글리프 단위나 반 단위로 반올림하거나, 3.7 같은 읽기 전용 평균을 위해 정확한 값 그대로 유지하여 일부만 채운 글리프로 그립니다.
VCL 클래스는 TsgcVCLCustomEditor를, FireMonkey 클래스는 TsgcFMXCustomEditor를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Value	Double	값입니다. 0부터 Count까지이며 Precision에 따라 반올림됩니다.
Count	Integer	글리프의 수. 1부터 20까지. 기본값은 5입니다.
Precision	TsgcEditorRatingPrecision	erpWhole(기본값)은 글리프 단위, erpHalf는 반 단위로 반올림하고, erpExact는 반올림하지 않습니다. Value보다 먼저 Count와 Precision을 설정하십시오.
Glyph	TsgcEditorRatingGlyph	ergStar(기본값), ergHeart, ergCircle 또는 OnDrawGlyph를 통해 그리는 ergCustom.
GlyphSize	Double	논리 픽셀 단위의 글리프 크기. 0(기본값)이면 컨트롤에 맞는 가장 큰 크기입니다.
Spacing	Double	두 글리프 사이의 논리 픽셀 수. 기본값은 4입니다.
FillColor	TsgcUIColor	채워진 글리프의 색상. 0(기본값)은 테마 팔레트의 호박색입니다.
EmptyColor	TsgcUIColor	빈 글리프의 색상. 0(기본값)은 테마의 테두리 색상입니다.
AllowClear	Boolean	현재 값을 클릭하면 0으로 지워지고, Home은 0으로 이동합니다. 기본값은 False이며, 이 경우 사용자가 선택할 수 있는 가장 낮은 값은 한 단계입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.
AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
RatingModel	TsgcRatingModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환 값	설명
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	값이 바뀌었습니다.
OnHover	TsgcRatingHoverEvent	(Sender; aValue: Double). 미리 보는 동안 마우스 아래의 값이며, 미리 보기가 끝나면 -1입니다.
OnDrawGlyph	TsgcRatingDrawGlyphEvent	(Sender; aCanvas: TsgcUICanvas; const aRect: TsgcUIRect; aFill: Double). ergCustom의 글리프를 aRect에 장치 픽셀로 그립니다. 빈 글리프에는 aFill 0으로 호출되고, 이어서 채워진 부분이 왼쪽부터 0에서 1의 비율로 호출됩니다.

키보드

Right와 Up은 정밀도의 한 단계를 더하고, Left와 Down은 한 단계를 뺍니다. Home은 가장 낮은 값으로, End는 Count로 이동하며, 숫자 키는 정수 값을 설정합니다.

예제: 직접 만든 글리프로 표시하는 10점 만점 점수

참고 항목

- DB 평점
- 범위 슬라이더
- 테마

스핀 편집기

클래스: TsgcVCLSpinEdit (유닛 sgcEditors_VCL_Value), TsgcFMXSpinEdit (유닛 sgcEditors_FMX_Value)
스핀 편집기는 위아래 버튼이 있는 숫자용 네이티브 편집 컨트롤입니다. 값은 Double이며, esvInteger는 정수로, esvFloat는 Decimals 자릿수로 반올림합니다. Units는 kg나 % 같은 흐린 접미사로, 텍스트 뒤에 그려지며 텍스트의 일부가 아닙니다. 버튼을 누르고 있으면 반복되며, 오래 누를수록 빨라집니다. 구문 분석은 관대하여, 공백과 천 단위 구분 기호는 무시되고 . 과 , 모두 소수점으로 받아들입니다.
VCL 클래스는 TsgcVCLCustomEditor를, FireMonkey 클래스는 TsgcFMXCustomEditor를 상속합니다. 둘 다 아래 표의 속성과 이벤트에 더해, Align, Anchors, Enabled, Hint, TabOrder, TabStop, OnEnter, OnExit, 키 및 마우스 이벤트처럼 각 프레임워크의 일반적인 것들을 게시합니다. 컨트롤을 참조하십시오.

속성

속성	형식	설명
Value	Double	값입니다. 먼저 ValueType, MinValue, MaxValue를 설정하십시오. 그렇지 않으면 값이 이전 제한 안으로 맞춰집니다.
MinValue, MaxValue	Double	제한입니다. 기본값 0과 0처럼 같은 값은 제한 없음을 뜻합니다.
Increment	Double	버튼, Up, Down, 휠의 단계. 기본값은 1입니다.
LargeIncrement	Double	PageUp과 PageDown의 단계. 기본값은 10입니다.
ValueType	TsgcEditorSpinValueType	esvInteger(기본값) 또는 esvFloat.
Decimals	Integer	esvFloat 값의 소수 자릿수. 기본값은 2입니다.
DisplayFormat	string	텍스트용 FormatFloat 마스크(예: #,##0.00). 비어 있으면(기본값) Decimals로 값을 씁니다.
Units	string	텍스트 뒤의 흐린 접미사이며, 값의 일부가 아닙니다.
Buttons	TsgcEditorSpinButtons	esbRight(기본값)는 위아래 버튼을 오른쪽에 쌓고, esbSides는 빼기를 왼쪽에 더하기를 오른쪽에 두며, esbNone은 버튼이 없습니다.
Wrap	Boolean	제한을 넘는 단계는 먼저 제한 값에서 멈추고, 다음 단계는 반대쪽 끝으로 이동합니다. 기본값은 False입니다.
AllowEmpty	Boolean	사용자가 텍스트를 지워 값을 비울 수 있게 합니다. 기본값은 False입니다.
IsEmpty	Boolean	public, 읽기 전용. 값이 비어 있는 동안 True입니다.
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	선택, 채우기, 포커스 링의 ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color. Color가 0이면 테마를 따릅니다.
ReadOnly	Boolean	값은 표시되지만 사용자가 변경할 수 없습니다. 기본값은 False입니다.

AccessibleName	string	화면 읽기 프로그램이 값 앞에 알리는 이름. 키보드와 접근성을 참조하십시오.
FormatSettings	TFormatSettings	public. 컨트롤이 사용하는 이름, 형식, 구분 기호입니다. 사용자 로캘의 설정으로 시작합니다. 현지화를 참조하십시오.
SpinModel	TsgcSpinModel	public, 읽기 전용. 컨트롤의 모델입니다. Model은 같은 객체를 TsgcEditorModel로 반환합니다.

메서드

메서드	반환값	설명
Clear		AllowEmpty가 True이면 값을 비웁니다. 그렇지 않으면 아무 일도 하지 않습니다.
SaveToPNG(aStream or aFileName, aScale = 1)		컨트롤을 현재 크기로 PNG에 그립니다. aScale은 크기를 곱합니다. 헤드리스 출력을 참조하십시오.
SaveToSVG(aStream or aFileName, aScale = 1)		같은 내용을 SVG로 출력합니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	새 값입니다. 확정되었거나, 단계로 변경되었거나, 코드로 설정되었습니다.
OnParseError	TsgcSpinParseErrorEvent	(Sender; const aText: string; var aValue: Double; var aAccept: Boolean). 확정된 텍스트가 숫자가 아닙니다. 값을 사용하려면 aValue를 설정하고 aAccept를 True로 설정하십시오. 그렇지 않으면 마지막 값으로 돌아갑니다.
OnValidate	TsgcSpinValidateEvent	(Sender; var aValue: Double; var aAccept: Boolean). 새 값이 설정되기 전에 호출됩니다. aAccept에는 True가 전달됩니다. aValue를 바꿔 조정하거나, aAccept를 False로 설정하여 이전 값을 유지하십시오.

키보드와 마우스

Up과 Down은 한 Increment, PageUp과 PageDown은 한 LargeIncrement, 휠은 한 칸당 한 Increment씩 이동합니다. 모두 먼저 입력한 텍스트를 확정합니다. Enter와 포커스 상실은 텍스트를 확정하고, Escape는 복원합니다. 버튼을 누르고 있으면 반복마다 한 Increment씩, 10번째 반복부터 2, 20번째부터 5, 30번째부터 10씩 이동합니다.

예제: 0.5도 단위의 각도

참고 항목

- DB 스핀 편집기
- 범위 슬라이더
- 현지화

DB 날짜 선택기

클래스: TsgcVCLDBDatePicker (유닛 sgcEditors_VCL_DB), TsgcFMXDBDatePicker (유닛 sgcEditors_FMX_DB)
DB 날짜 선택기는 날짜 선택기의 날짜를 날짜 또는 날짜 및 시간 필드에 바인딩합니다. TsgcVCLCustomDatePicker와 TsgcFMXCustomDatePicker를 상속하며, Date를 제외한 날짜 선택기의 속성을 게시합니다. 값이 곧 필드이기 때문입니다. 그 밖의 게시된 속성, 메서드, 이벤트는 날짜 선택기의 것입니다.
데이터 인식 컨트롤은 Registered 및 Trial 에디션에 포함됩니다. Community에서는 해당 유닛이 비어 있습니다. 이들은 DB 유닛만 사용하는 TsgcEditorFieldLink를 통해 바인딩하므로, FireMonkey 컨트롤에는 LiveBindings가 필요하지 않습니다.

속성

속성	형식	설명
DataSource	TDataSource	데이터셋의 데이터 소스.
DataField	string	값을 바인딩할 필드의 이름.
AllowEmpty	Boolean	기본값은 True이며, null 필드는 빈 텍스트를 표시하고 텍스트를 지우면 null을 기록합니다.
DisplayFormat, MinDate, MaxDate, DisabledDates, DisabledWeekDays, FirstDayOfWeek, ShowWeekNumbers, TextHint		날짜 선택기와 같습니다.
ReadOnly	Boolean	True이면 컨트롤을 읽기 전용으로 유지합니다. 필드를 수정할 수 없는 동안에도 읽기 전용입니다. 데이터셋이 없거나, 데이터셋 또는 필드가 읽기 전용이거나, 계산 필드인 경우입니다. 기본값은 False입니다.
Field	TField	public, 읽기 전용. 바인딩된 필드이며, 링크가 활성화되지 않은 동안은 nil입니다.

메서드

메서드	반환값	설명
SaveToPNG, SaveToSVG		날짜 선택기와 같습니다. 기본 컨트롤의 다른 메서드도 적용됩니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	사용자, 코드, 또는 레코드 이동으로 값이 바뀌었습니다.
OnDropDown, OnCloseUp, OnParseError, OnIsDateDisabled		날짜 선택기와 같습니다.

필드를 편집하는 방법

- 값은 레코드를 따릅니다. null 필드는 빈 텍스트를 표시합니다.

- 사용자가 변경하면 데이터셋이 편집 모드가 되고 값이 수정된 것으로 표시됩니다. 코드에서 값을 설정하는 것도 변경입니다. 데이터셋이 편집 모드로 들어갈 수 없으면 필드의 값이 돌아옵니다.
- Enter와 포커스 상실은 필드에 기록하고, 데이터셋은 게시 전에 보류 중인 값을 기록합니다. 값을 거부하는 필드는 예외를 발생시키며, TDBEdit처럼 포커스는 컨트롤에 남습니다.
- Escape는 필드의 값을 복원합니다.
- 날짜 선택기는 TDateTimeField의 시간 부분을 유지하고 날짜만 바꿉니다.

예제: 생년월일

참고 항목

- 날짜 선택기
- DB 시간 선택기
- 빠른 시작 데이터 인식
- TsgcEditorFieldLink

DB 시간 선택기

클래스: TsgcVCLDBTimePicker (유닛 sgcEditors_VCL_DB), TsgcFMXDBTimePicker (유닛 sgcEditors_FMX_DB)
DB 시간 선택기는 시간 선택기의 시간을 시간 또는 날짜 및 시간 필드에 바인딩합니다. TsgcVCLCustomDatePicker와 TsgcFMXCustomDatePicker를 상속하며, Time을 제외한 시간 선택기의 속성을 게시합니다. 값이 곧 필드이기 때문입니다. 그 밖의 게시된 속성, 메서드, 이벤트는 시간 선택기의 것입니다.
데이터 인식 컨트롤은 Registered 및 Trial 에디션에 포함됩니다. Community에서는 해당 유닛이 비어 있습니다. 이들은 DB 유닛만 사용하는 TsgcEditorFieldLink를 통해 바인딩하므로, FireMonkey 컨트롤에는 LiveBindings가 필요하지 않습니다.

속성

속성	형식	설명
DataSource	TDataSource	데이터셋의 데이터 소스.
DataField	string	값을 바인딩할 필드의 이름.
AllowEmpty	Boolean	기본값은 True이며, null 필드는 빈 텍스트를 표시하고 텍스트를 지우면 null을 기록합니다.
DisplayFormat, MinTime, MaxTime, ListInterval, TextHint		시간 선택기와 같습니다.
ReadOnly	Boolean	True이면 컨트롤을 읽기 전용으로 유지합니다. 필드를 수정할 수 없는 동안에도 읽기 전용입니다. 데이터셋이 없거나, 데이터셋 또는 필드가 읽기 전용이거나, 계산 필드인 경우입니다. 기본값은 False입니다.
Field	TField	public, 읽기 전용. 바인딩된 필드이며, 링크가 활성화되지 않은 동안은 nil입니다.

메서드

메서드	반환값	설명
SaveToPNG, SaveToSVG		시간 선택기와 같습니다. 기본 컨트롤의 다른 메서드도 적용됩니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	사용자, 코드, 또는 레코드 이동으로 값이 바뀌었습니다.
OnDropDown, OnCloseUp, OnParseError		시간 선택기와 같습니다.

필드를 편집하는 방법

- 값은 레코드를 따릅니다. null 필드는 빈 텍스트를 표시합니다.

- 사용자가 변경하면 데이터셋이 편집 모드가 되고 값이 수정된 것으로 표시됩니다. 코드에서 값을 설정하는 것도 변경입니다. 데이터셋이 편집 모드로 들어갈 수 없으면 필드의 값이 돌아옵니다.
- Enter와 포커스 상실은 필드에 기록하고, 데이터셋은 게시 전에 보류 중인 값을 기록합니다. 값을 거부하는 필드는 예외를 발생시키며, TDBEdit처럼 포커스는 컨트롤에 남습니다.
- Escape는 필드의 값을 복원합니다.
- 시간 선택기는 TDateTimeField의 날짜 부분을 유지하고 시간만 바꿉니다.

예제: 예약의 시작 시간

참고 항목

- 시간 선택기
- DB 날짜 선택기
- 빠른 시작 데이터 인식
- TsgcEditorFieldLink

DB 스핀 편집기

클래스: TsgcVCLDBSpinEdit (유닛 sgcEditors_VCL_DB), TsgcFMXDBSpinEdit (유닛 sgcEditors_FMX_DB)
DB 스핀 편집기는 스핀 편집기의 값을 숫자 필드에 바인딩합니다. TsgcVCLSpinEdit와 TsgcFMXSpinEdit를 상속하므로 제한, 단계, 단위, 버튼을 유지합니다. 정수 필드이면 esvInteger로 전환됩니다. 그 밖의 게시된 속성, 메서드, 이벤트는 스핀 편집기의 것입니다.
데이터 인식 컨트롤은 Registered 및 Trial 에디션에 포함됩니다. Community에서는 해당 유닛이 비어 있습니다. 이들은 DB 유닛만 사용하는 TsgcEditorFieldLink를 통해 바인딩하므로, FireMonkey 컨트롤에는 LiveBindings가 필요하지 않습니다.

속성

속성	형식	설명
DataSource	TDataSource	데이터셋의 데이터 소스.
DataField	string	값을 바인딩할 필드의 이름.
AllowEmpty	Boolean	기본값은 True이며, null 필드는 빈 텍스트를 표시하고 텍스트를 지우면 null을 기록합니다.
Value	Double	필드의 값. 폼에는 저장되지 않습니다.
ReadOnly	Boolean	True이면 컨트롤을 읽기 전용으로 유지합니다. 필드를 수정할 수 없는 동안에도 읽기 전용입니다. 데이터셋이 없거나, 데이터셋 또는 필드가 읽기 전용이거나, 계산 필드인 경우입니다. 기본값은 False입니다.
Field	TField	public, 읽기 전용. 바인딩된 필드이며, 링크가 활성화되지 않은 동안은 nil입니다.

메서드

메서드	반환값	설명
SaveToPNG, SaveToSVG		스핀 편집기와 같습니다. 기본 컨트롤의 다른 메서드도 적용됩니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	사용자, 코드, 또는 레코드 이동으로 값이 바뀌었습니다.
OnParseError, OnValidate		스핀 편집기와 같습니다. OnValidate는 값이 필드로 가기 전에 실행됩니다.

필드를 편집하는 방법

- 값은 레코드를 따릅니다. null 필드는 빈 텍스트를 표시합니다.
- 사용자가 변경하면 데이터셋이 편집 모드가 되고 값이 수정된 것으로 표시됩니다. 코드에서 값을 설정하는 것도 변경입니다. 데이터셋이 편집 모드로 들어갈 수 없으면 필드의 값이 돌아옵니다.
- Enter와 포커스 상실은 필드에 기록하고, 데이터셋은 게시 전에 보류 중인 값을 기록합니다. 값을 거부하는 필드는 예외를 발생시키며, TDBEdit처럼 포커스는 컨트롤에 남습니다.
- Escape는 필드의 값을 복원합니다.

- 정수 필드이면 ValueType이 esvInteger로 전환됩니다.

예제: 수량

예제: FireMonkey의 할인

참고 항목

- 스핀 편집기
- DB 평점
- 빠른 시작 데이터 인식
- TsgcEditorFieldLink

DB 평점

클래스: TsgcVCLDBRating (유닛 sgcEditors_VCL_DB), TsgcFMXDBRating (유닛 sgcEditors_FMX_DB)
DB 평점은 평점의 값을 숫자 필드에 바인딩합니다. TsgcVCLRating과 TsgcFMXRating을 상속하므로 Count, Precision, 글리프, 색상을 유지합니다. Precision이 erpExact이고 ReadOnly가 True이면 계산 필드나 집계 필드에서 3.7 같은 평균을 표시합니다. 그 밖의 게시된 속성, 메서드, 이벤트는 평점의 것입니다.
데이터 인식 컨트롤은 Registered 및 Trial 에디션에 포함됩니다. Community에서는 해당 유닛이 비어 있습니다. 이들은 DB 유닛만 사용하는 TsgcEditorFieldLink를 통해 바인딩하므로, FireMonkey 컨트롤에는 LiveBindings가 필요하지 않습니다.

속성

속성	형식	설명
DataSource	TDataSource	데이터셋의 데이터 소스.
DataField	string	값을 바인딩할 필드의 이름.
Value	Double	필드의 값. 폼에는 저장되지 않습니다.
ReadOnly	Boolean	True이면 컨트롤을 읽기 전용으로 유지합니다. 필드를 수정할 수 없는 동안에도 읽기 전용입니다. 데이터셋이 없거나, 데이터셋 또는 필드가 읽기 전용이거나, 계산 필드인 경우입니다. 기본값은 False입니다.
Field	TField	public, 읽기 전용. 바인딩된 필드이며, 링크가 활성화되지 않은 동안은 nil입니다.

메서드

메서드	반환값	설명
SaveToPNG, SaveToSVG		평점과 같습니다. 기본 컨트롤의 다른 메서드도 적용됩니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	사용자, 코드, 또는 레코드 이동으로 값이 바뀌었습니다.
OnHover, OnDrawGlyph		평점과 같습니다.

필드를 편집하는 방법

- 값은 레코드를 따릅니다. null 필드는 글리프를 채우지 않습니다.
- 사용자가 변경하면 데이터셋이 편집 모드가 되고 값이 수정된 것으로 표시됩니다. 코드에서 값을 설정하는 것도 변경입니다. 데이터셋이 편집 모드로 들어갈 수 없으면 필드의 값이 돌아옵니다.
- Enter와 포커스 상실은 필드에 기록하고, 데이터셋은 게시 전에 보류 중인 값을 기록합니다. 값을 거부하는 필드는 예외를 발생시키며, TDBEdit처럼 포커스는 컨트롤에 남습니다.
- Escape는 필드의 값을 복원합니다.
- AllowClear를 사용하면 현재 값을 클릭할 때 필드에 0을 기록합니다.

예제: 별과 평균

참고 항목

- 평점
- DB 스핀 편집기
- 빠른 시작 데이터 인식
- TsgcEditorFieldLink

DB 범위 슬라이더

클래스: TsgcVCLDBRangeSlider (유닛 sgcEditors_VCL_DB), TsgcFMXDBRangeSlider (유닛 sgcEditors_FMX_DB)
DB 범위 슬라이더는 범위 슬라이더의 Value를 DataField에, RangeMode에서는 EndValue를 EndDataField에 바인딩하므로, 컨트롤 하나로 두 필드에 저장된 범위를 편집할 수 있습니다. TsgcVCLRangeSlider와 TsgcFMXRangeSlider를 상속합니다. 그 밖의 게시된 속성, 메서드, 이벤트는 범위 슬라이더의 것입니다.
데이터 인식 컨트롤은 Registered 및 Trial 에디션에 포함됩니다. Community에서는 해당 유닛이 비어 있습니다. 이들은 DB 유닛만 사용하는 TsgcEditorFieldLink를 통해 바인딩하므로, FireMonkey 컨트롤에는 LiveBindings가 필요하지 않습니다.

속성

속성	형식	설명
DataSource	TDataSource	데이터셋의 데이터 소스.
DataField	string	값을 바인딩할 필드의 이름.
EndDataField	string	RangeMode에서 EndValue의 필드.
Value, EndValue	Double	두 필드의 값. 폼에는 저장되지 않습니다.
EndField	TField	public, 읽기 전용. EndValue의 필드이며, 링크가 활성화되지 않은 동안은 nil입니다.
ReadOnly	Boolean	True이면 컨트롤을 읽기 전용으로 유지합니다. 필드를 수정할 수 없는 동안에도 읽기 전용입니다. 데이터셋이 없거나, 데이터셋 또는 필드가 읽기 전용이거나, 계산 필드인 경우입니다. 기본값은 False입니다.
Field	TField	public, 읽기 전용. 바인딩된 필드이며, 링크가 활성화되지 않은 동안은 nil입니다.

메서드

메서드	반환값	설명
SaveToPNG, SaveToSVG		범위 슬라이더와 같습니다. 기본 컨트롤의 다른 메서드도 적용됩니다.

이벤트

이벤트	형식	설명
OnChange	TNotifyEvent	사용자, 코드, 또는 레코드 이동으로 값이 바뀌었습니다.
OnChangeComplete, OnFormatLabel		범위 슬라이더와 같습니다.

필드를 편집하는 방법

- 값은 레코드를 따릅니다. null 필드는 핸들을 Min에, null 끝 필드는 위쪽 핸들을 Max에 둡니다.
- 사용자가 변경하면 데이터셋이 편집 모드가 되고 값이 수정된 것으로 표시됩니다. 코드에서 값을 설정하는 것도 변경입니다. 데이터셋이 편집 모드로 들어갈 수 없으면 필드의 값이 돌아옵니다.

- Enter와 포커스 상실은 필드에 기록하고, 데이터셋은 게시 전에 보류 중인 값을 기록합니다. 값을 거부하는 필드는 예외를 발생시키며, TDBEdit처럼 포커스는 컨트롤에 남습니다.
- Escape는 필드의 값을 복원합니다.
- RangeMode에서는 끝 필드를 수정할 수 없는 동안 컨트롤이 읽기 전용입니다.

예제: 두 필드에 저장된 가격 범위

참고 항목

- 범위 슬라이더
- DB 평점
- 빠른 시작 데이터 인식
- TsgcEditorFieldLink

TsgcEditorModel

유닛: sgcEditors_Types
TsgcEditorModel은 sgcEditors의 모든 모델의 기본 클래스입니다. RTL과 sgcUICore만 사용하는 TPersistent이므로 VCL, FireMonkey, 창 없이도, Windows와 Linux64에서 동작합니다. 모델은 값과 옵션을 담고, 키와 마우스를 받으며, 두 레이어로 자신을 그립니다. Render는 콘텐츠 레이어를, RenderOverlay는 호버, 포커스 링, 드래그 중인 부분을 그립니다. 모든 VCL 및 FireMonkey 컨트롤은 모델 하나를 호스트하며, Model과 형식이 지정된 속성으로 public 노출합니다.

모델	유닛	사용처	컨트롤의 속성
TsgcCalendarModel	sgcEditors_Calendar	월 달력과 날짜 선택기의 드롭다운.	CalendarModel, DropDownCalendar
TsgcDatePickerModel	sgcEditors_Picker	날짜, 시간, 날짜 및 시간 선택기.	PickerModel
TsgcDateTimeDropDownModel	sgcEditors_Picker	날짜 및 시간 선택기의 드롭다운: 달력과 시간 목록을 나란히 배치한 것.	
TsgcEditorListModel	sgcEditors_List	시간 목록과 토큰 편집기의 자동 완성 목록.	DropDownTimeList, TokenModel.AutoCompleteList
TsgcSliderModel	sgcEditors_Slider	범위 슬라이더.	SliderModel
TsgcColorPickerModel	sgcEditors_Color	색상 선택기.	ColorModel
TsgcTokenModel	sgcEditors_Token	토큰 편집기.	TokenModel
TsgcRatingModel	sgcEditors_Rating	평점.	RatingModel
TsgcSpinModel	sgcEditors_Spin	스핀 편집기.	SpinModel

모델은 Date, Value, Tokens처럼 컨트롤과 같은 값 속성에 더해, HitTest와 각 부분의 사각형처럼 호스트용 멤버를 게시합니다.

속성

속성	형식	설명
Theme	TsgcUITheme	utLight(기본값) 또는 utDark. 테마를 참조하십시오.
AccentColor	TsgcUIColor	ARGB 강조 색상. 0(기본값)이면 테마의 강조 색상을 사용합니다.
TextStyle	TsgcUITextStyle	텍스트의 Size, Bold, Color.
Enabled	Boolean	비활성화된 모델은 입력을 받지 않으며 비활성 색상으로 그립니다. 기본값은 True입니다.

ReadOnly	Boolean	모델은 값을 바꾸는 입력을 받지 않습니다. 기본값은 False입니다.
Focused	Boolean	public. 오버레이에 포커스 링을 그립니다. 호스트는 컨트롤이 포커스를 얻거나 잃을 때 설정합니다.
HotPart	Integer	public. 마우스 아래의 부분이며, 없으면 0입니다. 각 모델은 자체적으로 부분에 번호를 매깁니다.
Painter	TsgcEditorsPainter	public, 읽기 전용. 마지막 렌더의 테마 색상과 프레임, 포커스 링, 쉐브론, 스핀 화살표 그리기.

메서드

메서드	반환값	설명
BeginUpdate, EndUpdate		여러 변경을 묶습니다. 바깥쪽 쌍의 EndUpdate가 보류 중인 각 종류의 변경을 한 번씩 보냅니다.
IsUpdating	Boolean	BeginUpdate와 EndUpdate 사이에서는 True입니다.
KeyDown(var aKey: Word; aShift)	Boolean	키로, Windows 가상 키 코드입니다. 코드는 CS_EDITORS_KEY_* 상수에 있으며 FireMonkey도 같은 값을 사용합니다. 모델이 처리했으면 True입니다.
KeyPress(var aKey: Char)	Boolean	문자입니다. 모델이 처리했으면 True입니다.
MouseDown, MouseUp(aButton, aShift, aX, aY)	Boolean	마우스 버튼(embLeft, embRight, embMiddle)이며, aX와 aY는 논리 픽셀입니다. 모델이 처리했으면 True입니다.
MouseMove(aShift, aX, aY), MouseWheel(aShift, aWheelDelta, aX, aY), MouseLeave	Boolean	포인터와 휠(논리 픽셀). 휠 한 칸은 120입니다.
Render(aCanvas, aWidth, aHeight, aScale = 1)		임의의 TsgcUICanvas에 콘텐츠 레이어를 그립니다. aWidth와 aHeight는 장치 픽셀입니다.
RenderOverlay(aCanvas, aWidth, aHeight, aScale = 1)		콘텐츠 레이어 위에 오버레이를 그립니다.
SaveToPNG(aStream or aFileName, aWidth, aHeight, aScale = 1)		두 레이어를 PNG에 그립니다. aWidth와 aHeight는 논리 픽셀이며 aScale이 이를 곱합니다.
SaveToSVG(aStream or aFileName, aWidth, aHeight, aScale = 1)		같은 내용을 SVG로 출력합니다.
MeasureText(aText, aFont)	Double	텍스트의 너비(장치 픽셀). OnMeasure가 할당되어 있으면 그것을, 아니면 sgcUICore에 내장된 글꼴을 사용합니다.
Assign(aSource)		같은 클래스의 다른 모델의 옵션과 값을 복사합니다.

이벤트

이벤트	형식	설명
OnModelChange	TsgcEditorModelChangeEvent	(Sender; aKind: TsgcEditorChangeKind). eckContent: 콘텐츠 레이어를 다시 그립니다. eckOverlay: 오버레이만

		다시 그립니다. eckValue: 값이 바뀌었습니다. eckLayout: 부분이 이동했으므로 내부 편집 컨트롤을 다시 배치합니다.
OnMeasure	TsgcEditorMeasureEvent	function (const aText: string; const aFont: TsgcUIFontStyle): Double. 호스트의 글꼴로 텍스트를 측정합니다(장치 픽셀).

컨트롤은 자신의 모델의 OnModelChange와 OnMeasure를 할당합니다. 직접 만든 모델에만 할당하십시오.

크기와 입력

모델은 마지막 Render의 크기 또는 SetViewSize에 전달한 크기(TsgcCalendarModel에서는 Width와 Height)로 논리 픽셀을 사용해 레이아웃과 히트 테스트를 수행합니다. 직접 호스트하는 모델에 마우스 입력을 보내기 전에 Render를 한 번 실행하거나 크기를 지정하십시오.

예제: 창 없이 슬라이더 모델 구동하기

참고 항목

- 아키텍처
- 헤드리스 출력
- TsgcEditorLocale

TsgcEditorLocale

유닛: sgcEditors_Date
TsgcEditorLocale은 날짜 모델이 그리고 구문 분석하는 데 사용하는 요일 및 월 이름, 날짜 및 시간 형식, 구분 기호, 한 주의 첫 요일을 담습니다. 컨트롤은 자체 로캘을 유지하고 FormatSettings에서 채우므로, 폼에서는 FormatSettings를 설정합니다. 현지화를 참조하십시오. TsgcEditorLocale은 직접 만드는 모델(헤드리스 TsgcCalendarModel 등) 및 유닛의 날짜 함수와 함께 사용합니다.
TsgcCalendarModel과 TsgcDatePickerModel에는 Locale 속성이 있습니다. 모델은 로캘을 소유하지 않으며, nil(기본값)이면 처음 사용할 때 사용자 로캘에서 가져온 공유 로캘인 sgcEditorDefaultLocale을 사용합니다. 로캘을 먼저 채우고 나중에 할당하십시오. 모델은 Locale이 할당될 때 다시 그려지며, 로캘의 속성이 바뀔 때는 다시 그려지지 않기 때문입니다.

속성

속성	형식	설명
ShortDayNames[aDay], LongDayNames[aDay]	string	TsgcEditorWeekDay의 이름(ewdMonday부터 ewdSunday까지).
ShortMonthNames[aMonth], LongMonthNames[aMonth]	string	월의 이름(1부터 12까지).
ShortDateFormat, LongDateFormat	string	날짜 형식. 날짜 선택기의 DisplayFormat이 비어 있으면 ShortDateFormat을 사용합니다.
ShortTimeFormat, LongTimeFormat	string	시간 형식. 시간 선택기의 DisplayFormat이 비어 있으면 ShortTimeFormat을 사용합니다.
TimeAMString, TimePMString	string	12시간제 형식의 AM 및 PM 텍스트.
DateSeparator, TimeSeparator	Char	형식에 / 와 : 가 있는 곳에 쓰입니다.
FirstDayOfWeek	TsgcEditorWeekDay	efdLocale이 나타내는 한 주의 첫 요일.

메서드

메서드	반환 값	설명
Create		로캘을 만들고 Capture를 호출합니다.
Capture		이름, 형식, 구분 기호는 RTL에서, 한 주의 첫 요일은 운영 체제에서 가져오며, 없으면 월요일입니다.
CaptureSettings(aSettings: TFormatSettings)		이름, 형식, 구분 기호를 TFormatSettings.Create('de-DE') 같은 aSettings에서 가져옵니다. FirstDayOfWeek는 유지됩니다.
Assign(aSource)		다른 로캘을 복사합니다.

함수

함수	반환값	설명

sgcEditorDefaultLocale	TsgcEditorLocale	모델의 Locale이 nil일 때 사용하는 공유 로캘. 해제하지 마십시오.
sgcEditorFormatDateTime(aDateTime, aFormat, aLocale = nil)	string	DisplayFormat과 로캘의 이름으로 날짜와 시간을 씁니다.
sgcEditorParseDateTime(aText, aFormat, aReference, out aDateTime, aLocale = nil)	Boolean	선택기의 관대한 구문 분석. 텍스트에 없는 필드는 aReference에서 가져옵니다.
sgcEditorFirstDayOf(aFirstDay, aLocale = nil)	TsgcEditorWeekDay	TsgcEditorFirstDayOfWeek를 해석합니다. efdLocale이면 로캘의 FirstDayOfWeek가 됩니다.

이벤트

TsgcEditorLocale에는 이벤트가 없습니다.

예제: 서버의 독일어 달력

참고 항목

- 현지화
- 월 달력
- TsgcEditorModel

TsgcEditorFieldLink

유닛: sgcEditors_Data
TsgcEditorFieldLink는 데이터 인식 컨트롤의 TDataLink입니다. DB 유닛만 사용하므로 VCL과 FireMonkey 컨트롤이 함께 사용하며, FireMonkey에는 TFieldDataLink가 없습니다. TFieldDataLink의 동작을 갖추고 있으므로 직접 만든 컨트롤을 필드에 바인딩하는 데에도 사용할 수 있습니다. Registered 및 Trial 에디션에 포함되며, Community에서는 유닛이 비어 있습니다.

속성

속성	형식	설명
DataSource	TDataSource	TDataLink에서 상속. 데이터셋의 데이터 소스.
FieldName	string	필드의 이름. 링크가 활성화될 때와 데이터셋의 필드가 바뀔 때 FieldByName으로 확인됩니다.
Field	TField	읽기 전용. 필드이며, 링크가 활성화되지 않았거나 FieldName이 비어 있으면 nil입니다.
ReadOnly	Boolean	TDataLink에서 상속. True이면 링크가 편집하지 않습니다.
CanModify	Boolean	읽기 전용. 링크가 ReadOnly가 아니고, 필드가 있으며, TField.CanModify가 True입니다. 즉 필드가 읽기 전용이나 계산 필드가 아니고 데이터셋이 수정할 수 있습니다.
Editing	Boolean	읽기 전용. 데이터셋이 편집 상태이고 필드를 수정할 수 있는 동안 True입니다.
IsModified	Boolean	읽기 전용. Modified 이후, UpdateRecord가 필드에 기록하거나, OnDataChange가 다시 불러오거나, Editing이 바뀔 때까지 True입니다.

메서드

메서드	반환값	설명
Edit	Boolean	필드를 수정할 수 있으면 데이터 소스를 통해 데이터셋을 편집 모드로 만듭니다. 이후 링크가 편집 중이면 True입니다.
Modified		컨트롤의 값을 변경된 것으로 표시합니다.
Reset		OnDataChange를 통해 필드에서 컨트롤을 다시 불러오고 IsModified를 지웁니다.
UpdateRecord		TDataLink에서 상속. IsModified이면 OnUpdateData를 발생시키므로 컨트롤이 필드에 한 번만 기록합니다. 데이터셋은 게시 전에 이를 호출합니다.

이벤트

이벤트	형식	설명
OnDataChange	TNotifyEvent	레코드 또는 필드가 바뀌었습니다. 스크롤, 취소, 게시, 새로 고침 또는 Reset입니다. 필드의 값을 표시하십시오. Edit가 데이터셋을 편집 모드로 만드는 동

		안에는 컨트롤이 사용자의 보류 중인 변경을 가지고 있으므로 발생하지 않습니다.
OnUpdateData	TNotifyEvent	컨트롤의 값을 Field에 기록합니다.
OnActiveChange	TNotifyEvent	데이터셋이 열리거나 닫혔습니다.
OnEditingChange	TNotifyEvent	Editing이 바뀌었습니다. IsModified가 지워집니다.

예제: 직접 만든 컨트롤 바인딩

참고 항목

- 빠른 시작 데이터 인식
- DB 평점
- TsgcEditorModel

텍스트 입력과 IME

sgcEditors의 모든 텍스트 필드는 실제 편집 컨트롤입니다. 선택기의 날짜 또는 시간 텍스트, 스핀 편집기의 숫자, 색상 선택기의 16진수 필드, 토큰 편집기의 마지막 칩 뒤의 입력란입니다. 월 달력, 범위 슬라이더, 평점에는 텍스트가 없으며, 스스로 포커스를 받는 그려지는 컨트롤입니다.

네이티브 편집 컨트롤을 쓰는 이유

내부 편집 컨트롤은 VCL에서는 테두리 없는 TEdit 자식이고 FireMonkey에서는 FMX TEdit이며, 모델이 텍스트를 그리는 위치에 놓입니다. 플랫폼의 편집 컨트롤이므로 일본어, 중국어, 한국어 입력의 IME 입력, 클립보드, 실행 취소, 컨텍스트 메뉴, 화면 읽기 프로그램이 특별한 코드 없이 모두 동작합니다. 그 둘레의 프레임, 버튼, 단위, 칩은 모델이 그립니다. VCL에서 편집 컨트롤은 단독 탭 정지가 되지 않습니다. 컨트롤이 포커스를 그곳으로 전달하며, 편집 컨트롤이 포커스를 가진 동안 포커스 링이 그려집니다.
Windows에서는 각 컨트롤의 래스터 캔버스가 플랫폼 글꼴 파일을 한 번만 불러오고 내부 편집 컨트롤이 같은 글꼴과 픽셀 크기를 사용하므로, 입력한 텍스트와 그려진 텍스트가 일치합니다. 다른 플랫폼과 헤드리스 출력은 sgcUICore에 내장된 글꼴인 Roboto를 사용합니다.

날짜나 시간 입력

- 텍스트는 DisplayFormat을 따르며, 세그먼트 d, dd, ddd, dddd, m, mm, mmm, mmmm, yy, yyyy, h, hh, n, nn, s, ss, am/pm, ampm과 리터럴로 구성됩니다. `/`는 로캘의 날짜 구분 기호를, `:`는 시간 구분 기호를 씁니다. DisplayFormat이 비어 있으면 컨트롤의 FormatSettings에서 짧은 날짜, 짧은 시간 또는 둘 다를 사용합니다.
- 구문 분석은 관대합니다. `/ - . : ,` 중 하나 또는 공백이 필드를 구분하고, 로캘의 월 이름과 요일 이름을 이해하며, 두 자리 연도는 50을 기준으로 해석하고, 맨 앞의 네 자리 연도는 연, 월, 일로 읽습니다. 텍스트에 없는 필드는 연도를 오늘에서, 시간을 0으로 가져오며, 시간 필드가 없는 형식은 현재 값의 시간을 유지합니다.
- 선택기는 변경할 때마다 텍스트를 구문 분석하고 열려 있는 달력을 그에 맞춰 이동합니다. 값은 Enter, 포커스 상실, 드롭다운을 닫을 때 확정됩니다.
- Up과 Down은 캐럿 위치의 세그먼트를 증감하고 다시 선택하므로 월이나 시를 돌릴 수 있습니다. 필드는 Windows의 날짜 및 시간 선택기처럼 순환하며, 결과는 MinDate, MaxDate, MinTime, MaxTime으로 제한됩니다.
- 구문 분석할 수 없는 텍스트는 텍스트, 값, False로 설정된 aAccept와 함께 OnParseError를 발생시킵니다. 텍스트를 사용하려면 aValue와 aAccept를 설정하고, 그대로 두면 마지막 유효한 값으로 돌아갑니다. 제한 밖이거나 달력에서 비활성화된 날짜는 확정 시 거부됩니다.
- AllowEmpty를 True로 설정하면 빈 텍스트가 값을 지우고 IsEmpty가 True가 됩니다. TextHint는 편집 컨트롤이 비어 있는 동안 표시됩니다. VCL에서는 Delphi 2009부터, FireMonkey에서는 편집 컨트롤의 TextPrompt로 표시됩니다.

숫자 입력

스핀 편집기도 관대하게 구문 분석합니다. 공백과 로캘의 천 단위 구분 기호는 무시되며 `.`와 `,` 모두 소수점으로 받아들입니다. 텍스트는 DisplayFormat, FormatFloat 마스크, 또는 DisplayFormat이 비어 있으면 Decimals로 표기됩니다. Units는 텍스트 뒤에 그려지는 흐린 접미사이며 텍스트의 일부가 아닙니다. OnParseError는 선택기와 같게 동작하며, OnValidate는 aAccept가 True로 전달된 상태에서 모든 새 값을 바꾸거나 거부할 수 있게 합니다.

토큰 입력

- Delimiters(기본값은 쉼표와 세미콜론)의 각 문자가 토큰을 끝내며, Enter는 항상 그렇습니다. Tab과 포커스 상실도 입력한 텍스트를 확정합니다. `a, b, c`를 붙여넣으면 칩 세 개가 됩니다.
- OnValidateToken은 공백이 제거된 텍스트를 받습니다. 토큰을 정규화하려면 바꾸고, 거부하려면 aValid를 지우십시오. InvalidTokens가 etiReject이면 토큰이 추가되지 않고, etiMark이면 추가되어 잘못된 것으로 그려집니다.
- AllowDuplicates, CaseSensitive, MaxTokens는 추가할 수 있는 것을 제한합니다. AddToken, RemoveToken, Clear는 코드에서 토큰을 바꾸며, OnTokenAdded와 OnTokenRemoved는 모든 변경을 알려 줍니다.

예제

참고 항목

- 키보드와 접근성
- 현지화
- 날짜 선택기
- 토큰 편집기

키보드와 접근성

sgcEditors의 어떤 기능도 마우스를 필요로 하지 않습니다. 그려지는 컨트롤은 스스로 포커스를 받고, 텍스트가 있는 컨트롤은 포커스를 내부 편집 컨트롤로 전달합니다. 포커스된 컨트롤은 오버레이 레이어에 강조 색상의 2픽셀 포커스 링을 그립니다.

키

컨트롤	키
월 달력	화살표 키는 하루 또는 한 주씩 이동하며 비활성화된 날은 건너뜁니다. PageUp과 PageDown은 한 달씩, Ctrl을 누르면 1년씩 이동합니다. Enter 또는 Space로 선택합니다. Shift와 화살표 키로 범위를 넓힙니다. Ctrl+Up은 연도 및 10년 보기로 축소하며, Enter로 다시 확대합니다.
선택기	F4 또는 Alt+Down은 드롭다운을 열고 닫습니다. Up과 Down은 캐럿 위치의 세그먼트를 증감하거나 열려 있는 드롭다운 안에서 이동합니다. Enter는 확정하고, Escape는 드롭다운을 닫고 값을 복원합니다.
스핀 편집기	Up과 Down은 Increment씩, PageUp과 PageDown은 LargeIncrement씩 증감합니다. 마우스 휠은 Increment씩 증감하며, 누르고 있는 버튼은 오래 누를수록 더 빠르게 반복됩니다.
범위 슬라이더	화살표 키는 Step씩, PageUp과 PageDown은 LargeStep씩 변경하고, Home과 End는 Min과 Max로 이동합니다. 범위 모드는 아래의 포커스 모델을 참조하십시오.
평점	화살표 키는 Precision의 한 단계씩 값을 바꾸고, Home과 End는 가장 낮은 값과 Count로 이동하며, 숫자 키는 그 값을 설정합니다. AllowClear를 사용하면 0 또는 현재 값을 다시 선택하여 지웁니다.
색상 선택기	화살표 키는 견본 사이를 이동하거나 사각형과 막대의 커서를 움직이고, Enter 또는 Space는 선택하며, Home과 End는 양 끝으로 이동합니다. 16진수 편집 컨트롤은 #RGB, #RGBA, #RRGGBB, #RRGGBBAA 형식의 텍스트를 받습니다.
토큰 편집기	Enter 또는 구분 문자는 입력한 토큰을 추가합니다. 빈 입력에서 Backspace를 누르면 마지막 칩에 포커스가 가고, Backspace를 한 번 더 누르면 제거됩니다. Left와 Right는 칩을 이동하고, Delete는 포커스된 칩을 제거하며, Escape는 입력으로 돌아갑니다. Up과 Down은 자동 완성 목록 안에서 이동합니다.

포커스 모델

대부분의 컨트롤은 탭 정지가 하나입니다. 두 컨트롤은 두 부분으로 되어 있으며, FireMonkey는 컨트롤이 키를 보기 전에 Tab으로 포커스를 이동하므로 VCL과 FireMonkey가 다르게 접근합니다.

컨트롤	VCL	FireMonkey
범위 모드의 범위 슬라이더	Tab은 아래쪽 핸들에서 위쪽 핸들로 이동하며, 다음 Tab은 컨트롤을 벗어납니다. Shift+Tab은 위쪽 핸들에 도착합니다. Ctrl+Left 또는 Ctrl+Down은 아래쪽 핸들을, Ctrl+Right 또는 Ctrl+Up은 위쪽 핸들을 선택합니다.	탭 정지가 하나입니다. 포커스는 아래쪽 핸들에서 시작하며, 클릭하면 잡은 핸들이 유지됩니다. Ctrl+Left 또는 Ctrl+Down은 아래쪽 핸들을, Ctrl+Right 또는 Ctrl+Up은 위쪽 핸들을 선택합니다.
색상 선택기	탭 정지가 둘입니다. 각 부분, 그다음 16진수 편집 컨트롤입니다. 각 부분에서 Enter는 색상을 선택하며 폼의 기본 버튼에는 전달되지 않습니다. 읽기 전용 선택기는 탭 정지가 하나입니다.	탭 정지 하나에 내부 정지 둘입니다. Ctrl과 화살표 키로 각 부분과 16진수 편집 컨트롤 사이를 이동합니다.

FireMonkey에서는 포커스가 프레임워크에 의해 대기열에 들어가며, 포커스가 컨트롤과 그 내부 편집 컨트롤 사이를 이동해도 OnEnter와 OnExit는 컨트롤당 한 번만 발생합니다.

접근성

- 내부 편집 컨트롤은 네이티브 편집 컨트롤이므로 화면 읽기 프로그램이 그 텍스트를 평소처럼 읽고 편집합니다.
- 그려지는 VCL 컨트롤은 MSAA 창 프록시를 통해 WM_GETTEXT에 `AccessibleName: value`로 응답하며, 값이 바뀔 때마다 EVENT_OBJECT_VALUECHANGE로 NotifyWinEvent를 호출하므로, 화면 읽기 프로그램이 새 슬라이더 위치나 평점을 알려 줍니다.
- FireMonkey 컨트롤은 같은 텍스트를 Hint에 넣으며, 직접 지정한 Hint도 유지합니다.
- 두 기본 제공 테마의 텍스트는 대비가 4.5:1 이상이며, 사용자 지정 강조 색상에서 파생되는 색상은 이를 유지하도록 조정됩니다. QA가 이를 검사합니다.

예제

참고 항목

- 텍스트 입력과 IME
- 테마
- 범위 슬라이더
- 색상 선택기

테마

모든 컨트롤에는 Theme, AccentColor, TextStyle이 있습니다. 프레임, 호버, 눌린 상태, 선택, 비활성 상태의 색상은 TsgcEditorsPainter가 이로부터 파생하므로, 속성 하나로 컨트롤 전체의 모양이 바뀝니다.

속성	효과
Theme	utLight(기본값) 또는 utDark. sgcUICore의 라이트와 다크 색상 세트입니다. utCustom은 라이트 세트를 사용합니다.
AccentColor	선택, 핸들, 채우기, 포커스 링의 색상으로, $FF2A78D6 같은 ARGB TsgcUIColor입니다. 0은 테마의 강조 색상을 사용하며, 알파가 없는 색상은 불투명으로 간주합니다.
TextStyle	텍스트의 Size(기본값 11), Bold, Color. Color가 0이면 테마의 텍스트 색상을 따릅니다.
Enabled	비활성화된 컨트롤은 표면, 텍스트, 테두리, 강조 색상을 배경 쪽으로 섞습니다.

파생 색상

호버와 눌린 상태의 채우기는 표면 위에 강조 색상을 옅게 섞은 것입니다. 선택된 날 같은 강조 채우기 위의 텍스트는 더 잘 읽히는 어두운 색 또는 밝은 색이며, 그 텍스트가 4.5:1의 대비에 이를 때까지 채우기를 어둡게 또는 밝게 합니다. 범위의 선택 띠는 일반 텍스트와 같은 대비를 유지합니다. 그래서 어떤 강조 색상도 두 테마 모두에서 읽기 쉬운 상태로 남습니다.

컨트롤 색상

- 평점에는 FillColor와 EmptyColor가 있습니다. FillColor가 0이면 호박색이고 EmptyColor가 0이면 테마를 따릅니다. Glyph는 ergStar, ergHeart, ergCircle 또는 OnDrawGlyph를 통해 그리는 ergCustom을 선택합니다.
- 색상 선택기는 Swatches를 표시하며, 기본값은 sgcUICore 팔레트와 회색에서 가져온 24색입니다.

토큰 칩의 너비는 테마와 글꼴에 따라 달라지므로, Theme 또는 TextStyle을 바꾸면 모든 칩을 다시 측정하고 컨트롤을 다시 배치합니다.

예제

참고 항목

- 키보드와 접근성
- 헤드리스 출력

현지화

각 컨트롤에는 public FormatSettings 속성이 있습니다. 사용자 로캘의 설정으로 시작하며, 이를 할당하면 컨트롤이 사용하는 이름, 형식, 구분 기호가 바뀝니다. 각 컨트롤은 자체 사본을 유지하므로, 한 폼의 두 컨트롤이 서로 다른 두 로캘을 사용할 수 있습니다.

날짜와 시간

- 달력과 선택기는 FormatSettings에서 짧은 요일 이름과 긴 요일 이름, 짧은 월 이름과 긴 월 이름, 짧은 날짜 및 시간 형식과 긴 날짜 및 시간 형식, AM 및 PM 문자열, 날짜 및 시간 구분 기호를 가져옵니다.
- FirstDayOfWeek의 기본값은 efdLocale이며, 한 주의 첫 요일을 운영 체제에서 가져오고 없으면 월요일입니다. efdMonday, efdSaturday, efdSunday는 이를 강제로 지정합니다.
- WeekNumbering의 기본값은 ewnISO8601이며, 1주차는 그해의 첫 목요일을 포함하는 주입니다. ewnSimple은 1월 1일을 포함하는 주를 1주차로 세며, 한 주의 첫 요일에서 시작합니다.
- DisplayFormat이 비어 있으면 날짜 선택기에서는 로캘의 짧은 날짜, 시간 선택기에서는 짧은 시간, 날짜 및 시간 선택기에서는 둘을 공백으로 구분한 것을 뜻합니다. 구문 분석은 로캘의 월 이름과 요일 이름을 이해합니다. 텍스트 입력과 IME를 참조하십시오.

숫자

스핀 편집기와 범위 슬라이더의 레이블 및 값 툴팁은 FormatSettings의 소수점과 천 단위 구분 기호로 숫자를 표기합니다. 스핀 편집기의 DisplayFormat과 슬라이더의 LabelFormat은 FormatFloat 마스크이며, 슬라이더의 OnFormatLabel은 통화나 단위 같은 임의의 텍스트를 쓸 수 있습니다.

컨트롤의 텍스트

달력 제목, 요일 이름, 시간 목록은 로캘에서 가져오며, 단위, 힌트, 접근성 이름은 직접 지정하는 속성입니다.

예제

ISO 주 번호가 있는 독일어 달력, 독일어 짧은 형식의 날짜 및 시간 선택기, 독일어 구분 기호를 쓰는 스핀 편집기와 슬라이더:

참고 항목

- 텍스트 입력과 IME
- DPI
- TsgcEditorLocale

DPI

sgcEditors는 모든 컨트롤을 논리 픽셀, 즉 96 DPI에서의 픽셀로 배치하고, 화면의 배율로 그립니다. 코드에서 할 일은 없습니다.

VCL

- 컨트롤의 배율은 PPI를 96으로 나눈 값입니다. 콘텐츠와 오버레이는 그 배율로 래스터화되며, 마우스 위치는 모델에 전달되기 전에 그 값으로 나뉩니다.
- 폼이 다른 DPI의 모니터로 이동하면, ChangeScale과 WM_DPICHANGED가 내부 편집 컨트롤과 그 글꼴을 컨트롤과 함께 이동하고 배율을 조정합니다.
- 드롭다운은 열리는 모니터의 PPI로 그려지므로 폼이 두 모니터에 걸쳐 있어도 선명합니다. Delphi 7은 Screen.PixelsPerInch를 사용합니다.

FireMonkey

컨트롤의 래스터는 Width에 장면의 배율을 곱한 값이므로 레티나와 고DPI 안드로이드 화면에서도 선명합니다. 모델은 논리 레이아웃을 유지합니다.

헤드리스 출력

SaveToPNG와 SaveToSVG는 논리 픽셀 단위의 크기와 배율을 받습니다. 배율 2는 너비와 높이가 두 배인 PNG를 쓰며, 192 DPI 화면의 컨트롤과 같은 레이아웃입니다.

측정된 텍스트

토큰 칩의 너비 같은 텍스트 너비는 배율에 따라 달라집니다. 토큰 편집기는 측정기, 배율, 테마를 키로 하는 측정된 칩의 캐시를 유지하므로, DPI가 바뀌면 모든 칩을 다시 측정하고 컨트롤을 다시 배치합니다.

예제

참고 항목

- 헤드리스 출력
- 테마

헤드리스 출력

sgcEditors의 모델은 창 없이 렌더링됩니다. 컨트롤을 보고서, 문서, 웹 페이지에 그리거나, 스냅샷 테스트를 만들거나, Windows나 Linux64의 서버에서 렌더링하는 데 사용하십시오.

모델

TsgcCalendarModel, TsgcDatePickerModel, TsgcSliderModel, TsgcColorPickerModel, TsgcTokenModel, TsgcRatingModel, TsgcSpinModel 같은 모델을 만들고, 속성을 설정한 다음 TsgcEditorModel의 다음 메서드 중 하나를 호출합니다. 이들은 콘텐츠 레이어와 오버레이를 그립니다.

메서드	출력
SaveToPNG(aStream, aWidth, aHeight, aScale)	스트림에 PNG.
SaveToPNG(aFileName, aWidth, aHeight, aScale)	PNG 파일.
SaveToSVG(aStream, aWidth, aHeight, aScale)	스트림에 SVG.
SaveToSVG(aFileName, aWidth, aHeight, aScale)	SVG 파일.

aWidth와 aHeight는 논리 픽셀이며, aScale(기본값 1)이 이를 곱합니다. 헤드리스 출력은 sgcUICore에 내장된 글꼴인 Roboto를 사용하므로, 같은 모델은 모든 컴퓨터에서 같은 픽셀을 만듭니다. Render와 RenderOverlay 메서드는 소유한 임의의 TsgcUICanvas에 그립니다.

컨트롤

모든 VCL 및 FireMonkey 컨트롤에도 스트림 또는 파일 이름과 배율을 받는 SaveToPNG와 SaveToSVG가 있습니다. 컨트롤을 현재 크기로 렌더링하며, 내부 편집 컨트롤의 텍스트는 모델이 그립니다.

Linux64

Delphi 10.2부터 프레임워크 중립 유닛이 Linux64용으로 컴파일되므로, Linux 서비스도 데스크톱 애플리케이션과 같은 달력, 슬라이더, 평점을 렌더링할 수 있습니다.

예제

참고 항목

- DPI
- 아키텍처
- TsgcEditorModel

에디션

sgcEditors는 모든 에디션에서 같은 소스로 빌드됩니다.

에디션	포함 내용
Registered	모든 것: VCL과 FireMonkey의 컨트롤 9종, 각 프레임워크의 데이터 인식 컨트롤 5종, 색상 선택기의 HSV 사각형과 알파 막대, 토큰 편집기의 자동 완성 목록, 범위 슬라이더의 두 핸들.
Trial	Registered와 같은 기능입니다. SGC_TRIAL로 컴파일한 빌드는 빌드에 명시된 릴리스 기간이 지난 뒤 컴포넌트가 생성될 때 체험판이라는 메시지를 표시합니다. 그 밖의 것은 달라지지 않습니다.
Community	VCL과 FireMonkey의 컨트롤 9종 모두를 아래에 나열된 고급 기능 없이 제공하며, 체험판 알림이 없습니다.

고급 기능

고급 기능은 SGC_EDITORS_ADVANCED 정의 아래에서 컴파일되며, sgcVer.inc는 Community 에디션에서 이를 끕니다. 정확히 다음과 같은 것이 바뀝니다.

기능	Community에서
색상 선택기	Parts의 ecpHSV와 ecpAlpha는 무시됩니다. 사각형, 색조 막대, 알파 막대는 그려지지 않으며, 알파는 항상 $FF이고 16진수 텍스트는 #RRGGBB입니다. 견본과 16진수 편집 컨트롤은 남습니다.
토큰 편집기	AutoCompleteItems, AutoCompleteMatch, AutoCompleteMinChars, AutoCompleteMaxItems는 남으므로 Registered 빌드에서 저장한 폼을 불러올 수 있지만, 목록은 열리지 않습니다.
범위 슬라이더	RangeMode는 남지만 핸들은 하나만 그려져 편집됩니다. EndValue는 유지되지만 표시되지 않습니다.
데이터 인식 컨트롤	유닛 sgcEditors_Data, sgcEditors_VCL_DB, sgcEditors_FMX_DB는 빈 유닛으로 컴파일되므로, 데이터 인식 클래스 10개와 그 등록은 존재하지 않습니다. 팔레트에는 컴포넌트가 28개가 아니라 18개 표시됩니다.

Community에서는 전체 달력, 드롭다운이 있는 선택기 3종, 단위가 있는 스핀 편집기, 반 별 평점, 토큰 검증, 테마, 키보드, DPI, 헤드리스 PNG 및 SVG 출력을 사용할 수 있습니다.

모든 에디션을 위한 코드

에디션은 sgcVer.inc에서 컴파일 시점에 선택하며, SGC_COMMUNITY는 SGC_TRIAL도 끕니다. Community에 대해 컴파일되어야 하면서 고급 에디션에만 있는 멤버를 사용하는 유닛은 sgcVer.inc를 포함하고 그 코드를 {$IFDEF SGC_EDITORS_ADVANCED} 안에 넣습니다. 데이터 인식 컨트롤이 있는 폼은 해당 클래스가 없으므로 Community 빌드에서 불러오지 못합니다. sgcEditors_Const에는 버전과 에디션이 상수로 있습니다.

참고 항목

- 설치
- 빠른 시작 데이터 인식

.NET 개요

sgcEditors .NET은 Delphi sgcEditors 라이브러리를 감싼 래퍼가 아니라, 관리형 C#으로 포팅한 것입니다. C# 클래스는 **esegece.sgcEditors** 네임스페이스에 있으며 Delphi 이름을 그대로 유지하므로, **TsgcCalendarModel**, **TsgcSpinModel**, **TsgcEditorLocale**과 열거형은 두 언어에서 똑같이 읽힙니다. 9개 컨트롤은 모든 호스트에서 Delphi 명명 패턴을 따릅니다. **TsgcVCLDatePicker**는 **TsgcWinFormsDatePicker**, **TsgcWpfDatePicker**, **TsgcAvaloniaDatePicker**에 해당합니다. 컨트롤은 **sgcUICore**의 .NET 포트로 그려지며, eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 라이트 테마와 다크 테마를 사용하고, 입력하는 텍스트에는 프레임워크의 내부 텍스트 상자를 사용합니다.

패키지

포트는 5개의 NuGet 패키지로 나뉘어 있으므로, 프로젝트는 사용하는 호스트만 가져옵니다.

PackageId	포함 내용	대상 프레임워크
esegece.sgcEditors	9개 컨트롤의 모델(**TsgcCalendarModel**, **TsgcDatePickerModel**, **TsgcSliderModel**, **TsgcRatingModel**, **TsgcSpinModel**, **TsgcColorPickerModel**, **TsgcTokenModel**), 목록과 드롭다운 모델, 날짜 계산, 로캘, 헤드리스 SVG 출력. 네이티브 종속성은 없습니다. esegece.sgcUICore에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcEditors.Rendering	헤드리스 PNG 출력입니다. **sgcEditors_Raster**의 확장 메서드 **SaveToPNG**와 **RenderToRaster**를 제공하며, Windows, Linux, macOS에서 esegece.sgcUICore.Rendering의 SkiaSharp 캔버스에 그립니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcEditors.WinForms	**TsgcWinFormsCustomEditor** 기반의 9개 **TsgcWinForms** 컨트롤.	net472, net8.0-windows, net9.0-windows
esegece.sgcEditors.Wpf	**TsgcWpfCustomEditor** 기반의 9개 **TsgcWpf** 컨트롤.	net472, net8.0-windows, net9.0-windows
esegece.sgcEditors.Avalonia	**TsgcAvaloniaCustomEditor** 기반의 9개 **TsgcAvalonia** 컨트롤이며, Avalonia 11로 Windows, Linux, macOS에서 실행됩니다.	net8.0, net9.0

세 호스트 패키지는 esegece.sgcEditors.Rendering에 종속되고 이것이 esegece.sgcEditors에 종속되므로, `PackageReference` 한 줄로 호스트에 필요한 모든 것이 설치됩니다. NuGet 설치를 참조하십시오.

에디션

에디션	패키지 ID	포함 항목
Registered	esegece.sgcEditors, .Rendering, .WinForms, .Wpf, .Avalonia	모든 것: 세 호스트의 9개 컨트롤, 테마, 키보드와 접근성 지원, 헤드리스 PNG 및 SVG 출력, 그리고 세 가지 고급 기능인 색 피커의 HSV 사각형과 알파 막대, 토큰 편집기의 자동 완성 목록, 범위 슬라이더의 두 번째 썸.

Community	같은 ID에 .Community 접미사가 붙습니다. 예를 들면 esegece.sgcEditors.WinForms.Community입니다.	세 가지 고급 기능을 제외한 모든 것입니다. 속성은 그대로 남아 있으므로 같은 코드가 두 에디션 모두에서 컴파일됩니다. **Parts**는 **ecpHSV**와 **ecpAlpha**를 무시하고(알파는 항상 $FF), **AutoCompleteItems**는 목록을 열지 않으며, **RangeMode**는 썸 하나만 그리고 편집하지만 **EndValue**는 유지됩니다.

이는 Delphi 에디션과 같은 구분입니다. 에디션을 참조하십시오. Delphi의 데이터 인식 컨트롤에 해당하는 .NET 컨트롤은 어떤 에디션에도 없습니다. 값 속성은 각 프레임워크의 데이터 바인딩으로 바인딩합니다. .NET 데이터 바인딩을 참조하십시오.

호스트

호스트	Delphi 대응	플랫폼
WinForms	VCL(TsgcVCL 컨트롤)	Windows, .NET Framework 4.7.2, .NET 8 및 9.
WPF	VCL과 FMX	Windows, .NET Framework 4.7.2, .NET 8 및 9. 컨트롤은 XAML에서 선언할 수 있습니다.
Avalonia	FMX(TsgcFMX 컨트롤)	Windows, Linux, macOS, .NET 8 및 9.
헤드리스	모델과 해당 SaveToPNG 및 SaveToSVG	대상 프레임워크의 모든 플랫폼에서 사용할 수 있습니다. SVG는 esegece.sgcEditors, PNG는 esegece.sgcEditors.Rendering입니다.

모든 호스트의 API가 같습니다. Delphi 컨트롤과 이름이 같은 속성, 메서드, 이벤트에 더해, 기본 클래스의 **Theme**, **AccentColor**, **TextStyle**, **ReadOnly**, **FormatSettings**, **DropDown**, **CloseUp**, **SaveToPNG**, **SaveToSVG**가 있습니다. 클래스 이름은 .NET 클래스를, Delphi에서 달라지는 점은 Delphi와 .NET의 차이점을 참조하십시오.

참고 항목

- NuGet 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 데이터 바인딩
- .NET 헤드리스 출력
- .NET 데모
- 플랫폼

NuGet 설치

패키지는 nuget.org에 있습니다. 호스트의 패키지를 추가하면 코어 패키지와 렌더링 패키지가 종속성으로 함께 설치됩니다.

Registered 에디션

```
dotnet add package esegece.sgcEditors.WinForms
dotnet add package esegece.sgcEditors.Wpf
dotnet add package esegece.sgcEditors.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcEditors.WinForms.Community
dotnet add package esegece.sgcEditors.Wpf.Community
dotnet add package esegece.sgcEditors.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcEditors.Wpf" Version="2026.10.0" />
</ItemGroup>
```

한 프로젝트에서는 Registered 패키지와 Community 패키지 중 하나만 참조하십시오. 둘 다 참조하면 안 됩니다. 같은 네임스페이스에 같은 형식이 들어 있기 때문입니다.

프로젝트 설정

- WinForms: `<UseWindowsForms>true</UseWindowsForms>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트.
- WPF: `<UseWPF>true</UseWPF>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트. XAML에서는 `xmlns:sgc="clr-namespace:esegece.sgcEditors;assembly=esegece.sgcEditors.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: `net8.0` 또는 `net9.0` 프로젝트이며, 다른 Avalonia 데스크톱 애플리케이션과 마찬가지로 Avalonia 11.3의 **Avalonia.Desktop** 패키지와 **Avalonia.Themes.Fluent** 패키지를 사용합니다. 내부 텍스트 상자는 Fluent 테마에서 모양을 가져옵니다. XAML에서는 `xmlns:sgc="using:esegece.sgcEditors"`로 네임스페이스를 매핑합니다.
- 헤드리스: SVG에는 esegece.sgcEditors만으로 충분합니다. PNG에는 esegece.sgcEditors.Rendering을 추가하십시오. SkiaSharp와 Windows 및 Linux용 네이티브 자산이 함께 설치됩니다.

C#에서는 모든 호스트에 `using esegece.sgcEditors;`와 `using esegece.sgcUICore;`가 필요합니다. 후자는 **TsgcUITheme**, **TsgcUITextStyle** 등 테마의 형식을 위한 것입니다.

참고 항목

- .NET 개요
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 데모

빠른 시작 WinForms

이 빠른 시작에서는 9개 WinForms 컨트롤 중 4개를 사용하여 작은 예약 양식을 코드로 만듭니다. 드롭다운 달력이 있는 날짜 피커, 단위가 있는 스핀 편집기, 범위 슬라이더, 토큰 편집기입니다.

단계

1. `net8.0-windows` Windows Forms 프로젝트를 만들고 **esegece.sgcEditors.WinForms** 패키지(Community 에디션에서는 **esegece.sgcEditors.WinForms.Community**)를 추가합니다.
2. `using esegece.sgcEditors;`와 `using esegece.sgcUICore;`를 추가합니다.
3. 컨트롤을 만들고 속성을 설정한 다음 폼에 추가합니다. 일반 WinForms 컨트롤이므로 Location, Size, Dock, Anchor, TabIndex, Enabled가 평소처럼 동작합니다.
4. 이벤트를 할당합니다. **OnChange**는 값이 변경될 때 모든 컨트롤이 발생시키며, 나머지 이벤트는 Delphi 컨트롤의 것입니다.

예제

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcUICore;
using esegece.sgcEditors;

public class BookingForm : Form
{
    private readonly TsgcWinFormsDatePicker dtpArrival;
    private readonly TsgcWinFormsSpinEdit spnNights;
    private readonly TsgcWinFormsRangeSlider sldBudget;
    private readonly TsgcWinFormsTokenEdit tokExtras;

    public BookingForm()
    {
        Text = "Booking";
        ClientSize = new Size(360, 220);

        // a date picker with a drop-down calendar; dates are OLE Automation dates (double)
        dtpArrival = new TsgcWinFormsDatePicker { Location = new Point(12, 12), Size = new Size(200, 28) };
        dtpArrival.MinDate = DateTime.Today.ToOADate();
        dtpArrival.Date = DateTime.Today.AddDays(7).ToOADate();
        dtpArrival.DisabledWeekDays = TsgcEditorWeekDays.ewdSunday;

        // a spin edit with units and limits
        spnNights = new TsgcWinFormsSpinEdit { Location = new Point(12, 52), Size = new Size(120, 28) };
        spnNights.ValueType = TsgcEditorSpinValueType.esvInteger;
        spnNights.MinValue = 1;
        spnNights.MaxValue = 30;
        spnNights.Value = 2;
        spnNights.Units = "nights";

        // a range slider (RangeMode is a Registered feature, Community draws one thumb)
        sldBudget = new TsgcWinFormsRangeSlider { Location = new Point(12, 92), Size = new Size(320, 48) };
        sldBudget.BeginInit();
        sldBudget.Min = 0;
        sldBudget.Max = 500;
        sldBudget.Step = 10;
        sldBudget.RangeMode = true;
        sldBudget.Value = 80;
        sldBudget.EndValue = 240;
        sldBudget.EndInit();
        sldBudget.TickInterval = 100;
```

```
        sldBudget.ShowLabels = true;

        // a token edit: type a word and press Enter or a comma to add a chip
        tokExtras = new TsgcWinFormsTokenEdit { Location = new Point(12, 152), Size = new Size(320, 36) };
        tokExtras.TextHint = "Extras";
        tokExtras.Tokens.Add("Breakfast");

        // every control raises OnChange; the Sender is the control
        dtpArrival.OnChange = s => Text = "Arrival " + DateTime.FromOADate(dtpArrival.Date).ToShortDateString();
        spnNights.OnChange = s => Text = spnNights.Value + " nights";
        tokExtras.OnTokenAdded = (s, aIndex, aText) => Text = "Added " + aText;

        foreach (TsgcWinFormsCustomEditor vEditor in new TsgcWinFormsCustomEditor[] { dtpArrival, spnNights, sldBudget, tok
        {
            vEditor.Theme = TsgcUITheme.utLight;
            Controls.Add(vEditor);
        }
    }
}
```

참고

- 날짜와 시간 값은 `double`에 담긴 OLE 자동화 날짜이며, Delphi의 TDateTime입니다. `DateTime.ToOADate`와 `DateTime.FromOADate`로 변환합니다.
- 범위 슬라이더는 **Value**와 **EndValue**를 **Min**과 **Max** 범위로 조정하기 전에 **BeginInit**과 **EndInit** 사이에서 대기하므로, 할당 순서는 중요하지 않습니다. Windows Forms 디자이너는 `ISupportInitialize`를 통해 둘 다 호출합니다.
- **Theme**은 컨트롤마다 따로 지정합니다. 모든 컨트롤이 **TsgcWinFormsCustomEditor**에서 파생되므로, 컨트롤을 순회하면 폼 전체의 테마를 설정할 수 있습니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- .NET 데이터 바인딩
- 날짜 피커

빠른 시작 WPF

WPF 컨트롤은 다른 컨트롤과 마찬가지로 XAML에서 선언할 수 있습니다. 이 빠른 시작에서는 날짜 시간 피커, 반 별 평점, 색 피커를 선언하고, 나머지는 코드 비하인드에서 설정합니다.

단계

1. `net8.0-windows` WPF 프로젝트를 만들고 **esegece.sgcEditors.Wpf** 패키지(Community 에디션에서는 **esegece.sgcEditors.Wpf.Community**)를 추가합니다.
2. 창에서 네임스페이스를 매핑합니다. `xmlns:sgc="clr-namespace:esegece.sgcEditors;assembly=esegece.sgcEditors.Wpf"`.
3. 컨트롤을 선언합니다. 열거형에는 값 이름을 지정합니다. 예: `Precision="erpHalf"`. Width와 Height가 없는 컨트롤에는 기본 크기가 적용됩니다(대부분의 피커는 160 x 28, 날짜 시간 피커는 200 x 28, 월 달력은 280 x 240).
4. 코드 비하인드에서 이벤트를 할당합니다.

예제

창:

```
<Window x:Class="QuickStart.QuickStartWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcEditors;assembly=esegece.sgcEditors.Wpf"
        Title="Review" Width="360" Height="300">
  <StackPanel Margin="12">
    <sgc:TsgcWpfDateTimePicker x:Name="dttVisit" Width="220" Height="28" HorizontalAlignment="Left" />
    <sgc:TsgcWpfRating x:Name="rtgScore" Precision="erpHalf" Value="3.5" Margin="0,8,0,0"
                       Width="160" Height="32" HorizontalAlignment="Left" />
    <sgc:TsgcWpfColorPicker x:Name="clrTag" Margin="0,8,0,0" Width="240" Height="180"
                            HorizontalAlignment="Left" />
  </StackPanel>
</Window>
```

코드 비하인드:

```
using System;
using System.Windows;
using esegece.sgcUICore;
using esegece.sgcEditors;

namespace QuickStart
{
    public partial class QuickStartWindow : Window
    {
        public QuickStartWindow()
        {
            InitializeComponent();
            dttVisit.DateTime = DateTime.Now.ToOADate();
            dttVisit.OnChange = s => Title = "Visit " + DateTime.FromOADate(dttVisit.DateTime);
            rtgScore.OnChange = s => Title = "Score " + rtgScore.Value;
            clrTag.Swatches.Clear();
            clrTag.Swatches.Add("#E53935");
            clrTag.Swatches.Add("#43A047");
            clrTag.Swatches.Add("#1E88E5");
            clrTag.Theme = TsgcUITheme.utDark;
        }
    }
}
```

참고

- 값 속성은 기본적으로 양방향으로 바인딩되는 종속성 속성입니다. 피커의 **Date**, **Time**, **DateTime**, 슬라이더의 **Value** 와 **EndValue**, 평점과 스핀 편집기의 **Value**, 색 피커의 **SelectedColor**입니다. .NET 데이터 바인딩을 참조하십시오.
- 색 피커에는 **SelectedWpfColor**가 있으며, 이는 불투명한 `System.Windows.Media.Color`로 나타낸 **SelectedColor** 입니다.
- 드롭다운은 컨트롤 아래에, 화면 아래쪽 근처에서는 위에 놓이는 WPF 팝업이며, 컨트롤이 있는 모니터의 DPI로 그려집니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- .NET 데이터 바인딩
- 평점

빠른 시작 Avalonia

Avalonia 컨트롤은 Windows, Linux, macOS에서 실행됩니다. 이 빠른 시작은 코드로 작성한 완전한 Avalonia 애플리케이션으로, 최대 14일 범위를 선택하는 월 달력과 30분마다 드롭다운 목록이 있는 시간 피커를 사용합니다.

단계

1. `net8.0` 콘솔 프로젝트를 만들고 `<OutputType>WinExe</OutputType>`을 설정한 다음, **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 11.3, 그리고 **esegece.sgcEditors.Avalonia** 패키지(Community 에디션에서는 **esegece.sgcEditors.Avalonia.Community**)를 추가합니다.
2. 애플리케이션에 Fluent 테마를 추가합니다. 피커의 내부 텍스트 상자는 여기에서 모양을 가져옵니다.
3. 컨트롤을 만들어 패널에 추가합니다. XAML에서는 `xmlns:sgc="using:esegece.sgcEditors"`로 네임스페이스를 매핑합니다.

예제

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Layout;
using Avalonia.Themes.Fluent;
using esegece.sgcUICore;
using esegece.sgcEditors;

public class App : Application
{
    public override void Initialize()
    {
        Styles.Add(new FluentTheme());
    }

    public override void OnFrameworkInitializationCompleted()
    {
        if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime vDesktop)
            vDesktop.MainWindow = new PlannerWindow();
        base.OnFrameworkInitializationCompleted();
    }
}

public class PlannerWindow : Window
{
    public PlannerWindow()
    {
        Title = "Planner";
        Width = 320;
        Height = 380;

        // a month calendar with range selection and ISO week numbers
        TsgcAvaloniaMonthCalendar calTrip = new TsgcAvaloniaMonthCalendar { Width = 280, Height = 240 };
        calTrip.SelectionMode = TsgcEditorSelectionMode.esmRange;
        calTrip.ShowWeekNumbers = true;
        calTrip.MaxRangeDays = 14;
        calTrip.OnRangeSelect = s => Title = DateTime.FromOADate(calTrip.RangeStart).ToShortDateString() +
            " to " + DateTime.FromOADate(calTrip.RangeEnd).ToShortDateString();

        // a time picker with a drop-down list every 30 minutes, from 08:00 to 18:00
        TsgcAvaloniaTimePicker tmpStart = new TsgcAvaloniaTimePicker { Width = 140 };
        tmpStart.ListInterval = 30;
        tmpStart.MinTime = new TimeSpan(8, 0, 0).TotalDays;
```

```
        tmpStart.MaxTime = new TimeSpan(18, 0, 0).TotalDays;
        tmpStart.Time = new TimeSpan(9, 30, 0).TotalDays;

        StackPanel pnlMain = new StackPanel { Margin = new Thickness(12), Spacing = 8 };
        pnlMain.Children.Add(calTrip);
        pnlMain.Children.Add(tmpStart);
        Content = pnlMain;
    }
}

public static class Program
{
    [STAThread]
    public static void Main(string[] args)
    {
        AppBuilder.Configure<App>().UsePlatformDetect().StartWithClassicDesktopLifetime(args);
    }
}
```

참고

- 시간은 OLE 자동화 날짜에서 하루에 대한 비율을 나타내는 소수이므로, `TimeSpan.TotalDays`가 시간 값입니다.
- 값 속성은 기본적으로 양방향으로 바인딩되는 스타일 속성입니다. .NET 데이터 바인딩을 참조하십시오.
- 범위 선택은 모든 에디션에 포함됩니다. 범위 슬라이더의 두 번째 썸은 고급 기능입니다. .NET 개요를 참조하십시오.
- **AutomationProperties.Name**은 화면 읽기 프로그램을 위해 컨트롤의 이름을 지정하며, 그 값은 자동화 피어의 값 패턴을 통해 노출됩니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- .NET 데이터 바인딩
- 월 달력

.NET 클래스

모든 Delphi 클래스는 .NET에서도 같은 이름을 유지합니다. VCL과 FireMonkey 컨트롤은 호스트 접두사가 붙은 WinForms, WPF, Avalonia 컨트롤이 되며, 프레임워크 중립 클래스는 모든 호스트에서 동일합니다.

호스트 클래스

Delphi	WinForms	WPF	Avalonia
TsgcVCLMonthCalendar, TsgcFMXMonthCalendar	TsgcWinFormsMonthCalendar	TsgcWpfMonthCalendar	TsgcAvaloniaMonthC
TsgcVCLDatePicker, TsgcFMXDatePicker	TsgcWinFormsDatePicker	TsgcWpfDatePicker	TsgcAvaloniaDatePi
TsgcVCLTimePicker, TsgcFMXTimePicker	TsgcWinFormsTimePicker	TsgcWpfTimePicker	TsgcAvaloniaTimePi
TsgcVCLDateTimePicker, TsgcFMXDateTimePicker	TsgcWinFormsDateTimePicker	TsgcWpfDateTimePicker	TsgcAvaloniaDateTi
TsgcVCLRangeSlider, TsgcFMXRangeSlider	TsgcWinFormsRangeSlider	TsgcWpfRangeSlider	TsgcAvaloniaRangeS
TsgcVCLRating, TsgcFMXRating	TsgcWinFormsRating	TsgcWpfRating	TsgcAvaloniaRating
TsgcVCLSpinEdit, TsgcFMXSpinEdit	TsgcWinFormsSpinEdit	TsgcWpfSpinEdit	TsgcAvaloniaSpinEd
TsgcVCLColorPicker, TsgcFMXColorPicker	TsgcWinFormsColorPicker	TsgcWpfColorPicker	TsgcAvaloniaColorPi
TsgcVCLTokenEdit, TsgcFMXTokenEdit	TsgcWinFormsTokenEdit	TsgcWpfTokenEdit	TsgcAvaloniaTokenE
TsgcVCLCustomEditor, TsgcFMXCustomEditor	TsgcWinFormsCustomEditor	TsgcWpfCustomEditor	TsgcAvaloniaCustom
TsgcVCLCustomDatePicker, TsgcFMXCustomDatePicker	TsgcWinFormsCustomDatePicker	TsgcWpfCustomDatePicker	TsgcAvaloniaCustom
내부 TEdit	TsgcWinFormsEditorInnerEdit (a TextBox)	TsgcWpfEditorInnerEdit (a TextBox)	TsgcAvaloniaEditorIn TextBox)
드롭다운	TsgcWinFormsEditorPopup (a ToolStripDropDown)	TsgcWpfEditorPopup (a Popup)	TsgcAvaloniaEditorP Popup)

기본 클래스는 sgcUICore의 **TsgcUIWinFormsControl**, **TsgcUIWpfControl**, **TsgcUIAvaloniaControl**에서 파생됩니다. 접근성 클래스는 **TsgcWinFormsEditorAccessibleObject**, **TsgcWpfEditorAutomationPeer**, **TsgcAvaloniaEditorAutomationPeer**이며, 컨트롤의 이름과 값을 알려 줍니다.

모든 컨트롤의 멤버

멤버	설명

Model	컨트롤의 모델이며, 위 표에 나온 형식입니다. 형식이 지정된 속성(CalendarModel, SpinModel 등)도 같은 개체를 반환합니다.
InnerEdit	내부 텍스트 상자입니다. 형식이 지정된 텍스트가 없는 컨트롤(월 달력, 범위 슬라이더, 평점)에서는 null입니다.
Popup	피커와 토큰 자동 완성 목록의 드롭다운입니다.
Theme, AccentColor, TextStyle	모양입니다. **TsgcUITheme**(utLight, utDark), `uint`로 표현하는 강조 색($AARRGGBB, 0은 테마 색), 글꼴 크기와 굵기가 있습니다.
ReadOnly	값을 읽고 복사할 수 있지만 변경할 수는 없습니다.
FormatSettings	컨트롤이 서식 지정과 구문 분석에 사용하는 **TsgcEditorFormatSettings**의 복사본이며, 기본값은 현재 문화권입니다. 복사본을 변경한 다음 다시 할당하십시오.
DroppedDown, DropDown(), CloseUp(bool)	드롭다운의 상태입니다. 열기, 그리고 확정하거나 확정하지 않고 닫기가 있습니다.
OnChange	값이 변경될 때 발생하며, Sender는 컨트롤입니다.
SaveToPNG, SaveToSVG	컨트롤의 현재 모습을 지정한 배율로 스트림이나 파일에 그립니다.

프레임워크 중립 형식

형식	설명
TsgcEditorModel	기본 모델입니다. 테마, 포커스, 사용 가능 여부, 호버, 키보드, 마우스, **Render**, **RenderOverlay**, **SaveToSVG**를 다룹니다. TsgcEditorModel을 참조하십시오.
TsgcEditorLocale	서식 설정에서 가져온 요일과 월 이름, 그리고 한 주의 첫 요일입니다. TsgcEditorLocale을 참조하십시오.
TsgcEditorFormatSettings	에디터가 읽는 Delphi TFormatSettings의 부분 집합입니다. `Create()`는 현재 문화권을 사용하고, `Create(CultureInfo)`는 다른 문화권을 사용합니다.
TsgcEditorStrings	Delphi의 TStrings입니다. **DisabledDates**, **Swatches**, **Tokens**, **AutoCompleteItems**에서 사용합니다.
TsgcEditorListModel	시간 드롭다운과 토큰 자동 완성의 목록입니다.
TsgcDateTimeDropDownModel	날짜 시간 피커의 드롭다운입니다. 달력 옆에 시간 목록이 놓입니다.
sgcEditors_Types, sgcEditors_Date	Delphi의 자유 함수와 상수를 담은 정적 클래스입니다. 키 코드(CS_EDITORS_KEY_*), 숫자 서식 지정, 16진수와 HSV 변환, 날짜 계산이 있습니다.
sgcEditors_Raster	esegece.sgcEditors.Rendering에 있습니다. 모든 모델의 확장 메서드 **SaveToPNG**와 **RenderToRaster**입니다.

열거형

열거형은 Delphi의 이름과 값 접두사를 그대로 사용합니다. **TsgcEditorSelectionMode**(esm), **TsgcEditorFirstDayOfWeek**(efd), **TsgcEditorWeekNumbering**(ewn), **TsgcEditorCalendarView**(ecv), **TsgcEditorOrientation**(eor), **TsgcEditorTickPlacement**(etp), **TsgcEditorInvalidTokens**(eti), **TsgcEditorMatchMode**(emm), **TsgcEditorRatingPrecision**(erp), **TsgcEditorRatingGlyph**(erg),

TsgcEditorSpinValueType(esv), **TsgcEditorSpinButtons**(esb)입니다. Delphi의 집합은 `[Flags]` 열거형이며 `|`로 조합합니다. **TsgcEditorWeekDays**, **TsgcEditorColorParts**, **TsgcEditorChangeKinds**, **TsgcEditorShiftState**가 여기에 해당합니다.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- 컨트롤
- 아키텍처

Delphi와 .NET의 차이점

.NET 포트는 Delphi의 이름, 기본값, 키보드 동작, 렌더링 규칙을 유지합니다. 모델의 SVG 출력은 Delphi 출력과 바이트 단위로 동일하며, PNG 출력은 SkiaSharp 래스터라이저의 허용 오차 안에서 일치합니다. 아래의 차이점은 호스트 프레임워크와 .NET 클래스 라이브러리에서 비롯됩니다.

데이터 인식 컨트롤 없음

.NET에는 어떤 에디션에도 DB 컨트롤과 **TsgcEditorFieldLink**가 없습니다. 대신 값 속성을 각 프레임워크의 데이터 바인딩으로 바인딩합니다. WinForms는 `INotifyPropertyChanged`를 통한 `Binding`, WPF는 종속성 속성, Avalonia는 스타일 속성이며, WPF와 Avalonia는 기본적으로 양방향입니다. .NET 데이터 바인딩을 참조하십시오.

컴포넌트 스트리밍 없음

.NET에는 DFM이 없으므로 **DisabledDates**와 **Swatches**에 대한 디자인 타임 문자열 목록 편집기가 없습니다. 코드에서 값을 채우십시오. 범위 슬라이더는 Delphi에서는 Loaded까지 기다리지만, .NET에서는 Windows Forms 디자이너와 XAML 로더가 호출하는 후크인 **BeginInit**과 **EndInit** 사이로 **Value**와 **EndValue**의 범위 조정을 미룹니다.

값, 이벤트, 목록

- TDateTime은 `double`이며 OLE 자동화 날짜입니다. `DateTime.ToOADate`와 `DateTime.FromOADate`로 변환합니다. 시간은 하루에 대한 비율을 나타내는 소수입니다.
- TsgcUIColor는 `uint`이며 $AARRGGBB 형식이고, 0은 테마 색을 뜻합니다. **SelectedVCLColor**에 해당하는 것은 **SelectedWinFormsColor**(`System.Drawing.Color`), **SelectedWpfColor**(`System.Windows.Media.Color`), **SelectedAvaloniaColor**(`Avalonia.Media.Color`)이며, 모두 불투명합니다.
- 이벤트는 Delphi와 같은 이름(**OnChange**, **OnSelect**, **OnTokenAdded** 등)의 델리게이트 속성입니다. 처리기는 `=`로 하나만 할당합니다. Delphi의 `var` 매개 변수는 `ref` 매개 변수가 되며, `ref` 매개 변수가 있는 람다 식에서는 모든 매개 변수 형식을 선언해야 합니다. **TNotifyEvent**는 **TsgcUINotifyEvent**이며, Sender는 컨트롤입니다.
- TStrings 속성은 **TsgcEditorStrings**이며, **Add**, **Insert**, **Delete**, **Clear**, **IndexOf**(대소문자를 구분하지 않음), **Assign**, **BeginUpdate**, **EndUpdate**를 제공합니다. 토큰 목록의 개체는 **GetObject**로 가져옵니다.
- 매개 변수가 없는 Delphi 함수는 `IsRange()`, `CellCount()`, `HexText()`처럼 메서드로 남습니다. Delphi 집합은 `|`로 조합하는 `[Flags]` 열거형입니다.
- TFormatSettings는 **TsgcEditorFormatSettings**입니다. **FormatSettings** 속성은 복사본을 반환하므로, Delphi 레코드처럼 변경한 다음 다시 할당하십시오.

```
using System;
using esegece.sgcEditors;

public static class DifferencesSample
{
    public static void Setup(TsgcWinFormsMonthCalendar aCalendar, TsgcWinFormsSpinEdit aSpin,
        TsgcWinFormsColorPicker aColor, TsgcWinFormsTokenEdit aTokens)
    {
        // events are delegate properties named as in Delphi: assign with =, one handler each
        aCalendar.OnSelect = s => Console.WriteLine("Selected " + DateTime.FromOADate(aCalendar.Date));

        // Delphi var parameters are ref parameters
        aCalendar.OnIsDateDisabled = (object s, double aDate, ref bool aDisabled) =>
        {
            if (DateTime.FromOADate(aDate).Day == 13)
                aDisabled = true;
        };
        aSpin.OnValidate = (object s, ref double aValue, ref bool aAccept) =>
        {
            aAccept = aValue != 13;
        };

        // TDateTime is a double (OLE Automation date)
```

```
        aCalendar.MinDate = new DateTime(2026, 1, 1).ToOADate();

        // the Delphi TStrings are TsgcEditorStrings
        aCalendar.DisabledDates.Clear();
        aCalendar.DisabledDates.Add("2026-12-25");
        aTokens.Tokens.Add("Delphi");

        // TsgcUIColor is a uint, $AARRGGBB; SelectedWinFormsColor is the opaque System.Drawing.Color
        aColor.SelectedColor = 0xFF1E88E5;
        Console.WriteLine(aColor.SelectedWinFormsColor);

#if !SGC_COMMUNITY
        // the HSV square and the alpha bar are drawn in the Registered edition only
        aColor.Parts = TsgcEditorColorParts.ecpSwatches | TsgcEditorColorParts.ecpHSV |
            TsgcEditorColorParts.ecpAlpha | TsgcEditorColorParts.ecpHex;
#endif
    }
}
```

호스트별 텍스트 상자, 키보드, 포커스

- 입력하는 텍스트는 프레임워크의 내부 `TextBox`이므로, 캐럿, 선택, 클립보드, 실행 취소, IME는 프레임워크의 것입니다. **TextHint**는 WinForms에서는 큐 배너, WPF에서는 내부 편집기가 그리는 힌트, Avalonia에서는 `Watermark`입니다.
- Tab, Enter, Escape는 세 호스트 모두에서 Delphi와 같이 동작합니다. Avalonia에서는 컨트롤이 처리한 키가 해당 컨트롤 자체의 KeyDown 처리기에 전달되지 않습니다.
- 모델은 논리 픽셀로 동작합니다. WPF와 Avalonia는 DIP로 레이아웃하므로 내부 편집기와 드롭다운은 어떤 DPI에서도 같은 크기를 유지하며, WinForms는 모니터의 DPI에 따라 크기를 조정합니다.
- 접근성: WinForms는 `AccessibleObject`로 이름과 값을 알리고, WPF는 값 패턴이 있는 `AutomationPeer`, Avalonia는 `AutomationProperties`와 해당 자동화 피어로 알립니다. VCL과 같은 창 텍스트는 없습니다.
- WPF에서 토큰 편집기의 **Tokens**는 종속성 속성이 아니라 CLR 속성이며, 변경 사항은 **OnChange**로 알립니다. Avalonia에서는 직접 속성이며 변경될 때마다 발생합니다.

참고 항목

- .NET 개요
- .NET 클래스
- .NET 데이터 바인딩
- 키보드와 접근성

.NET 데이터 바인딩

Delphi의 데이터 인식 컨트롤은 **TsgcEditorFieldLink**를 통해 컨트롤을 TDataSet 필드에 연결합니다. .NET에서는 일반 컨트롤의 값 속성을 각 프레임워크의 데이터 바인딩으로 바인딩합니다. 이는 두 에디션 모두 같으므로 별도의 DB 컨트롤은 없습니다.

컨트롤	값 속성
MonthCalendar	Date, RangeStart, RangeEnd
DatePicker, TimePicker, DateTimePicker	Date, Time, DateTime(셋 모두 같은 OLE 자동화 날짜를 날짜, 시간 또는 둘 다로 본 것입니다), IsEmpty(읽기 전용)
RangeSlider	Value, EndValue
Rating	Value
SpinEdit	Value, IsEmpty(읽기 전용)
ColorPicker	SelectedColor(`uint`)와 호스트 형식의 읽기 전용 색
TokenEdit	Tokens

WinForms

WinForms 컨트롤은 `INotifyPropertyChanged`를 구현하고 값 속성에 대해 이를 발생시키므로, `DataSourceUpdateMode.OnPropertyChanged`를 지정한 `Binding`은 사용자가 입력하거나 증감하거나 끄는 동안 즉시 값을 다시 씁니다.

```
using System;
using System.ComponentModel;
using System.Windows.Forms;
using esegece.sgcEditors;

public class Booking : INotifyPropertyChanged
{
    private double FArrival = DateTime.Today.ToOADate();
    private double FNights = 2;

    public event PropertyChangedEventHandler PropertyChanged;

    // the editors work with OLE Automation dates, so the bound property is a double
    public double Arrival
    {
        get { return FArrival; }
        set { if (FArrival != value) { FArrival = value; DoChanged(nameof(Arrival)); } }
    }

    public double Nights
    {
        get { return FNights; }
        set { if (FNights != value) { FNights = value; DoChanged(nameof(Nights)); } }
    }

    private void DoChanged(string aName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(aName));
    }
}
```

```
public static class BookingBinding
{
    public static void Bind(TsgcWinFormsDatePicker aArrival, TsgcWinFormsSpinEdit aNights, Booking aBooking)
    {
        // the editors implement INotifyPropertyChanged, so OnPropertyChanged writes back
        // at once, while the user types or steps, not only when the control loses the focus
        aArrival.DataBindings.Add("Date", aBooking, nameof(Booking.Arrival), false,
            DataSourceUpdateMode.OnPropertyChanged);
        aNights.DataBindings.Add("Value", aBooking, nameof(Booking.Nights), false,
            DataSourceUpdateMode.OnPropertyChanged);
    }
}
```

WPF

값 속성은 기본적으로 양방향으로 바인딩되도록 등록된 종속성 속성입니다. **DateProperty**, **TimeProperty**, **DateTimeProperty**, **RangeStartProperty**, **RangeEndProperty**, **ValueProperty**, **EndValueProperty**, **SelectedColorProperty**, 그리고 읽기 전용인 **IsEmptyProperty**입니다. 컨트롤은 바인딩된 값을 한계 안으로 조정할 수 있으며, 그러면 조정된 값을 다시 씁니다. **Tokens**는 CLR 속성이며 바인딩되지 않습니다.

```
using System.ComponentModel;
using System.Windows.Controls;
using System.Windows.Data;
using esegece.sgcEditors;

public class StayViewModel : INotifyPropertyChanged
{
    private double FNights = 2;
    private double FScore = 4;

    public event PropertyChangedEventHandler PropertyChanged;

    public double Nights
    {
        get { return FNights; }
        set { if (FNights != value) { FNights = value; DoChanged(nameof(Nights)); } }
    }

    public double Score
    {
        get { return FScore; }
        set { if (FScore != value) { FScore = value; DoChanged(nameof(Score)); } }
    }

    private void DoChanged(string aName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(aName));
    }
}

public static class StayView
{
    public static StackPanel Create(StayViewModel aViewModel)
    {
        TsgcWpfSpinEdit spnNights = new TsgcWpfSpinEdit { MinValue = 1, MaxValue = 30, Units = "nights" };
        TsgcWpfRating rtgScore = new TsgcWpfRating { Precision = TsgcEditorRatingPrecision.erpHalf };

        // ValueProperty binds two way by default, as in XAML: Value="{Binding Nights}"
        spnNights.SetBinding(TsgcWpfSpinEdit.ValueProperty, new Binding(nameof(StayViewModel.Nights)));
        rtgScore.SetBinding(TsgcWpfRating.ValueProperty, new Binding(nameof(StayViewModel.Score)));

        StackPanel pnlStay = new StackPanel { DataContext = aViewModel };
        pnlStay.Children.Add(spnNights);
        pnlStay.Children.Add(rtgScore);
        return pnlStay;
```

```
    }
}
```

Avalonia

값 속성은 기본적으로 양방향으로 바인딩되는 스타일 속성이며 이름은 WPF와 같습니다. **IsEmptyProperty**, **SelectedAvaloniaColorProperty**, **TokensProperty**는 직접 속성입니다. **Tokens**는 **TsgcEditorStrings**에 양방향으로 바인딩하십시오. 세터가 문자열을 복사합니다.

```
using System.ComponentModel;
using Avalonia.Controls;
using Avalonia.Data;
using esegece.sgcEditors;

public class PriceFilter : INotifyPropertyChanged
{
    private double FLow = 50;
    private double FHigh = 300;

    public event PropertyChangedEventHandler PropertyChanged;

    public double Low
    {
        get { return FLow; }
        set { if (FLow != value) { FLow = value; DoChanged(nameof(Low)); } }
    }

    public double High
    {
        get { return FHigh; }
        set { if (FHigh != value) { FHigh = value; DoChanged(nameof(High)); } }
    }

    private void DoChanged(string aName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(aName));
    }
}

public static class PriceFilterView
{
    public static TsgcAvaloniaRangeSlider Create(PriceFilter aFilter)
    {
        TsgcAvaloniaRangeSlider sldPrice = new TsgcAvaloniaRangeSlider { Min = 0, Max = 500, Step = 10, RangeMode = true };
        sldPrice.DataContext = aFilter;

        // Value and EndValue are styled properties, two way by default,
        // as in XAML: Value="{Binding Low}" EndValue="{Binding High}"
        sldPrice.Bind(TsgcAvaloniaRangeSlider.ValueProperty, new Binding(nameof(PriceFilter.Low)));
        sldPrice.Bind(TsgcAvaloniaRangeSlider.EndValueProperty, new Binding(nameof(PriceFilter.High)));
        return sldPrice;
    }
}
```

BindingWpf 데모는 날짜 피커, 시간 피커, 스핀 편집기, 평점, 범위 슬라이더를 예약 목록에 바인딩합니다. .NET 데모를 참조하십시오.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- .NET 데모
- 빠른 시작 데이터 인식

.NET 헤드리스 출력

모든 컨트롤은 모델을 통해 그려지며, 모델에는 창이 필요하지 않습니다. 서비스, 테스트, 보고서에서 모델을 만들고 속성을 설정한 다음, Linux를 포함한 모든 플랫폼에서 SVG 또는 PNG로 저장할 수 있습니다.

- **SaveToSVG**는 esegece.sgcEditors에 있는 **TsgcEditorModel**의 메서드이며 네이티브 종속성이 없습니다.
- **SaveToPNG**는 esegece.sgcEditors.Rendering에 있는 **sgcEditors_Raster**의 확장 메서드이며 SkiaSharp로 그립니다. **RenderToRaster**는 래스터 캔버스 자체를 반환하며, 호출자가 해제해야 합니다.

둘 다 스트림 또는 파일 이름, 논리 픽셀 단위의 너비와 높이, 선택적인 배율을 받습니다. 배율 2는 280 x 240 달력에 대해 560 x 480 이미지를 기록합니다. 둘 다 콘텐츠 레이어를 그린 다음 오버레이 레이어(호버, 포커스 링, 끄는 중인 썸)를 그리므로, 출력은 컨트롤이 보여 주는 모습과 같습니다.

예제

```
using System;
using esegece.sgcUICore;
using esegece.sgcEditors;

public static class HeadlessSample
{
    public static void Run()
    {
        // the models draw without any window: no WinForms, WPF or Avalonia needed
        TsgcCalendarModel oCalendar = new TsgcCalendarModel();
        oCalendar.Date = new DateTime(2026, 10, 14).ToOADate();
        oCalendar.ShowWeekNumbers = true;
        oCalendar.Theme = TsgcUITheme.utDark;

        // SVG comes with esegece.sgcEditors; width and height are logical pixels
        oCalendar.SaveToSVG("calendar.svg", 280, 240);
        // PNG is an extension method of esegece.sgcEditors.Rendering; the last argument is the scale
        oCalendar.SaveToPNG("calendar@2x.png", 280, 240, 2);

        TsgcRatingModel oRating = new TsgcRatingModel();
        oRating.Precision = TsgcEditorRatingPrecision.erpHalf;
        oRating.Value = 3.5;
        oRating.SaveToPNG("rating.png", 160, 32);

        TsgcSpinModel oSpin = new TsgcSpinModel();
        oSpin.ValueType = TsgcEditorSpinValueType.esvFloat;
        oSpin.Decimals = 2;
        oSpin.Units = "kg";
        oSpin.Value = 12.5;
        oSpin.SaveToSVG("spin.svg", 120, 28);
        Console.WriteLine(oSpin.Text); // 12.50 in an English culture, the text follows FormatSettings
    }
}
```

참고

- 호스트에도 **SaveToPNG**와 **SaveToSVG**가 있으며, 컨트롤의 크기를 사용합니다.
- 모델은 숫자와 날짜를 현재 문화권으로 서식 지정합니다. 모든 컴퓨터에서 같은 출력을 얻으려면 `TsgcEditorFormatSettings.Create(CultureInfo.InvariantCulture)`를 스핀 모델과 슬라이더 모델 또는 호스트 컨트롤의 **FormatSettings**에 할당하십시오.
- 드롭다운도 모델(**TsgcDateTimeDropDownModel**, **TsgcEditorListModel**)이며 같은 메서드로 그려집니다.

참고 항목

- .NET 개요

- .NET 클래스
- 헤드리스 출력

.NET 데모

데모는 `.net\demos\Editors`에 있으며, 데모마다 프로젝트, `Program.cs`, README.md가 들어 있는 폴더가 하나씩 있습니다. 사용자 인터페이스는 XAML 없이 코드로 구성합니다. 모든 데모에는 `--test` 스위치가 있어 화면에 표시하지 않고 자체 검사를 실행하고, PASS와 FAIL 줄을 출력하며, 실패한 검사 수를 종료 코드로 반환합니다. Delphi 데모의 `/test` 스위치와 같습니다.

데모	호스트	보여 주는 내용
Gallery	WinForms	하나의 창에 9개 컨트롤을 모두 담은 데모입니다. 주 번호가 있는 월 달력, 날짜 피커, 15분 간격 목록이 있는 시간 피커, 날짜 시간 피커, 단위가 있는 스핀 편집기, 평점, 눈금이 있는 범위 슬라이더, 자동 완성 목록이 있는 토큰 편집기, 견본, HSV 사각형, 알파, 16진수 편집기가 있는 색 피커입니다. 테마 전환, 읽기 전용 전환, PNG 내보내기, 이벤트 로그도 있습니다.
GalleryWpf	WPF	같은 창을 **TsgcWpf** 컨트롤로 만든 것입니다.
GalleryAvalonia	Avalonia	같은 창을 **TsgcAvalonia** 컨트롤로 만든 것입니다. Windows, Linux, macOS에서 실행됩니다.
BookingForm	WinForms	7개 컨트롤을 사용한 호텔 예약 양식입니다. 숙박 기간(최대 28박)을 위한 범위 모드 달력, 한계와 30분 간격 목록이 있는 체크인 및 체크아웃 시간 피커, 투숙객 수를 위한 스핀 편집기, 자동 완성 목록이 있는 최대 6개 요청용 토큰 편집기, 호텔 등급을 위한 평점, 레이블 색을 위한 견본과 16진수가 있는 색 피커입니다.
BindingWpf	WPF	MVVM입니다. 뷰 모델에 예약 목록이 있고, 선택한 예약에 날짜 피커, 시간 피커, 스핀 편집기, 평점, 범위 슬라이더가 바인딩됩니다. Delphi 데이터 인식 데모에 해당하는 .NET 버전입니다.

Gallery 데모에서는 HSV 사각형, 알파 막대, 범위 슬라이더의 두 번째 썸, 자동 완성 목록이 Registered 에디션에서만 표시되며, BookingForm의 자동 완성 목록도 마찬가지입니다. Community 에디션에서는 같은 코드가 이것들 없이 실행됩니다.

데모 빌드

제품 트리 안에서 데모는 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip은 기본적으로 Community 에디션의 NuGet 패키지를 사용합니다. 모든 데모는 두 에디션 모두에서 빌드하고 실행할 수 있습니다. Registered 패키지로 데모를 빌드하려면 **SgcEditorsEdition**을 설정합니다.

```
dotnet build Gallery\Gallery.csproj -p:SgcEditorsEdition=Registered
dotnet run --project BookingForm\BookingForm.csproj -- --test
```

참고 항목

- .NET 개요
- NuGet 설치
- 빠른 시작 WinForms
- .NET 데이터 바인딩

문제 해결

sgcEditors의 흔한 문제와 그 원인입니다.

문제	원인과 해결 방법
입력한 날짜가 이전 값으로 돌아갑니다.	텍스트를 구문 분석할 수 없었거나, 날짜가 MinDate와 MaxDate 밖에 있거나, DisabledDates, DisabledWeekDays, OnIsDateDisabled로 비활성화되었습니다. OnParseError를 처리하여 직접 정한 표현을 받아들이고, 코드에서 날짜를 설정하기 전에 IsDateEnabled를 호출하십시오.
시간 선택기에 드롭다운 버튼이 없습니다.	ListInterval이 0입니다. 목록 항목 사이의 분 수(기본값 30)로 설정하십시오. 항목은 MinTime과 MaxTime 안에 머뭅니다.
코드에서 설정한 값이 바뀝니다.	값은 제한 안으로 맞춰집니다. Value보다 먼저 MinValue, MaxValue, ValueType, Decimals를 설정하십시오. MinValue와 MaxValue가 같으면 제한 없음입니다. Step이 있는 슬라이더의 Max는 HTML range 입력처럼 단계 격자로 내려갑니다.
FireMonkey 드롭다운에서 한 선택이 사라집니다.	드롭다운 바깥을 클릭하면 닫히며 열려 있는 동안 한 선택이 취소됩니다. Enter 또는 선택으로 닫으십시오.
FireMonkey에서 Tab이 두 번째 핸들로 이동하지 않습니다.	FireMonkey는 Tab으로 포커스를 이동합니다. 위쪽 핸들에는 Ctrl+Right 또는 Ctrl+Up을, 아래쪽 핸들에는 Ctrl+Left 또는 Ctrl+Down을 사용하십시오.
색상 선택기에 사각형이나 알파 막대가 없습니다.	Parts에 ecpHSV 또는 ecpAlpha가 없거나, 빌드가 이를 무시하는 Community입니다.
자동 완성 목록이 열리지 않습니다.	AutoCompleteItems가 비어 있거나, AutoCompleteMinChars보다 적은 문자를 입력했거나, 빌드가 Community입니다.
범위 슬라이더에 핸들이 하나만 표시됩니다.	RangeMode가 False이거나, 빌드가 Community입니다.
Class TsgcVCLDBDatePicker not found when a form loads.	애플리케이션이 데이터 인식 컨트롤이 없는 Community 에디션으로 빌드되었습니다.
데이터 인식 컨트롤을 편집할 수 없습니다.	ReadOnly가 True이거나, DataSource 또는 그 데이터셋이 없거나, 데이터셋 또는 필드가 읽기 전용이거나, 필드가 계산 필드이거나, 범위 모드의 슬라이더에서 끝 필드를 수정할 수 없습니다.
데이터 인식 컨트롤의 Value를 설정하면 데이터셋이 편집 모드가 됩니다.	코드에서 설정한 값은 사용자의 변경으로 간주됩니다. 편집하지 않고 레코드를 바꾸려면 대신 필드에 기록하십시오.
헤드리스 PNG가 화면과 다른 글꼴을 사용합니다.	화면은 플랫폼 글꼴(Windows에서는 Segoe UI)을 사용하고, 헤드리스 출력은 어디서나 같은 결과가 나오도록 내장된 Roboto를 사용합니다.
달력, 슬라이더, 평점에서 IME 입력이 동작하지 않습니다.	이 컨트롤들에는 텍스트가 없습니다. IME는 선택기, 스핀 편집기, 색상 선택기, 토큰 편집기의 내부 편집 컨트롤에서 동작합니다.
편집기가 수백 개인 폼이 느리게 열립니다.	시간의 대부분은 컨트롤과 그 내부 편집 컨트롤의 창을 만드는 데 쓰입니다. 컨트롤 270개에 약 340 ms입니다. 숨겨진 페이지의 컨트롤은 그 페이지가 처음 표시될 때 만드십시오.

예제

참고 항목

- 텍스트 입력과 IME
- 빠른 시작 데이터 인식
- 에디션

소개

sgcGauges는 Delphi와 C++Builder 애플리케이션에 계기 게이지를 추가합니다. VCL용, FireMonkey용, 그리고 사용자 인터페이스 프레임워크 없이 사용하는 경우를 모두 지원합니다. 게이지는 방사형, 선형, 디지털, LED, LED 막대, 진행 링, VU 미터, 시계, 나침반, 애넌시에이터의 10가지 유형입니다. 모든 유형은 TsgcVCL 접두사가 붙은 VCL 컨트롤, TsgcFMX 접두사가 붙은 FireMonkey 컨트롤, 그리고 TsgcRadialGauge처럼 프레임워크에 독립적인 모델 클래스로 제공되며, 모델 클래스는 창 없이 PNG와 SVG로 렌더링합니다.
게이지에는 색상 범위를 가진 하나 이상의 스케일, 하나 이상의 포인터(바늘, 마커, 막대 또는 범위), 그리고 값이 임계값을 넘을 때 이벤트를 발생시키는 임계값이 있습니다. 새 값은 애니메이션과 이징 곡선으로 포인터를 움직입니다. 게이지에는 밝은 테마와 어두운 테마가 있고, 크기와 DPI에 맞게 조정되며, 게이지의 Value는 어느 스레드에서든 쓸 수 있습니다.
모든 게이지는 두 개의 레이어로 그려집니다. 정적 레이어에는 문자판, 스케일, 눈금, 레이블, 범위가 들어 있으며 구성이 바뀔 때만 다시 그려집니다. 동적 레이어에는 포인터와 값 텍스트가 들어 있습니다. 새 값은 동적 레이어만 다시 그리므로 값이 바뀌어도 많은 게이지를 배치한 패널이 빠르게 유지됩니다.
sgcGauges는 eSeGeCe 비주얼 스위트의 일부입니다. 캔버스, 래스터라이저, 글꼴, SVG 라이터, 테마를 갖춘 공유 런타임 패키지인 sgcUICore로 그려지므로 게이지가 스위트의 다른 제품과 어울립니다. sgcGauges에는 sgcUICore가 필요하며 함께 설치됩니다. Registered 에디션의 sgcGaugesWS 브리지는 sgcWebSockets를 통해 MQTT, WebSocket, 서버 전송 이벤트 메시지로 게이지에 값을 공급합니다.

에디션

sgcGauges는 Registered, Trial, Community로 제공됩니다. Community에는 VCL과 FireMonkey용으로 방사형, 선형, 디지털(7 세그먼트), LED, LED 막대, 진행 링 게이지가 포함되며, 범위, 임계값, 애니메이션, 테마, 헤드리스 PNG와 SVG 출력도 사용할 수 있습니다. VU 미터, 시계, 나침반, 애넌시에이터, 첫 번째 이후의 스케일과 포인터, 14 세그먼트와 도트 매트릭스 표시, 브리지는 포함되지 않습니다. 에디션을 참조하십시오.

시작하기

- 아키텍처: 모델, 호스트, 두 개의 렌더 레이어, 스레드.
- 게이지 유형: 10가지 게이지와 그 클래스 및 유닛 한눈에 보기.
- 설치와 플랫폼: 설치 프로그램, 패키지, Delphi 및 C++Builder 버전, 플랫폼.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 헤드리스, 빠른 시작 C++Builder: 몇 분 만에 만드는 첫 게이지.
- 구성 요소: 컨트롤마다 한 페이지씩, 방사형 게이지부터 애넌시에이터까지, 그리고 모델 클래스 TsgcCustomGauge, TsgcGaugeScale, TsgcGaugePointer.
- 스케일, 범위, 포인터, 임계값, 애니메이션, 테마, 헤드리스 출력, 에디션.
- 브리지: MQTT, WebSocket, 서버 전송 이벤트 메시지로 게이지에 값을 공급합니다.
- 문제 해결: 흔한 문제와 그 원인.

아키텍처

sgcGauges는 게이지가 무엇인지와 어디에 표시되는지를 분리합니다. 모델 클래스는 구성과 값을 보유하고 sgcUICore 캔버스에 자신을 그립니다. 호스트 컨트롤은 모델을 VCL 또는 FireMonkey 폼에 배치합니다.

모델

모든 게이지 유형은 **TsgcCustomGauge**(유닛 sgcGauges_Model)를 상속하는 클래스이며, TsgcRadialGauge나 TsgcLinearGauge 등이 있습니다. 모델은 창이 없는 TPersistent이므로 콘솔 프로그램, 서비스, Linux64에서도 실행됩니다. 모델은 다음을 가지고 있습니다.

- **Scales**: TsgcGaugeScale의 컬렉션. 각각 값의 범위, 눈금, 레이블, 색상 **Ranges** 컬렉션을 가집니다.
- **Pointers**: TsgcGaugePointer의 컬렉션. 각각 하나의 스케일에 하나의 값을 표시합니다.
- **Thresholds**: TsgcGaugeThreshold의 컬렉션. 포인터가 넘으면 OnThreshold를 발생시키는 값입니다.
- **Animation**: 포인터 이동의 길이와 이징.
- 공통 외관: Theme, Title, TitleFont, Units, ShowValue, ValueFormat, ValueFont, BackgroundColor, FaceColor.

새 모델에는 항상 0부터 100까지의 스케일 하나와 포인터 하나가 있습니다. **Value**는 Pointers[0].Value의 바로 가기이며, SetValue(aPointer, aValue, aAnimate)는 어느 포인터의 값이든 씁니다.

호스트

VCL 컨트롤은 **TsgcVCLCustomGauge**(유닛 sgcGauges_VCL)를, FireMonkey 컨트롤은 **TsgcFMXCustomGauge**(유닛 sgcGauges_FMX)를 상속합니다. 각 컨트롤은 자기 유형의 모델을 만들며 public **Gauge** 속성이 이를 반환하고, 그 유형이 그리는 모델 속성을 게시합니다. 호스트는 모델이 바뀌면 다시 그리고, 애니메이션 프레임을 실행하고, OnThreshold를 메인 스레드에서 전달하며, SaveToPNG와 SaveToSVG로 클라이언트 크기 그대로 자신을 저장합니다.

두 개의 렌더 레이어

모델은 두 개의 레이어로 그립니다.

- **정적 레이어**(RenderStatic): 배경, 문자판, 베젤, 스케일, 눈금, 레이블, 범위, 임계값 표시, 제목. 속성이 바뀌면 gckStatic으로 OnChange가 발생하고 StaticStamp가 증가합니다.
- **동적 레이어**(RenderDynamic): 포인터와 값 텍스트. 새 값은 gckDynamic의 OnChange만 발생시킵니다.

호스트는 정적 레이어를 캐시된 픽셀로 보관하고, 값이 바뀌거나 포인터가 움직이는 동안에는 동적 레이어만 그립니다. BeginUpdate와 EndUpdate는 여러 변경을 하나의 OnChange로 모으며, 변경 중 하나라도 정적이면 정적 OnChange가 됩니다. Render는 두 레이어를 한 번에 그립니다.

크기, 각도, 색

- 스케일과 포인터의 크기(Radius, Offset, Thickness, Width, Length)는 게이지 크기에 대한 비율이므로 게이지는 어떤 크기에서도 같게 보입니다. 텍스트도 게이지에 맞게 커지고 작아집니다.
- 각도는 도 단위이며 0은 3시 방향이고 양의 각도는 시계 방향입니다. 시계와 나침반은 각각 12시 방향과 북쪽에서부터 셉니다.
- 색은 $AARRGGBB 형식의 TsgcUIColor 값입니다. 색이 0이면 테마의 색을 사용합니다.
- 값은 쓸 때 해당 스케일의 범위로 제한됩니다. NaN은 무시되고 무한대 값은 가장 가까운 경계가 됩니다.

스레드

Value, SetValue, 포인터의 Value와 EndValue는 어느 스레드에서든 쓸 수 있습니다. 모델에서는 OnChange와 OnThreshold가 그 스레드에서 호출됩니다. VCL과 FireMonkey 컨트롤은 값을 쓴 스레드와 상관없이 OnThreshold를 메인 스레드에서 호출합니다. 그 밖의 모든 속성은 메인 스레드에서 사용하도록 되어 있습니다.

유닛

유닛	내용
sgcGauges_Types	열거형과 도우미 함수.
sgcGauges_Model	TsgcCustomGauge, 스케일, 범위, 포인터, 임계값, 애니메이션.
sgcGauges_Render	모든 게이지 유형이 그리는 데 사용하는 페인터.
sgcGauges_Radial , _Linear , _Digital , _LED , _ProgressRing , _VUMeter , _Clock , _Compass , _Annunciator	10가지 게이지 유형의 모델 클래스. LED와 LED 막대는 sgcGauges_LED를 공유합니다.
sgcGauges_VCL	VCL 컨트롤.
sgcGauges_FMX	FireMonkey 컨트롤.
sgcGaugesWS_Source , _Source_MQTT , _Source_WebSocket , _Source_SSE	sgcGaugesWS 브리지. Registered 에디션 전용.

참고 항목

- 게이지 유형
- 헤드리스 출력
- 애니메이션

게이지 유형

sgcGauges에는 10가지 게이지 유형이 있습니다. 각각은 자체 유닛의 모델 클래스이며, sgcGauges_VCL의 VCL 컨트롤이고, sgcGauges_FMX의 FireMonkey 컨트롤입니다. 컨트롤은 **SGC Gauges** 팔레트 페이지에 있습니다.

게이지	모델 클래스(유닛)	VCL 및 FireMonkey 컨트롤	에디션	보여 주는 내용
방사형	TsgcRadialGauge (sgcGauges_Radial)	TsgcVCLRadialGauge, TsgcFMXRadialGauge	Community	전체, 반원, 4분의 1 또는 사용자 지정 호를 가진 다이얼이며 베젤과 값 텍스트가 있습니다. RadialStyle, StartAngle, SweepAngle, BezelStyle, ValuePosition.
선형	TsgcLinearGauge (sgcGauges_Linear)	TsgcVCLLinearGauge, TsgcFMXLinearGauge	Community	가로 또는 세로 막대, 온도계, 목표선이 있는 불릿 그래프. Orientation, LinearStyle, Target, BulbSize.
디지털	TsgcDigitalGauge (sgcGauges_Digital)	TsgcVCLDigitalGauge, TsgcFMXDigitalGauge	Community (7 세그먼트)	값 또는 텍스트의 세그먼트 디스플레이: 7 세그먼트, 14 세그먼트 또는 5 x 7 도트 매트릭스. DisplayType, Digits, Text, Skew, SegmentColor, GhostColor, ShowGhost, Alignment.
LED	TsgcLEDGauge (sgcGauges_LED)	TsgcVCLLEDGauge, TsgcFMXLEDGauge	Community	켜지거나 꺼지는 램프이며 글로우와 선택적인 깜박임이 있습니다. Active, OnColor, OffColor, Shape, Blink, BlinkInterval.
LED 막대	TsgcLEDBarGauge (sgcGauges_LED)	TsgcVCLLEDBarGauge, TsgcFMXLEDBarGauge	Community	피크 홀드가 있는 LED 세그먼트의 행 또는 열. Segments, SegmentGap, Orientation, PeakHold, PeakHoldTime.
진행 링	TsgcProgressRingGauge (sgcGauges_ProgressRing)	TsgcVCLProgressRingGauge, TsgcFMXProgressRingGauge	Community	값에 따라 채워지는 링이며 가운데에 백분율을 표시합니다. Thickness, TrackColor,

				RoundCaps, ShowPercent.
VU 미터	TsgcVUMeterGauge (sgcGauges_VUMeter)	TsgcVCLVUMeterGauge, TsgcFMXVUMeterGauge	Registered	바늘 또는 막대와 피크 홀드가 있는 오디오 레벨 미터.
시계	TsgcClockGauge (sgcGauges_Clock)	TsgcVCLClockGauge, TsgcFMXClockGauge	Registered	라이브 또는 지정한 시각의 아날로그 시계이며 UTC 오프셋을 선택적으로 지정할 수 있습니다.
나침반	TsgcCompassGauge (sgcGauges_Compass)	TsgcVCLCompassGauge, TsgcFMXCompassGauge	Registered	회전하는 카드 또는 회전하는 바늘과 목표 방위가 있는 나침반.
애넌시에이터	TsgcAnnunciatorGauge (sgcGauges_Annunciator)	TsgcVCLAnnunciatorGauge, TsgcFMXAnnunciatorGauge	Registered	네 가지 상태를 가진 알람 타일 그리드이며, 확인될 때까지 깜박입니다.

모든 게이지가 공유하는 것

모든 유형은 TsgcCustomGauge를 상속하므로 Scales, Pointers, Thresholds, Animation, Theme, 값 텍스트 속성을 가지지만, 각 컨트롤은 그 유형이 그리는 것만 게시합니다. 예를 들어 LED에는 스케일이 없고, 시계에는 Value가 없으며, 나침반은 Value 대신 Heading을 게시합니다. Units는 값 텍스트 옆에 그려지며, 값 텍스트는 ValueFormat을 사용하고 비어 있으면 스케일의 소수 자릿수를 사용합니다.

참고 항목

- 아키텍처
- 에디션
- 빠른 시작 VCL

설치

sgcGauges는 sgcUI 배포판의 일부입니다. 하나의 설치 프로그램이 sgcUI 제품을 모두 다루며, 라이선스에 포함된 제품만 설치합니다.

sgcUI 설치 프로그램으로 설치

1. sgcUI 설치 프로그램을 실행하고 eSeGeCe 계정의 사용자 이름과 암호로 로그인합니다.
2. 설치할 Delphi 및 C++Builder 버전을 선택합니다.
3. 구성 요소 페이지에서 **sgcGauges**와 원하는 다른 제품을 선택하십시오. 라이선스 서버가 보유한 제품을 반환하고, 설치 프로그램은 제품마다 하나의 체크박스를 표시합니다. 라이선스에 포함되지 않은 제품은 회색으로 표시됩니다. All-Access 라이선스는 모든 제품을 잠금 해제합니다.
4. 설치 프로그램은 파일을 복사한 다음, 선택한 각 IDE용으로 의존 관계 순서에 따라 패키지를 사용자의 컴퓨터에서 컴파일하고 디자인 타임 패키지를 설치합니다.

sgcGauges와 함께 제공되는 것

sgcUI 비주얼 제품의 공유 기반인 **sgcUICore**는 sgcGauges와 함께 제공됩니다. 자체 체크박스는 없습니다. sgcGauges를 선택하면 설치 프로그램이 자동으로 설치합니다.

설치되는 폴더

모든 제품은 하나의 폴더에 설치되며, 기본값은 `Documents\esegece\sgcUI`입니다(체험판은 `sgcUI_Trial`, Community 에디션은 `sgcUI_Community`).

폴더	내용
`Source`	설치된 모든 제품의 소스 코드. 이것이 sgcUI의 유일한 라이브러리 경로입니다.
`Packages`	설치된 모든 제품의 패키지로, 제품과 IDE 버전마다 한 세트입니다.
`Lib<V>`	IDE 버전과 플랫폼별로 컴파일된 유닛. 예를 들어 `LibD13` 입니다.
`Demos\Gauges`	sgcGauges 데모: Gallery, GalleryFMX, Headless, HMIPanel, DashboardGauges, MQTTLive. Delphi와 C++Builder용.

설치 프로그램은 선택한 각 IDE와 플랫폼의 라이브러리 경로와 찾아보기 경로에 컴파일된 유닛의 폴더와 함께 `sgcUI\Source`를 추가합니다. sgcUICore는 같은 `Source` 폴더에 있으므로 자체 경로가 필요하지 않습니다.

컴포넌트 팔레트

컴포넌트는 **SGC Gauges** 팔레트 페이지에 있습니다. VCL 폼에는 VCL 컨트롤이, FireMonkey 폼에는 FMX 컨트롤이 표시됩니다. Registered와 Trial 에디션은 20개의 컨트롤(프레임워크마다 10가지 게이지 유형)을 등록하며, 브리지는 세 개의 피드 소스(TsgcGaugeFeedSource_WebSocket, TsgcGaugeFeedSource_MQTT, TsgcGaugeFeedSource_SSE)를 추가합니다. Community는 12개의 컨트롤(프레임워크마다 Community의 여섯 가지 게이지 유형)을 등록합니다.

체험판과 Community

- **Trial**: sgcUI 체험판 설치 프로그램은 sgcGauges를 포함한 모든 제품을 체험판 모드로 설치합니다.
- **Community**: 무료 sgcUI Community 설치 프로그램에는 고급 기능이 빠진 sgcGauges가 포함됩니다. 에디션을 참조하십시오.

패키지를 수동으로 빌드하기

패키지를 직접 컴파일하려면 `sgcUI\Packages`에서 사용 중인 IDE 버전용 패키지를 여십시오. `<V>`는 `sgcGaugesD13.dpk`처럼 Delphi 버전 접미사입니다.

패키지	내용
`sgcUICore<V>` , `sgcUICoreVCL<V>` , `sgcUICoreFMX<V>`	공유 sgcUICore 기반과 그 VCL 및 FireMonkey 부분(런타임).
`sgcGauges<V>`	프레임워크에 독립적인 유닛: 형식, 모델, 페인터, 10가지 게이지 모델(런타임).
`sgcGaugesVCL<V>` / `dclsgcGauges<V>`	VCL 컨트롤(런타임 / 디자인 타임).
`sgcGaugesFMX<V>` / `dclsgcGaugesFMX<V>`	Delphi XE6 이상의 FireMonkey 컨트롤(런타임 / 디자인 타임).
`sgcGaugesWS<V>` / `dclsgcGaugesWS<V>`	sgcWebSockets로의 브리지. Registered 에디션 전용. sgcWebSockets 패키지가 필요합니다.

의존성 순서대로 빌드하십시오. `sgcUICore`, `sgcUICoreVCL`, `sgcUICoreFMX`, 그다음 `sgcGauges`, `sgcGaugesVCL`, `sgcGaugesFMX`, 마지막으로 디자인 타임 패키지입니다. 프로젝트 그룹 `sgcGauges<V>.groupproj`에는 sgcGauges 패키지가 그 순서로 들어 있습니다. 그런 다음 디자인 타임 패키지를 설치하고 사용 중인 Delphi 버전의 라이브러리 경로에 `sgcUI\Source`를 추가하십시오 (Tools > Options > Language > Delphi Options > Library > Library path). C++Builder에서는 사용 중인 버전에 맞는 `.cbproj` 프로젝트, 예를 들어 `sgcGaugesC13.cbproj`와 `sgcGaugesVCLC13.cbproj`를 열어 빌드하십시오. 생성된 `.hpp` 헤더는 컴파일된 패키지 옆에 만들어집니다.

참고 항목

- 플랫폼
- 빠른 시작 VCL

플랫폼

대상	지원
Delphi	Delphi 7, 2007, 2009, 2010, XE부터 XE8, 10부터 10.4, 11, 12, 13.
C++Builder	C++Builder 2007, 2010, XE부터 XE8까지, 10부터 10.4까지, 11, 12, 13. 패키지 프로젝트는 sgcGaugesC2007부터 sgcGaugesC13까지입니다. 패키지는 C++Builder 프로젝트가 포함하고 링크하는 .hpp 헤더와 .lib 파일을 생성합니다. 빠른 시작 C++Builder를 참조하십시오.
VCL	Windows, Win32와 Win64의 TsgcVCL 컨트롤. 모든 Delphi 버전.
FireMonkey	Delphi XE6 이상의 TsgcFMX 컨트롤(SGC_GAUGES_FMX). FireMonkey 플랫폼에서 동작합니다. FireMonkey와 sgcUICore만 사용합니다.
Linux64	Delphi 10.2 이상의 프레임워크에 독립적인 유닛(SGC_GAUGES_LINUX): 모델, 페인터, 10가지 게이지 모델과 헤드리스 PNG 및 SVG 출력. Linux 컨트롤은 없습니다.
.NET	동일한 API를 제공하는 C# 버전입니다. 10가지 게이지를 WinForms, WPF, Avalonia 컨트롤로 사용할 수 있고 헤드리스 PNG 및 SVG 출력도 지원하며, .NET Framework 4.7.2와 .NET 8 및 9에서 동작합니다. .NET 개요를 참조하십시오.

패키지

sgcGauges에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcGaugesD13.dpk, C++Builder 13에서는 sgcGaugesC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcGauges	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛: 모델, 페인터, 10가지 게이지 모델과 헤드리스 PNG 및 SVG 출력.	rtl, sgcUICore
sgcGaugesVCL	런타임, VCL	sgcGauges_VCL (TsgcVCLRadialGauge, TsgcVCLLinearGauge, TsgcVCLDigitalGauge, TsgcVCLLEDGauge, TsgcVCLLEDBarGauge, TsgcVCLProgressRingGauge, TsgcVCLVUMeterGauge, TsgcVCLClockGauge, TsgcVCLCompassGauge 및 TsgcVCLAnnunciatorGauge)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcGauges
sgcGaugesFMX	런타임, FireMonkey, XE6부터	sgcGauges_FMX (TsgcFMXRadialGauge, TsgcFMXLinearGauge, TsgcFMXDigitalGauge, TsgcFMXLEDGauge, TsgcFMXLEDBarGauge, TsgcFMXProgressRingGauge, TsgcFMXVUMeterGauge, TsgcFMXClockGauge, TsgcFMXCompassGauge 및 TsgcFMXAnnunciatorGauge)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcGauges

dclsgcGauges	디자인 타임, VCL	VCL 게이지를 **SGC Gauges** 팔레트 페이지에 등록하며(Community 에디션에는 TsgcVCLVUMeterGauge, TsgcVCLClockGauge, TsgcVCLCompassGauge 및 TsgcVCLAnnunciatorGauge 없음), Scales, Pointers, Thresholds, Tiles용 컬렉션 편집기도 제공합니다.	designide, rtl, vcl, sgcGauges, sgcGaugesVCL
dclsgcGaugesFMX	디자인 타임, FireMonkey, XE6부터	FireMonkey 게이지를 **SGC Gauges** 팔레트 페이지에 등록합니다.	designide, rtl, fmx, sgcGauges, sgcGaugesFMX, dclsgcGauges
sgcGaugesWS, dclsgcGaugesWS	런타임 및 디자인 타임, sgcWebSockets 브리지	라이브 피드 소스 (TsgcGaugeFeedSource_WebSocket, TsgcGaugeFeedSource_MQTT 및 TsgcGaugeFeedSource_SSE), 프레임워크에 독립적이며 **SGC Gauges** 팔레트 페이지에 있습니다.	rtl, sgcGauges, sgcWebSockets

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcGaugesFMX에는 dclsgcGauges가 필요하므로 dclsgcGauges를 먼저 설치하십시오. 선택 사항인 sgcGaugesWS 브리지의 유닛을 제외하면 sgcGauges 유닛은 sgcWebSockets 유닛을 사용하지 않습니다.

IDE 버전과 대상 플랫폼

Delphi	sgcGauges	sgcGaugesVCL	sgcGaugesFMX	dclsgcGauges, dclsgcGaugesFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다. sgcGaugesWS는 sgcGauges와 같은 플랫폼용으로, dclsgcGaugesWS는 dclsgcGauges와 같은 플랫폼용으로 빌드됩니다.

C++Builder	sgcGauges, sgcGaugesVCL	sgcGaugesFMX	dclsgcGauges, dclsgcGaugesFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcGauges만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcGauges만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcGaugesC13.cbproj, sgcGaugesVCLC13.cbproj, sgcGaugesFMXC13.cbproj, dclsgcGaugesC13.cbproj, dclsgcGaugesFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. sgcGaugesWS와 dclsgcGaugesWS는 sgcGauges 및 dclsgcGauges와 같은 C++Builder 버전과 플랫폼용으로 빌드되지만, sgcWebSockets 패키지의 Win64x 빌드가 필요하므로 Win64x는 제외됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXRadialGauge, TsgcFMXLinearGauge, TsgcFMXDigitalGauge, TsgcFMXLEDGauge, TsgcFMXLEDBarGauge, TsgcFMXProgressRingGauge, TsgcFMXVUMeterGauge, TsgcFMXClockGauge, TsgcFMXCompassGauge 및 TsgcFMXAnnunciatorGauge)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcGaugesVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcGauges_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcGauges는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)이 사용하는 것은 모델, 페인터, 10가지 게이지 모델과 헤드리스 PNG 및 SVG 출력입니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

버전 제약

- 프레임워크에 독립적인 유닛과 VCL 유닛은 Delphi 7에서 컴파일됩니다. 제네릭, 익명 메서드, 인라인 변수, 레코드 헬퍼는 사용하지 않습니다.
- 출력은 Win32, Win64, Linux64에서 같습니다. 게이지의 PNG와 SVG 파일은 세 플랫폼에서 바이트 단위로 동일합니다.
- 열거형에는 게이지 고유의 접두사(grs, gpk, gdt, gtd, ge 등)가 있으므로 C++Builder 헤더가 다른 sgcUI 제품과 충돌하지 않습니다.

참고 항목

- 에디션
- 아키텍처

빠른 시작 VCL

사용 중인 Delphi 버전용 sgcGauges 패키지를 설치합니다. 컴포넌트는 **SGC Gauges** 팔레트 페이지에 있습니다.

디자인 타임

1. **TsgcVCLRadialGauge**를 폼에 놓습니다. Title에 Speed를, Units에 km/h를 입력합니다.
2. Scales 컬렉션을 열어 첫 번째 스케일을 선택하고 Maximum을 260으로, MajorInterval을 20으로 설정합니다. 그 Ranges 컬렉션을 열어 범위를 추가하고 StartValue를 200, EndValue를 260, Color를 $FFE34948 같은 빨강으로 설정합니다.
3. 그 옆에 **TsgcVCLLEDGauge**를 놓고 OnColor를 같은 빨강으로, Blink를 True로 설정합니다.
4. 방사형 게이지의 Thresholds 컬렉션에 Value 200, Hysteresis 10, Direction gtdBoth인 임계값을 추가하고, LED의 Active 속성에 aRising을 설정하는 OnThreshold 핸들러를 작성합니다.
5. 타이머나 작업자 스레드에서 코드로 Value를 설정하고 애플리케이션을 실행합니다. 바늘은 새 값마다 300 ms 동안 움직입니다.

코드에서

게이지는 Parent가 생기는 즉시 동작합니다. 값은 쓸 때 스케일로 제한되므로, 값보다 먼저 스케일의 범위를 설정하십시오.

참고 항목

- 게이지 유형
- 임계값
- 애니메이션

빠른 시작 FMX

FireMonkey 컨트롤은 Delphi XE6 이상에서 TsgcFMX 접두사를 가진 VCL 컨트롤과 같은 속성을 가집니다. sgcGauges 패키지를 설치하고 FireMonkey 폼을 열어 **SGC Gauges** 팔레트 페이지의 컴포넌트를 사용하십시오.

디자인 타임

1. **TsgcFMXRadialGauge**를 폼에 놓고 RadialStyle을 grsHalf로, Theme을 utDark로 설정합니다.
2. Title, Units, ValueFormat과 Scales 컬렉션의 첫 번째 스케일의 Maximum을 설정합니다.
3. **TsgcFMXLinearGauge**를 놓고 Orientation을 gloVertical로, LinearStyle을 glsThermometer로 설정합니다.
4. 아무 FireMonkey 플랫폼에서나 애플리케이션을 실행합니다.

FireMonkey에서 알아 둘 점

- 컨트롤은 장면의 스케일로 그려지므로 고 DPI 화면과 모바일 기기에서도 선명합니다.
- OnThreshold는 값을 쓴 스레드와 상관없이 TThread.Queue를 통해 메인 스레드에서 호출됩니다.

코드에서

참고 항목

- 빠른 시작 VCL
- 테마
- 플랫폼

빠른 시작 헤드리스

모델 클래스는 창 없이 렌더링합니다. 콘솔 프로그램, 웹 서버, 보고서는 게이지를 만들고 값을 설정한 다음 PNG 또는 SVG 파일로 저장할 수 있습니다. 모델 유닛은 VCL도 FireMonkey도 사용하지 않으므로 같은 코드가 Windows와 Linux64에서 동작합니다.

단계

1. uses 절에 sgcGauges_Model과 sgcGauges_Radial처럼 게이지 유형의 유닛을 추가합니다.
2. `TsgcRadialGauge.Create`처럼 모델을 만들고 컨트롤에서와 같이 스케일, 범위, 값을 설정합니다.
3. 논리 픽셀 크기와 선택적인 스케일을 지정하여 SaveToPNG 또는 SaveToSVG를 호출한 다음 모델을 해제합니다.

코드에서

헤드리스 렌더링은 목표 값을 그립니다. 애니메이션과 프레임 타이밍은 컨트롤의 몫입니다. 헤드리스 출력을 참조하십시오.

참고 항목

- 헤드리스 출력
- 아키텍처

빠른 시작 C++Builder

사용 중인 C++Builder 버전용 sgcGauges 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 include 경로와 라이브러리 경로에 추가한 다음, **SGC Gauges** 팔레트 페이지의 컴포넌트를 Delphi와 똑같이 사용하십시오. 설치 프로그램이 두 경로를 IDE에 추가해 줍니다. sgcGauges와 sgcUICore의 생성된 .hpp 헤더와 .lib 파일은 그 하나의 폴더에 있습니다.

C++에서 알아 둘 점

- 컨트롤을 선언하는 유닛의 헤더를 include하십시오. VCL 컨트롤은 `sgcGauges_VCL.hpp`, FireMonkey 컨트롤은 `sgcGauges_FMX.hpp`입니다. 컬렉션과 이벤트 형식은 `sgcGauges_Model.hpp`에, 열거형은 `sgcGauges_Types.hpp`에 있습니다.
- 컬렉션에는 C++에서 기본 인덱스가 없습니다. Delphi에서 `Gauge.Scales[0]`이라고 쓰는 곳에 `Gauge->Scales->Items[0]`이라고 쓰십시오.
- 이벤트는 클로저이므로 그 핸들러는 클래스의 __fastcall 멤버 함수입니다.
- 색은 0xAARRGGBB 형식의 부호 없는 32비트 값입니다.
- 모델 클래스에는 `new TsgcRadialGauge()`처럼 인수가 없는 생성자가 있으며, 직접 delete해야 합니다. 헤드리스 출력을 참조하십시오.

코드에서

참고 항목

- 빠른 시작 VCL
- 설치

방사형 게이지

클래스: TsgcVCLRadialGauge (유닛 sgcGauges_VCL), TsgcFMXRadialGauge (유닛 sgcGauges_FMX) 및 모델 TsgcRadialGauge (유닛 sgcGauges_Radial)
다이얼: 호 위의 스케일, 문자판 둘레의 베젤, 포인터, 값 텍스트. 호와 그 크기는 컨트롤에 맞춰지므로 반원 게이지는 아래쪽이 평평해지고 넓은 컨트롤을 채웁니다.
이 게이지는 모든 에디션에 포함됩니다. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
RadialStyle	TsgcGaugeRadialStyle	호: grsFull(기본값, 135도에서 270도), grsHalf(180도에서 180도), grsQuarter(180도에서 90도) 또는 grsCustom.
StartAngle	Double	grsCustom에서 호의 시작 위치(도). 0은 3시 방향이고 양수는 시계 방향입니다. 기본값 135.
SweepAngle	Double	grsCustom에서 호의 길이(도). 기본값 270.
BezelStyle	TsgcGaugeBezelStyle	gbzNone, gbzThin(기본값) 또는 gbzThick.
ValuePosition	TsgcGaugeValuePosition	값 텍스트가 놓이는 위치: gvpBottom(기본값), gvpCenter 또는 gvpTop.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Thresholds	TsgcGaugeThresholds	포인터가 넘을 때 OnThreshold를 발생시키는 값.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
Units	string	km/h처럼 값 텍스트 옆에 그려지는 텍스트.
ShowValue	Boolean	값 텍스트를 그립니다. 기본값 True.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 소수 자릿수를 사용합니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

이벤트

OnFormatValue(Sender, aPointer, aValue, var aText)는 값 텍스트를 대체합니다.

예제

빨간 범위가 있는 반원 회전계:

참고 항목

- 게이지 유형
- TsgcCustomGauge

선형 게이지

클래스: TsgcVCLLinearGauge (유닛 sgcGauges_VCL), TsgcFMXLinearGauge (유닛 sgcGauges_FMX) 및 모델 TsgcLinearGauge (유닛 sgcGauges_Linear)
막대, 온도계 또는 불릿 그래프로 그려지는 가로 또는 세로 게이지입니다. 기본 포인터는 막대입니다.
이 게이지는 모든 에디션에 포함됩니다. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Orientation	TsgcGaugeOrientation	gloHorizontal(기본값)은 왼쪽에서 오른쪽으로, gloVertical은 아래에서 위로 이어집니다.
LinearStyle	TsgcGaugeLinearStyle	glsBar(기본값): 스케일 트랙과 포인터. glsThermometer: 스케일 0은 최솟값 쪽에 구부가 있는 관이 되고 포인터 0은 액체로 그려집니다. glsBullet: 두꺼운 띠, 더 가는 막대 뒤의 범위, 그리고 Target 선.
Target	Double	불릿 스타일의 목표선. 스케일 안에 있을 때 스케일 0을 가로질러 그려집니다. 기본값 0.
BulbSize	Double	온도계 구부의 반지름. 게이지 크기에 대한 비율. 기본값 0.06.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Thresholds	TsgcGaugeThresholds	포인터가 넘을 때 OnThreshold를 발생시키는 값.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
Units	string	km/h처럼 값 텍스트 옆에 그려지는 텍스트.
ShowValue	Boolean	값 텍스트를 그립니다. 기본값 True.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 소수 자릿수를 사용합니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

이벤트

OnFormatValue(Sender, aPointer, aValue, var aText)는 값 텍스트를 대체합니다.

예제

세로 온도계와 불릿 그래프:

참고 항목

- 게이지 유형
- TsgcCustomGauge

디지털 게이지

클래스: TsgcVCLDigitalGauge (유닛 sgcGauges_VCL), TsgcFMXDigitalGauge (유닛 sgcGauges_FMX) 및 모델 TsgcDigitalGauge (유닛 sgcGauges_Digital)
포인터 0의 값 또는 텍스트를 표시하는 세그먼트 디스플레이입니다. 7 세그먼트 유형은 숫자, 빼기 부호, 소수점, 콜론을 표시하고, 14 세그먼트 유형은 문자가 추가되며, 5 x 7 도트 매트릭스는 ASCII 32부터 126까지 표시합니다. 꺼진 세그먼트는 켜진 세그먼트 뒤에 흐리게 그릴 수 있습니다.
글리프 위치의 수는 Text에서, 또는 Digits로 설정하지 않으면 스케일의 Minimum과 Maximum 텍스트 중 더 긴 쪽에서 정해집니다. 들어가지 않는 값은 대시로 표시되고, 들어가지 않는 Text는 Digits 글리프 뒤에서 잘립니다. 세그먼트 유형에서 '.' 또는 ','는 앞 글리프의 소수점을 켜며 자체 위치를 차지하지 않습니다.
7 세그먼트 디스플레이는 모든 에디션에 포함됩니다. Community에서는 14 세그먼트와 도트 매트릭스 유형이 7 세그먼트로 그려집니다. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
DisplayType	TsgcGaugeDisplayType	gdt7Segment(기본값), gdt14Segment 또는 gdtDotMatrix.
Digits	Integer	글리프 위치의 수. 0(기본값)은 텍스트에서 개수를 가져옵니다.
Text	string	값 대신 표시할 텍스트. 비어 있으면(기본값) 값을 표시합니다.
Skew	Double	글리프를 기울입니다. 단위는 도이며 -30부터 30까지. 양수는 위쪽을 오른쪽으로 움직입니다. 기본값 0.
SegmentColor	TsgcUIColor	켜진 세그먼트의 색. 0은 ValueFont 색을 사용하며, 없으면 테마 텍스트 색을 사용합니다. 넘어선 임계값의 ValueColor가 우선합니다.
GhostColor	TsgcUIColor	꺼진 세그먼트의 색. 0은 켜진 색을 흐리게 한 색을 사용합니다.
ShowGhost	Boolean	꺼진 세그먼트를 그립니다. 기본값 True.
Alignment	TsgcUITextHAlign	더 짧은 텍스트의 위치. 기본값 thaRight.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Thresholds	TsgcGaugeThresholds	포인터가 넘을 때 OnThreshold를 발생시키는 값.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
Units	string	km/h처럼 값 텍스트 옆에 그려지는 텍스트.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 소수 자릿수를 사용합니다.

ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

이벤트

OnFormatValue(Sender, aPointer, aValue, var aText)는 값 텍스트를 대체합니다.

예제

녹색 여섯 자리 카운터와 고정 텍스트:

참고 항목

- 게이지 유형
- TsgcCustomGauge

LED

클래스: TsgcVCLLEDGauge (유닛 sgcGauges_VCL), TsgcFMXLEDGauge (유닛 sgcGauges_FMX) 및 모델 TsgcLEDGauge (유닛 sgcGauges_LED)
글로우가 있고 깜박일 수 있는 표시 램프 하나입니다. Active가 True인 동안 램프가 켜집니다. Active는 Value <> 0을 보여 주는 뷰이므로, 값에 0 또는 1을 쓰는 피드 바인딩으로 램프를 켜고 끌 수 있습니다. LED는 NaN을 제외한 모든 값을 스케일로 제한하지 않고 그대로 유지하며, 애니메이션하지 않습니다.
이 게이지는 모든 에디션에 포함됩니다. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Active	Boolean	램프를 켭니다. 설정하면 값에 1 또는 0이 쓰입니다. 폼에 저장되지 않습니다.
OnColor	TsgcUIColor	켜진 램프의 색. 0은 테마 팔레트의 녹색입니다.
OffColor	TsgcUIColor	꺼진 램프의 색. 0은 켜진 색을 반투명하게 한 색을 사용합니다.
Shape	TsgcGaugeLEDShape	gshCircle(기본값), gshSquare 또는 gshRect.
Blink	Boolean	켜진 램프가 BlinkInterval마다 켜졌다 꺼졌다 합니다. 기본값 False. 디자인 시점에는 깜박이지 않습니다.
BlinkInterval	Integer	깜박임 주기(밀리초). 기본값 500, 최소 50.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

예제

켜져 있는 동안 깜박이는 빨간 알람 램프:

참고 항목

- 게이지 유형
- TsgcCustomGauge
- LED 막대

LED 막대

클래스: TsgcVCLLEDBarGauge (유닛 sgcGauges_VCL), TsgcFMXLEDBarGauge (유닛 sgcGauges_FMX) 및 모델 TsgcLEDBarGauge (유닛 sgcGauges_LED)
포인터 0의 스케일을 나누는 LED 세그먼트의 행 또는 열입니다. 세그먼트는 그 중심이 포인터의 원점과 값 사이에 있을 때 켜집니다. 세그먼트는 그것이 속한 스케일 범위의 색을 가지며, 범위가 없으면 스케일을 3등분한 녹색, 호박색, 빨간색을 가집니다. 피크 홀드는 레벨이 떨어진 뒤에도 한동안 가장 높은 세그먼트를 켜 둡니다.
이 게이지는 모든 에디션에 포함됩니다. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Segments	Integer	세그먼트의 수, 1부터 100까지. 기본값 10.
SegmentGap	Double	두 세그먼트 사이의 간격. 세그먼트에 대한 비율이며 최대 0.9. 기본값 0.2.
Orientation	TsgcGaugeOrientation	gloHorizontal(기본값)은 왼쪽에서 오른쪽, gloVertical은 아래에서 위.
PeakHold	Boolean	피크 세그먼트를 켜 둡니다. 기본값 False.
PeakHoldTime	Integer	피크를 유지하는 시간(밀리초). 기본값 1000.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

예제

파란 범위와 피크 홀드가 있는 20 세그먼트의 세로 레벨 막대:

참고 항목

- 게이지 유형
- TsgcCustomGauge
- 범위

진행 링

클래스: TsgcVCLProgressRingGauge (유닛 sgcGauges_VCL), TsgcFMXProgressRingGauge (유닛 sgcGauges_FMX) 및 모델 TsgcProgressRingGauge (유닛 sgcGauges_ProgressRing)
포인터 0의 값에 따라 12시 방향에서 시계 방향으로 채워지는 링이며 가운데에 백분율을 표시합니다. 막대는 포인터의 Color를, 없으면 값을 포함하는 범위의 색을, 그것도 없으면 테마의 강조색을 사용합니다. 기본 포인터는 막대입니다.
이 게이지는 모든 에디션에 포함됩니다. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Thickness	Double	링의 너비. 게이지 크기에 대한 비율. 기본값 0.15.
TrackColor	TsgcUIColor	빈 링의 색. 0은 테마를 사용합니다.
RoundCaps	Boolean	전체 링보다 짧은 막대의 양 끝을 둥글게 합니다. 기본값 True.
ShowPercent	Boolean	True(기본값)는 스케일 안에서 값의 백분율을 ValueFormat 또는 '0'과 % 기호로 표시합니다. False는 값 텍스트와 Units를 표시합니다.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Thresholds	TsgcGaugeThresholds	포인터가 넘을 때 OnThreshold를 발생시키는 값.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
Units	string	km/h처럼 값 텍스트 옆에 그려지는 텍스트.
ShowValue	Boolean	값 텍스트를 그립니다. 기본값 True.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 소수 자릿수를 사용합니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

이벤트

ShowPercent가 False일 때 OnFormatValue(Sender, aPointer, aValue, var aText)는 값 텍스트를 대체합니다.

예제

업로드 진행률. 값은 바이트 수:

참고 항목

- 게이지 유형
- TsgcCustomGauge

VU 미터

클래스: TsgcVCLVUMeterGauge (유닛 sgcGauges_VCL), TsgcFMXVUMeterGauge (유닛 sgcGauges_FMX) 및 모델 TsgcVUMeterGauge (유닛 sgcGauges_VUMeter)
오디오 레벨 미터입니다. 기본 스케일은 데시벨 매핑으로 -20에서 +3 dB까지 이어지므로 위치가 진폭을 따르고, 눈금은 바깥쪽에 있으며 0 dB부터 빨간 범위입니다. 포인터는 실제 미터처럼 진폭으로 움직입니다. MajorInterval이 0인 동안에는 스케일에 전통적인 VU 표시가 나타납니다.
이 게이지에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 모델이 아무것도 그리지 않으며 컨트롤도 설치되지 않습니다. 에디션을 참조하십시오. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
MeterStyle	TsgcGaugeVUMeterStyle	gvmNeedle(기본값): 90도 호 위의 바늘. gvmBar: 트랙과 범위가 있는 띠 위의 가로 막대.
PeakHold	Boolean	레벨의 피크를 표시합니다. 기본값 False.
PeakHoldTime	Integer	피크가 떨어지기 전에 유지되는 시간(밀리초). 기본값 1500.
PeakDecay	Double	홀드 후 피크가 떨어지는 속도(dB/초). 기본값 20. 0이면 홀드 후 피크를 즉시 떨어뜨립니다.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Thresholds	TsgcGaugeThresholds	포인터가 넘을 때 OnThreshold를 발생시키는 값.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
Units	string	km/h처럼 값 텍스트 옆에 그려지는 텍스트.
ShowValue	Boolean	값 텍스트를 그립니다. 기본값 True.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 소수 자릿수를 사용합니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.
Value	Double	포인터 0의 값. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.

메서드

ResetPeak는 피크를 지웁니다.

이벤트

OnFormatValue(Sender, aPointer, aValue, var aText)는 값 텍스트를 대체합니다.

예제

참고 항목

- 게이지 유형
- TsgcCustomGauge

시계

클래스: TsgcVCLClockGauge (유닛 sgcGauges_VCL), TsgcFMXClockGauge (유닛 sgcGauges_FMX) 및 모델 TsgcClockGauge (유닛 sgcGauges_Clock)
60개의 분 눈금, 12개의 숫자, 시침, 분침, 초침이 있는 아날로그 시계입니다. Time을 표시하며, Live이면 현재 시각을 표시합니다. 현재 시각은 로컬 시간이거나, UseUTCOffset을 지정한 경우 UTC 시간에 UTCOffset분을 더한 시간입니다. 라이브 시계는 1초마다, ShowSeconds가 없으면 1분마다 다시 그립니다. 숫자는 스케일 0의 LabelFont를 사용합니다. 시계에는 Value도 포인터도 없습니다.
이 게이지에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 모델이 아무것도 그리지 않으며 컨트롤도 설치되지 않습니다. 에디션을 참조하십시오. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Time	TDateTime	Live가 False일 때 표시되는 시각.
Live	Boolean	현재 시각을 표시합니다. 기본값 False.
UseUTCOffset	Boolean	라이브 시계가 로컬 시간 대신 UTC 시간에 UTCOffset을 더한 시간을 표시합니다. 기본값 False.
UTCOffset	Integer	UTC로부터의 오프셋(분). 60이나 -300 등. 기본값 0.
Numerals	TsgcGaugeClockNumerals	gcnArabic(기본값), gcnRoman 또는 gcnNone.
ShowSeconds	Boolean	초침을 그립니다. 기본값 True.
SmoothSeconds	Boolean	초침이 똑딱거리지 않고 부드럽게 움직입니다. 기본값 False.
HourHandColor	TsgcUIColor	시침의 색. 0은 테마를 사용합니다.
MinuteHandColor	TsgcUIColor	분침의 색. 0은 테마를 사용합니다.
SecondHandColor	TsgcUIColor	초침의 색. 0은 테마를 사용합니다.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.

모델에는 현재 시각을 반환하는 함수인 public TimeSource 속성도 있습니다. 이를 지정하면 라이브 시계가 시스템 시계 대신 이것을 읽으므로, 테스트에서 고정된 프레임을 얻을 수 있습니다.

예제

UTC+5:30의 로마 숫자 라이브 시계와 고정 시각 시계:

참고 항목

- 게이지 유형
- TsgcCustomGauge

나침반

클래스: TsgcVCLCompassGauge (유닛 sgcGauges_VCL), TsgcFMXCompassGauge (유닛 sgcGauges_FMX) 및 모델 TsgcCompassGauge (유닛 sgcGauges_Compass)
방위 표시기입니다. Heading은 북쪽에서 시계 방향으로 0부터 360도까지의 스케일에서 포인터 0의 값입니다. 제한되지 않고 순환하며(450은 90, -30은 330), 이동은 가장 짧은 호를 따라 애니메이션되므로 359에서 1로는 2도 회전합니다. 카드에는 10도마다 눈금, 방위 기호, 30도마다 각도 레이블이 있으며, 스케일 0의 LabelFont로 그려집니다. ValueFormat을 설정하지 않으면 방위 텍스트는 정수 각도와 도 기호로 표시됩니다.
이 게이지에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 모델이 아무것도 그리지 않으며 컨트롤도 설치되지 않습니다. 에디션을 참조하십시오. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Heading	Double	도 단위의 방위. 포인터 0의 값입니다. 어느 스레드에서든 쓸 수 있습니다. 폼에 저장되지 않습니다.
CompassStyle	TsgcGaugeCompassStyle	gcsRotatingCard(기본값): 카드가 회전하고 방위는 위쪽의 고정된 표시 아래에 놓입니다. gcsRotatingNeedle: 카드는 고정되고 바늘이 방위를 가리킵니다.
ShowCardinals	Boolean	N, E, S, W와 중간 방위를 그립니다. 기본값 True.
ShowDegrees	Boolean	카드의 각도 레이블을 그립니다. 기본값 True. 방위 텍스트는 ShowValue를 따릅니다.
TargetHeading	Double	목표 표시의 방위. 기본값 0.
ShowTarget	Boolean	테두리에 테마의 강조색으로 목표 표시를 그립니다. 기본값 False.
Scales	TsgcGaugeScales	눈금, 레이블, 색상 범위를 가진 스케일. TsgcGaugeScale을 참조하십시오.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오.
Animation	TsgcGaugeAnimation	포인터 이동의 Enabled, Duration, Easing.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
ShowValue	Boolean	값 텍스트를 그립니다. 기본값 True.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 소수 자릿수를 사용합니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.

이벤트

OnFormatValue(Sender, aPointer, aValue, var aText)는 방위 텍스트를 대체합니다.

예제

참고 항목

- 게이지 유형
- TsgcCustomGauge

애넌시에이터

클래스: TsgcVCLAnnunciatorGauge (유닛 sgcGauges_VCL), TsgcFMXAnnunciatorGauge (유닛 sgcGauges_FMX) 및 모델 TsgcAnnunciatorGauge (유닛 sgcGauges_Annunciator)
경보 패널: 제목 아래에 행 단위로 채워지는 타일 그리드입니다. 각 타일에는 Caption, State(gasNormal, gasAdvisory, gasWarning, gasAlarm), Acknowledged 플래그가 있으며 상태에 해당하는 색을 가집니다. 확인되지 않은 경고나 알람은 깜박이고, 권고는 계속 켜져 있습니다. State가 바뀌면 Acknowledged가 지워지므로 새 상태마다 별도의 확인이 필요합니다. 캡션은 ValueFont의 크기와 굵기를 사용하며 타일 너비에 맞게 잘립니다. 애넌시에이터에는 스케일, 포인터, 임계값이 없습니다.
이 게이지에는 Registered 또는 Trial 에디션이 필요합니다. Community에서는 모델이 아무것도 그리지 않으며 컨트롤도 설치되지 않습니다. 에디션을 참조하십시오. VCL 클래스는 TsgcVCLCustomGauge를, FireMonkey 클래스는 TsgcFMXCustomGauge를 상속합니다. 둘 다 아래 표의 속성, Theme, OnThreshold, MaxFPS(애니메이션의 프레임 속도 제한, 기본값 60)에 더해 Align, Anchors, Enabled, Visible, PopupMenu, OnClick, 마우스 이벤트 등 각 프레임워크의 일반 항목을 게시합니다. public Gauge 속성은 모델을 반환하며, SaveToPNG와 SaveToSVG는 컨트롤을 클라이언트 크기의 aScale 배로 저장합니다.

속성

속성	형식	설명
Tiles	TsgcGaugeAnnunciatorTiles	타일. Caption, State, Acknowledged를 가진 TsgcGaugeAnnunciatorTile의 항목.
Columns	Integer	한 행의 타일 수. 기본값 4.
FlashInterval	Integer	깜박임 주기(밀리초). 기본값 500, 최소 50.
NormalColor	TsgcUIColor	정상 타일의 색. 0은 테마의 문자판 색을 사용합니다.
AdvisoryColor	TsgcUIColor	권고 타일의 색. 0은 파랑.
WarningColor	TsgcUIColor	경고 타일의 색. 0은 호박색.
AlarmColor	TsgcUIColor	알람 타일의 색. 0은 빨강.
Title	string	게이지의 캡션.
TitleFont	TsgcUITextStyle	제목의 글꼴. Color가 0이면 테마를 따릅니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴.
BackgroundColor	TsgcUIColor	게이지 뒤의 색. 0은 테마의 배경을 사용합니다.
FaceColor	TsgcUIColor	문자판의 색. 0은 테마를 사용합니다.

메서드

메서드	설명
Acknowledge(aIndex)	정상이 아닌 타일 하나를 확인합니다. 깜박임이 멈춥니다.
AcknowledgeAll	정상이 아닌 모든 타일을 확인합니다.
TileAt(X, Y)	컨트롤의 한 지점에 있는 타일의 인덱스. 없으면 -1.

HoverTile	마우스 아래에 있는 타일의 인덱스. 없으면 -1.

이벤트

TsgcGaugeTileClickEvent 형식(유닛 sgcGauges_Annunciator)의 OnTileClick(Sender, aIndex)는 타일 위에서 왼쪽 버튼을 놓을 때 발생합니다.

예제

타일 세 개. 클릭하면 타일을 확인합니다.

참고 항목

- 게이지 유형
- TsgcCustomGauge

TsgcCustomGauge

유닛: sgcGauges_Model
10가지 게이지 모델의 기반 클래스로, 창이 없는 TPersistent입니다. 호스트 컨트롤은 자기 유형의 모델을 만들어 Gauge 속성으로 반환하고, 사용자 인터페이스가 없는 프로그램은 TsgcRadialGauge.Create처럼 모델을 직접 만들고 해제합니다. 새 모델에는 0부터 100까지의 스케일 하나와 포인터 하나가 있습니다.

속성

속성	형식	설명
Scales	TsgcGaugeScales	스케일. TsgcGaugeScale을 참조하십시오. Community는 첫 번째 것만 사용합니다.
Pointers	TsgcGaugePointers	포인터. TsgcGaugePointer를 참조하십시오. Community는 첫 번째 것만 사용합니다.
Thresholds	TsgcGaugeThresholds	임계값. TsgcGaugeThreshold를 참조하십시오.
Animation	TsgcGaugeAnimation	Enabled(기본값 True), Duration(밀리초, 기본값 300), Easing(기본값 geCubicOut).
Theme	TsgcUITheme	utLight(기본값) 또는 utDark.
Title, TitleFont	string, TsgcUITextStyle	게이지의 캡션과 그 글꼴.
Units	string	값 텍스트 옆에 그려지는 텍스트. ValueToText는 이를 덧붙이지 않습니다.
ShowValue	Boolean	값 텍스트를 그립니다. 기본값 True.
ValueFormat	string	값 텍스트의 FormatFloat 마스크. 비어 있으면 스케일의 주 간격의 소수 자릿수를 사용합니다.
ValueFont	TsgcUITextStyle	값 텍스트의 글꼴. Color가 0이면 테마를 따릅니다.
BackgroundColor, FaceColor	TsgcUIColor	게이지 뒤의 색과 문자판의 색. 0은 테마를 사용합니다.
Value	Double	포인터 0의 값. SetValue(0, Value)와 같습니다.
FormatSettings	TFormatSettings	Public. 값 텍스트와 레이블의 설정이며, 기본적으로 소수점 구분자는 '.'입니다.
StaticStamp	Integer	Public, 읽기 전용. 정적 변경이 있을 때마다 증가합니다. 호스트는 이를 비교하여 캐시한 정적 레이어가 최신인지 확인합니다.
Clock	TsgcGaugeClockFunc	Public. 애니메이션용 시간(밀리초)을 반환하는 함수. nil은 시스템 시계를 사용합니다.
Owner	TPersistent	Public. 호스트 컨트롤이며, 헤드리스 모델에서는 nil.
Designing	Boolean	Public. 아무것도 애니메이션되지 않는 디자인 시점에 호스트가 설정합니다.

메서드

메서드	설명
SetValue(aPointer, aValue, aAnimate = True)	포인터의 값을 씁니다. aAnimate와 Animation.Enabled가 True이면 애니메이션됩니다. 스레드 안전합니다.
BeginUpdate, EndUpdate	그 사이의 변경을 하나의 OnChange로 모읍니다.
DisplayValue(aPointer, aNowMs)	애니메이션 중 Clock의 어느 시각에 포인터가 그려지는 값.
ValueToText(aPointer, aValue)	ValueFormat과 OnFormatValue를 적용한 값의 텍스트.
IsAnimating(aNowMs), NextFrameMs(aNowMs)	포인터가 움직이는지, 그리고 다음 시간 지정 프레임(깜박임, 시계 틱)이 언제 필요한지.
ActiveScaleCount, ActivePointerCount, PointerScale(aPointer)	에디션이 그리는 스케일과 포인터, 그리고 포인터의 스케일.
Render, RenderStatic, RenderDynamic	aScale로 aWidth x aHeight 논리 픽셀의 TsgcUICanvas에 두 레이어 또는 한 레이어를 그립니다.
SaveToPNG, SaveToSVG	aScale로 aWidth x aHeight 논리 픽셀의 게이지를 스트림 또는 파일에 저장합니다.

이벤트

이벤트	설명
OnChange(Sender, aKind)	변경: 정적 레이어는 gckStatic, 값 또는 포인터 이동은 gckDynamic. 변경이 일어난 스레드에서 호출됩니다.
OnThreshold(Sender, aThreshold, aValue, aRising)	포인터가 임계값을 넘었습니다. 모델에서는 값을 쓴 스레드에서, 컨트롤에서는 메인 스레드에서 호출됩니다.
OnFormatValue(Sender, aPointer, aValue, var aText)	값 텍스트를 대체합니다.

예제

포인터 두 개를 가지고 변경 횟수를 세는 헤드리스 게이지:

참고 항목

- 아키텍처
- 헤드리스 출력
- 애니메이션

TsgcGaugeScale와 TsgcGaugeRange

유닛: sgcGauges_Model
TsgcGaugeScale은 게이지의 Scales 컬렉션(TsgcGaugeScales)의 항목입니다. 스케일은 값을 호 또는 축 위의 위치에 대응시키고 트랙, 눈금, 레이블, 색상 범위를 그립니다. TsgcGaugeRange는 스케일의 Ranges 컬렉션(TsgcGaugeRanges)의 항목입니다. 크기는 게이지 크기에 대한 비율입니다.

TsgcGaugeScale

속성	형식	설명
Minimum, Maximum	Double	값의 범위. 기본값 0과 100. 쓸 때 제한되는 값보다 먼저 설정하십시오.
Mapping	TsgcGaugeScaleMapping	gsmLinear(기본값) 또는 gsmDecibel. 후자에서는 위치가 진폭을 따릅니다.
Reversed	Boolean	스케일을 Maximum에서 Minimum 방향으로 진행시킵니다. 기본값 False.
Radius	Double	다이얼에서 스케일의 바깥쪽 반지름. 기본값 1.
Offset	Double	선형 게이지의 축을 가로지르는 띠의 위치. 기본값 0.
Thickness	Double	트랙의 너비. 기본값 0.05.
MajorInterval	Double	레이블이 있는 눈금 사이의 간격. 0(기본값)은 간격을 자동으로 고릅니다.
MinorCount	Integer	두 주 눈금 사이의 보조 눈금 수. 기본값 4.
TickPlacement	TsgcGaugeTickPlacement	gtpInside(기본값), gtpOutside 또는 gtpCross.
TickColor	TsgcUIColor	눈금의 색. 0은 테마를 사용합니다.
ShowLabels	Boolean	주 눈금의 레이블을 그립니다. 기본값 True.
LabelFormat	string	레이블의 FormatFloat 마스크. 비어 있으면 간격의 소수 자릿수를 사용합니다.
LabelFont	TsgcUITextStyle	레이블의 글꼴.
Ranges	TsgcGaugeRanges	색상 범위.
Visible	Boolean	스케일을 그립니다. 기본값 True.

MajorStep은 실제로 사용되는 간격을 반환하고 ValueToRatio(aValue)는 값의 위치를 0부터 1까지로 반환합니다.

TsgcGaugeRange

속성	형식	설명
StartValue, EndValue	Double	범위가 덮는 값.

Color	TsgcUIColor	띠의 색.
Thickness	Double	띠의 너비. 0은 스케일의 두께를 사용합니다.
Caption	string	범위의 이름. 그려지지는 않습니다.

예제

참고 항목

- 스케일
- 범위
- TsgcCustomGauge

TsgcGaugePointer, TsgcGaugeThreshold 및 TsgcGaugeAnimation

유닛: sgcGauges_Model
TsgcGaugePointer 는 Pointers 컬 렉 션(TsgcGaugePointers)의 항 목, TsgcGaugeThreshold 는 Thresholds 컬 렉 션(TsgcGaugeThresholds)의 항목, TsgcGaugeAnimation은 게이지의 Animation 속성입니다.

TsgcGaugePointer

속성	형식	설명
Kind	TsgcGaugePointerKind	gpkNeedle(기본값), gpkMarker, gpkBar 또는 gpkRange(Value에서 EndValue까지의 띠).
ScaleIndex	Integer	포인터가 사용하는 스케일. 기본값 0.
Value	Double	값. 스레드 안전하며, 쓰면 애니메이션됩니다.
EndValue	Double	gpkRange 포인터의 끝. 스레드 안전합니다.
Origin	Double	막대가 시작하는 위치. 설정하지 않으면(NaN) 스케일의 Minimum입니다. OriginValue는 실제로 사용되는 값을 반환합니다.
Color	TsgcUIColor	색. 0은 테마를 사용합니다.
Width, Length, Offset	Double	크기와 위치. 기본값 0.04, 0.9, 0.
NeedleStyle	TsgcGaugeNeedleStyle	gndTapered(기본값), gndLine 또는 gndArrow.
MarkerShape	TsgcGaugeMarkerShape	gmsTriangle(기본값), gmsCircle, gmsDiamond 또는 gmsLine.
CapVisible	Boolean	바늘의 축 위에 캡을 그립니다. 기본값 True.
Visible	Boolean	포인터를 그립니다. 기본값 True.

TsgcGaugeThreshold

속성	형식	설명
Value	Double	감시하는 값.
PointerIndex	Integer	감시하는 포인터. 기본값 0.
Direction	TsgcGaugeThresholdDirection	gtdRising(기본값), gtdFalling 또는 gtdBoth.
Hysteresis	Double	임계값이 다시 발생하기 전에 값이 얼마나 되돌아가야 하는지. 기본값 0.
Color	TsgcUIColor	스케일에 표시되는 임계값 표시의 색. 0(기본값)은 표시를 그리지 않습니다.

ValueColor	TsgcUIColor	임계값을 넘은 동안 값 텍스트의 색. 포인터 0의 임계값에 적용됩니다.
Enabled	Boolean	기본값은 True입니다.
Active	Boolean	Public, 읽기 전용. 값이 임계값을 넘어 있는 동안 True.

TsgcGaugeAnimation

속성	형식	설명
Enabled	Boolean	포인터 이동을 애니메이션합니다. 기본값 True.
Duration	Integer	이동의 길이(밀리초). 기본값 300.
Easing	TsgcGaugeEasing	geLinear, geQuadOut, geCubicOut(기본값), geCubicInOut 또는 geBackOut.

예제

참고 항목

- 포인터
- 임계값
- 애니메이션

스케일

스케일은 값을 게이지 위의 위치에 대응시킵니다. 방사형 게이지에서는 호, 선형 게이지에서는 선입니다. 게이지의 Scales 컬렉션에는 0부터 100까지의 스케일이 항상 하나 이상 있습니다. 포인터는 ScaleIndex로 지정된 스케일에 값을 표시합니다.

TsgcGaugeScale의 속성

속성	설명
Minimum, Maximum	스케일의 범위. 기본값 0과 100. 범위를 바꾸면 그 스케일에 있는 포인터의 값이 다시 제한됩니다.
Mapping	gsmLinear는 값을 고르게 배치합니다. gsmDecibel은 VU 미터처럼 데시벨 곡선 위에 배치합니다.
Reversed	스케일이 Maximum에서 Minimum 방향으로 진행됩니다.
Radius, Offset, Thickness	스케일 띠의 위치와 너비. 게이지 크기에 대한 비율입니다. Radius의 기본값은 1(호의 바깥쪽 가장자리)이고 Thickness는 0.05입니다. Radius가 0.7인 두 번째 스케일은 첫 번째 스케일 안쪽에 놓입니다.
MajorInterval, MinorCount	레이블이 있는 두 눈금 사이의 간격과 그 사이의 보조 눈금 수(기본값 4). MajorInterval이 0이면 약 5개 구간이 되는 깔끔한 간격을 고릅니다. MajorStep은 실제로 사용되는 간격을 반환합니다.
TickPlacement, TickColor	띠 안쪽(gtpInside), 바깥쪽(gtpOutside) 또는 띠를 가로지르는(gtpCross) 눈금과 그 색(0 = 테마).
ShowLabels, LabelFormat, LabelFont	주 눈금의 레이블. LabelFormat은 FormatFloat 마스크이며, 비어 있으면 레이블은 주 간격의 소수 자릿수를 사용합니다. 겹치는 레이블은 솎아 냅니다.
Ranges	스케일의 색상 띠. 범위를 참조하십시오.
Visible	스케일을 그립니다. 기본값 True.

ValueToRatio는 매핑과 Reversed를 적용한 스케일 위 값의 위치를 시작점의 0부터 끝점의 1까지로 반환합니다.

여러 스케일

Registered 에디션에서 게이지는 컬렉션의 모든 스케일을 그립니다. 예를 들어 바깥쪽에 km/h, 안쪽에 mph입니다. Community에서는 첫 번째 스케일만 그려집니다. 나머지 스케일은 컬렉션에 남으므로 Registered 에디션으로 만든 폼도 불러올 수 있습니다.

예제

참고 항목

- 범위
- 포인터
- 에디션

범위

범위는 녹색 정상 띠와 빨간 위험 띠처럼 스케일의 일부에 색을 입힙니다. 모든 스케일에는 TsgcGaugeRange 항목으로 이루어진 자체 Ranges 컬렉션이 있습니다.

TsgcGaugeRange의 속성

속성	설명
StartValue, EndValue	띠가 시작하고 끝나는 값(스케일 단위). 어떤 순서로 지정해도 됩니다.
Color	띠의 색. 0은 범위의 인덱스에 따라 테마 팔레트의 색을 사용합니다.
Thickness	띠의 너비. 게이지 크기에 대한 비율이며 스케일의 바깥쪽 가장자리에서부터 그려집니다. 0은 스케일 띠 전체를 채웁니다.
Caption	범위에 붙이는 이름이며 범위와 함께 보관됩니다. 게이지는 이를 그리지 않습니다.

범위와 포인터

자체 색이 없는 막대 포인터(Color 0)는 값이 속한 범위의 색을 사용하므로, 값이 빨간 띠에 들어가면 선형 게이지의 막대가 빨갛게 됩니다. 임계값으로 값 텍스트에 색을 입힐 수도 있습니다. 임계값을 참조하십시오.

예제

참고 항목

- 스케일
- 임계값
- 테마

포인터

포인터는 하나의 스케일에 하나의 값을 표시합니다. 게이지의 Pointers 컬렉션에는 항상 하나 이상의 포인터가 있으며, Value는 첫 번째 포인터의 값에 대한 바로 가기입니다.

포인터의 종류

종류	설명
gpkNeedle	방사형 게이지의 중심에서 뻗는 끝이 가늘어지는 바늘, 선 또는 화살표(NeedleStyle)이며, 축 위에 캡(CapVisible)이 있습니다. 기본 종류입니다.
gpkMarker	스케일 위의 작은 도형: 삼각형, 원, 마름모 또는 선(MarkerShape). 설정값이나 목표값에 유용합니다.
gpkBar	Origin에서 값까지 스케일을 채우는 띠.
gpkRange	Value에서 EndValue까지의 띠. 예를 들어 허용 오차 구간.

TsgcGaugePointer의 속성

속성	설명
Kind, ScaleIndex	포인터의 종류와 그 값을 표시하는 스케일.
Value, EndValue	값과 범위 포인터의 띠 끝. 둘 다 스케일로 제한되며 어느 스레드에서든 쓸 수 있습니다.
Origin	막대가 시작하는 위치이며 스케일로 제한됩니다. 설정하지 않으면 막대는 스케일의 Minimum에서 시작합니다. -50부터 50까지의 스케일에서 Origin이 0이면 가운데에서 왼쪽 또는 오른쪽으로 자라는 막대가 그려집니다.
Color	포인터의 색. 0이면 바늘과 마커는 테마 팔레트의 색을, 막대는 값이 속한 범위의 색(또는 강조색)을, 범위 포인터는 반투명 강조색을 사용합니다.
Width, Length, Offset	포인터의 크기와 위치. 게이지 크기에 대한 비율입니다(기본값 Width 0.04, Length 0.9).
NeedleStyle, MarkerShape, CapVisible, Visible	바늘의 모양(gndTapered, gndLine, gndArrow)과 마커의 모양, 바늘의 캡, 그리고 포인터를 그릴지 여부.

여러 포인터

Registered 에디션에서는 컬렉션의 모든 포인터가 그려지고, 애니메이션되고, 임계값이 평가됩니다. 예를 들어 현재 값의 바늘과 설정값의 마커입니다. Community에서는 첫 번째 포인터만 활성이며 나머지는 컬렉션에 남습니다.

예제

참고 항목

- 스케일

- 애니메이션
- 임계값

임계값

임계값은 한 포인터의 값을 감시하고 값이 그것을 넘으면 OnThreshold를 발생시킵니다. 알람, 경고 또는 값 텍스트의 색 변경에 사용하십시오.

임계값이 동작하는 방식

- 임계값은 값을 쓸 때 새 목표 값에 대해 평가되며, 애니메이션 중에 그려지는 값에 대해서는 평가되지 않습니다.
- gtdRising과 gtdBoth는 Value 이상에서 활성이 되고 Value에서 Hysteresis를 뺀 값 아래로 내려가면 활성이 해제됩니다. gtdFalling은 Value 이하에서 활성이 되고 Value에 Hysteresis를 더한 값 위로 올라가면 활성이 해제됩니다. 히스테리시스는 한계 주변에서 흔들리는 값이 계속 다시 발생하는 것을 막아 줍니다.
- gtdRising과 gtdFalling에서는 임계값이 활성이 될 때 OnThreshold가 발생합니다. gtdBoth에서는 두 방향의 교차에서 모두 발생하며 aRising이 방향을 알려 줍니다. 해제 이벤트도 필요한 알람에는 히스테리시스와 함께 gtdBoth를 사용하십시오.
- 임계값의 새 Value처럼 구성이 바뀌면 이벤트 없이 Active가 갱신됩니다.

TsgcGaugeThreshold의 속성

속성	설명
Value, Hysteresis, Direction	한계, 불감대, 방향(기본값 gtdRising).
PointerIndex	임계값이 감시하는 포인터. 기본값 0.
Color	0이 아니면 이 색의 표시가 스케일의 Value 위치에 그려집니다.
ValueColor	0이 아니면 임계값이 활성인 동안 값 텍스트가 이 색을 사용합니다(첫 번째 포인터의 임계값).
Enabled, Active	임계값을 끄고, 현재 활성인지 알려 줍니다(읽기 전용).

OnThreshold

`procedure(Sender: TObject; aThreshold: TsgcGaugeThreshold; aValue: Double; aRising: Boolean)`. VCL 과 FireMonkey 컨트롤은 값을 쓴 스레드와 상관없이 교차가 일어난 순서대로 메인 스레드에서 호출합니다. 호스트가 없는 모델은 값을 쓴 스레드에서 호출합니다.

예제

참고 항목

- 범위
- 포인터
- 빠른 시작 VCL

애니메이션

값이 바뀌면 포인터는 표시 중인 값에서 새 값으로 이징 곡선을 따라 짧은 시간 동안 움직입니다. 값 자체, 임계값, 이벤트는 즉시 바뀌고 움직이는 것은 그림뿐입니다.

TsgcGaugeAnimation의 속성

속성	설명
Enabled	값 변경을 애니메이션합니다. 기본값 True.
Duration	이동 시간(밀리초). 기본값 300. 0이면 애니메이션이 꺼집니다.
Easing	이동 곡선: geLinear, geQuadOut, geCubicOut(기본값), geCubicInOut 또는 끝에서 약간 넘어가는 geBackOut.

값이 애니메이션되지 않는 경우

- 디자인 시점, Enabled가 False인 경우, 또는 Duration이 0인 경우.
- 값을 `Gauge.SetValue(aPointer, aValue, False)`로 쓴 경우.
- Pointers 컬렉션이 업데이트되는 동안 폼 파일에서 값을 불러오는 경우.
- 헤드리스 렌더링에서는 항상 목표 값을 그립니다.

이동 중에 도착한 새 값은 그 순간 그려지고 있는 값에서 시작하므로 포인터가 튀지 않습니다.

프레임

VCL과 FireMonkey 컨트롤은 이동의 프레임을 동적 레이어에만, 초당 최대 MaxFPS번(기본값 60) 그립니다. 아무것도 움직이지 않을 때 게이지는 프레임을 전혀 그리지 않습니다. 게이지의 시간은 모델의 Clock 함수에서 가져옵니다. DisplayValue는 지정한 시각에 그려지는 값을 반환하고 IsAnimating은 포인터가 아직 움직이는지 알려 줍니다.

예제

참고 항목

- 포인터
- 아키텍처
- 헤드리스 출력

테마

게이지는 **Theme**에 설정된 sgcUICore 테마에서 색을 가져옵니다. utLight(기본값) 또는 utDark입니다. 게이지의 모든 색 속성은 기본적으로 0이며 이는 테마의 색을 뜻하므로, 자체 색이 없는 게이지는 테마를 완전히 따릅니다.

색

속성	0일 때
BackgroundColor	테마의 배경.
FaceColor	다이얼의 스케일 뒤에 있는 테마의 문자판 색.
Scales[i].TickColor	테마의 흐린 텍스트 색.
Ranges[i].Color, Pointers[i].Color	테마 팔레트의 색. 범위와 포인터를 참조하십시오.
TitleFont.Color, ValueFont.Color, LabelFont.Color	테마의 텍스트 색.

색은 $AARRGGBB로 씁니다. 예를 들어 불투명한 빨강은 $FFE34948입니다.

텍스트

- TitleFont, ValueFont와 각 스케일의 LabelFont는 Size, Bold, Color를 가진 TsgcUITextStyle 값입니다. 크기는 게이지에 대한 상대값이며, 텍스트는 컨트롤과 함께 커지고 작아집니다. 값 텍스트는 ValueFont의 두 배 크기로 그려집니다.
- ValueFormat은 값 텍스트의 FormatFloat 마스크입니다. 비어 있으면 값은 스케일의 소수 자릿수를 사용합니다. Units는 값 옆에 그려집니다.
- 숫자는 모델의 FormatSettings를 사용하며 기본적으로 소수점 구분자는 '.'이므로, 바꾸지 않는 한 게이지는 어느 로캘에서나 같게 보입니다.
- OnFormatValue를 사용하면 어떤 포인터의 값 텍스트든 대체할 수 있습니다.

예제

참고 항목

- 범위
- 빠른 시작 FMX
- 헤드리스 출력

헤드리스 출력

모든 게이지는 창 없이 렌더링됩니다. 모델 클래스는 sgcUICore 캔버스에 그리고 PNG와 SVG 파일을 저장하므로, Windows나 Linux64의 서버, 보고서, 콘솔 프로그램도 컨트롤과 같은 이미지를 만들 수 있습니다.

모델의 메서드

메서드	설명
SaveToPNG(aStream or aFileName, aWidth, aHeight, aScale = 1)	aWidth x aHeight 논리 픽셀에 aScale을 곱한 PNG 이미지. 스케일 2는 고 DPI 화면이나 인쇄를 위해 두 배 크기의 이미지를 만듭니다.
SaveToSVG(aStream or aFileName, aWidth, aHeight, aScale = 1)	같은 그림의 SVG 문서. SVG 안의 숫자는 로캘에 의존하지 않습니다.
Render(aCanvas, aWidth, aHeight, aScale = 1, aNowMs = -1)	두 레이어를 TsgcUICanvas에 그립니다. aNowMs가 0보다 작으면 포인터는 목표 값으로 그려집니다.
RenderStatic, RenderDynamic	정적 레이어만 또는 동적 레이어만 그립니다.

컨트롤

VCL과 FireMonkey 컨트롤에는 SaveToPNG(aStream 또는 aFileName, aScale = 1)와 같은 인수의 SaveToSVG가 있습니다. 컨트롤의 클라이언트 크기로 게이지를 저장하므로, 패널의 스크린샷에 창 캡처가 필요 없습니다.

어디서나 같은 출력

헤드리스 출력은 결정적입니다. 같은 게이지는 Win32, Win64, Linux64에서 바이트 단위로 동일한 PNG와 SVG 파일을 만듭니다. 모델 유닛은 VCL도 FireMonkey도 사용하지 않으며, Linux64에서는 Delphi 10.2부터 컴파일됩니다.

예제

참고 항목

- 빠른 시작 헤드리스
- 플랫폼
- 테마

에디션

sgcGauges는 모든 에디션에서 같은 소스로 빌드됩니다.

에디션	포함 항목
Registered	모든 것: VCL과 FireMonkey용 10가지 게이지 유형, 게이지당 여러 스케일과 포인터, 14 세그먼트와 도트 매트릭스 디스플레이, sgcWebSockets로 연결되는 sgcGaugesWS 브리지.
Trial	Registered와 같은 기능입니다. SGC_TRIAL로 컴파일한 빌드는 빌드에 명시된 릴리스 기간이 지난 뒤 컴포넌트가 생성될 때 체험판이라는 메시지를 표시합니다. 그 밖의 것은 달라지지 않습니다.
Community	VCL과 FireMonkey용 여섯 가지 게이지 유형. 아래에 나열된 고급 기능은 없으며 체험판 안내도 없습니다.

Community에 포함되는 것

방사형 게이지(모든 스타일), 선형 게이지(막대, 온도계, 불릿), 디지털 디스플레이(7 세그먼트), LED, LED 막대, 진행 링과 함께 범위, 임계값, 애니메이션, 밝은 테마와 어두운 테마, VCL과 FireMonkey 컨트롤, 헤드리스 PNG와 SVG 출력이 포함됩니다. 선형 게이지의 불릿 목표와 LED 막대의 피크는 추가 포인터가 아니라 속성이므로 Community에서도 동작합니다.

고급 기능

sgcVer.inc의 define SGC_GAUGES_ADVANCED가 이들을 켭니다. Community 에디션에서는 꺼져 있습니다.

- **여러 스케일과 포인터**: 첫 번째 이후의 스케일과 포인터는 컬렉션에 보관되므로 Registered 에디션으로 만든 폼도 불러올 수 있지만, Community는 이들을 그리거나 애니메이션하거나 임계값을 평가하지 않습니다.
- **네 가지 게이지 유형**: VU 미터, 시계, 나침반, 애넌시에이터, 그 유닛(sgcGauges_VUMeter, sgcGauges_Clock, sgcGauges_Compass, sgcGauges_Annunciator), 여덟 개의 컨트롤과 그 등록.
- **디지털 디스플레이**: 14 세그먼트와 도트 매트릭스 그리기. gdt14Segment와 gdtDotMatrix는 열거형에 남아 있으며 7 세그먼트로 그려집니다.
- **브리지**: sgcGaugesWS 패키지는 Community에서 빌드되거나 설치되지 않습니다.

모든 에디션을 위한 코드

고급 유형을 사용하는 코드는 define을 검사할 수 있으므로, 같은 유닛이 두 에디션에서 모두 빌드됩니다.

참고 항목

- 게이지 유형
- 설치
- 문제 해결

브리지 개요

sgcGaugesWS는 sgcGauges와 sgcWebSockets를 연결하는 브리지입니다. 소스는 sgcWebSockets 클라이언트가 받은 JSON 메시지를 읽고, 메시지 이벤트에 코드를 작성하지 않아도 값을 게이지에 씁니다. 브리지는 Registered 에디션의 일부이며 sgcWebSockets가 필요합니다. 전용 패키지 sgcGaugesWS와 dclsgcGaugesWS가 있고, 컴포넌트는 **SGC Gauges** 팔레트 페이지에 있습니다.

컴포넌트	유닛	클라이언트
TsgcGaugeFeedSource_MQTT	sgcGaugesWS_Source_MQTT	TsgcWSPClient_MQTT
TsgcGaugeFeedSource_WebSocket	sgcGaugesWS_Source_WebSocket	TsgcWebSocketClient
TsgcGaugeFeedSource_SSE	sgcGaugesWS_Source_SSE	TsgcHTTP1Client

소스가 동작하는 방식

소스는 Active가 True이고 Client가 지정되어 있는 동안, 실행 시 클라이언트에 자체 메시지 핸들러를 설치하고 이전에 있던 핸들러를 유지하므로, 애플리케이션의 핸들러도 소스 다음에 호출됩니다. Active를 False로 설정하거나, Client를 지우거나, 소스를 해제하면 이전 핸들러가 복원됩니다. 소스는 연결, 구독, 재연결을 하지 않습니다. 이는 애플리케이션이 클라이언트로 수행합니다.

바인딩

Bindings의 각 항목(TsgcGaugeFeedBinding, 유닛 sgcGaugesWS_Source)은 한 게이지의 포인터 하나에 값을 씁니다. 메시지는 한 번만 JSON으로 파싱되며 활성 상태인 모든 바인딩이 읽습니다.

속성	형식	설명
Gauge	TComponent	VCL 또는 FireMonkey 게이지 컨트롤. 다른 컴포넌트는 거부됩니다.
GaugeModel	TsgcCustomGauge	Public. 헤드리스 모델. Gauge보다 우선합니다. 모델을 해제하기 전에 지우십시오.
PointerIndex	Integer	값을 쓸 포인터. 기본값 0.
ValuePath	string	값의 JSON 경로. price 또는 data.items[0].v 등. 비어 있으면 메시지 전체를 읽습니다.
FilterPath	string	메시지를 선택하는 필드의 경로. 바인딩은 이 경로가 있는 메시지만 읽습니다.
FilterValue	string	비어 있지 않으면 FilterPath의 텍스트가 이 값과 같아야 합니다. ticker 등.
Topic	string	MQTT 전용: +와 #를 사용하는 토픽 필터. 비어 있으면 모든 퍼블리시를 읽습니다.
EventName	string	SSE 전용: 이벤트 유형. 비어 있으면 모든 이벤트를 읽습니다.
Active	Boolean	False이면 바인딩이 모든 메시지를 무시합니다. 기본값 True.

JSON 숫자, 소수점 구분자가 '.'인 숫자 문자열, 불리언 값(true는 1, false는 0)을 받아들입니다. 바인딩의 경로가 없는 메시지는 그 바인딩에서 건너뛰므로 부분 업데이트도 문제없습니다. 잘못된 JSON, 다른 종류의 값, 잘못된 포인터 인덱스는 OnError로 전달됩니다.

소스 속성과 이벤트

멤버	설명
Client	소스가 읽는 sgcWebSockets 클라이언트.
Active	False이면 소스가 모든 메시지를 무시합니다. 기본값 True.
Bindings	바인딩.
OnValue(Sender, aBinding, aValue)	바인딩이 값을 썼습니다.
OnError(Sender, aError)	메시지를 적용할 수 없었습니다.

스레드

값은 게이지의 스레드 안전한 SetValue로 메시지가 도착한 스레드에서 쓰이며, 게이지 컨트롤은 메인 스레드에서 다시 그리고 OnThreshold를 호출합니다. 한 메시지가 한 게이지에 쓰는 값은 하나의 변경이 됩니다. OnValue와 OnError는 소스의 잠금 아래에서 도착 스레드로 실행됩니다. 컨트롤에 접근하기 전에 그 안에서는 synchronize 대신 큐(TThread.Queue)를 사용하십시오. Bindings는 메인 스레드에서 변경하십시오.

빌드

검색 경로에서 sgcGauges와 sgcWebSockets 소스 폴더를 분리해 두십시오. 각 폴더에는 자체 sgcVer.inc가 있으며, 유닛은 자기 폴더의 것을 읽습니다.

예제

각 소스에는 클라이언트가 받은 것처럼 메시지를 적용하는 public 메서드가 있습니다. ProcessPublish(MQTT), ProcessMessage(WebSocket), ProcessChunk(SSE)입니다. 이들은 쓰인 값의 개수를 반환합니다. 이를 사용하면 클라이언트 없이 헤드리스 모델에 값을 공급할 수 있습니다.

참고 항목

- MQTT 소스
- WebSocket 소스
- 서버 전송 이벤트 소스
- TsgcCustomGauge

MQTT 소스

클래스: TsgcGaugeFeedSource_MQTT (유닛 sgcGaugesWS_Source_MQTT)
TsgcWSPClient_MQTT가 받은 퍼블리시로 게이지에 값을 공급합니다. Client는 MQTT 클라이언트입니다. 소스는 OnMQTTPublish 핸들러를 설치하고 이전 핸들러를 유지하며, 재연결을 거쳐도 혹은 계속 동작합니다.
각 바인딩의 Topic은 MQTT 토픽 필터입니다. +는 한 레벨과 일치하고 #는 해당 레벨과 그 아래의 모든 레벨과 일치합니다. 바인딩은 토픽이 일치하는 퍼블리시만 읽으며, Topic이 비어 있으면 모든 퍼블리시를 읽습니다. 소스는 구독하지 않으므로, 애플리케이션이 클라이언트에서 바인딩의 토픽을 포함하는 필터로 구독합니다. 모든 페이로드는 JSON으로 읽습니다.
ProcessPublish(aTopic, aText)는 클라이언트 없이 퍼블리시를 적용하고 쓰인 값의 개수를 반환합니다. 같은 유닛의 함수 sgcGaugeMQTTTopicMatch(aFilter, aTopic)는 토픽을 필터와 비교하여 검사합니다.

예제

sensors/<id>/temperature와 sensors/<id>/humidity 토픽에서 {"value": 21.5} 같은 페이로드로 값을 공급받는 두 개의 방사형 게이지:

참고 항목

- 브리지 개요
- WebSocket 소스
- 서버 전송 이벤트 소스

WebSocket 소스

클래스: TsgcGaugeFeedSource_WebSocket (유닛 sgcGaugesWS_Source_WebSocket)
TsgcWebSocketClient가 받은 텍스트 메시지로 게이지에 값을 공급합니다. Client는 WebSocket 클라이언트입니다. 소스는 OnMessage 핸들러를 설치하고 이전 핸들러를 유지하며, WatchDog의 재연결처럼 연결이 다시 맺어져도 혹은 계속 동작합니다. 모든 텍스트 메시지는 바인딩의 ValuePath, FilterPath, FilterValue로 클라이언트가 알리는 스레드에서 JSON으로 읽으므로, 클라이언트의 NotifyEvents 모드는 어느 것이든 안전합니다.
FilterPath와 FilterValue는 한 연결에서 시세와 체결 채널처럼 여러 종류를 전달하는 스트림의 메시지를 선택합니다. ProcessMessage(aText)는 클라이언트 없이 메시지를 적용하고 쓰인 값의 개수를 반환합니다.

예제

스트림의 시세 메시지로 값을 공급받는 가격 표시와 변동 막대:

참고 항목

- 브리지 개요
- MQTT 소스
- 서버 전송 이벤트 소스

서버 전송 이벤트 소스

클래스: TsgcGaugeFeedSource_SSE (유닛 sgcGaugesWS_Source_SSE)
TsgcHTTP1Client의 서버 전송 이벤트로 게이지에 값을 공급합니다. Client는 HTTP 클라이언트입니다. 소스는 OnSSEMessage 핸들러를 설치하고 이전 핸들러를 유지합니다. 애플리케이션은 Client.GetSSE로 스트림을 시작합니다.
소스는 스트림의 청크를 이벤트로 나눕니다. data: 줄은 줄 바꿈(LF)으로 연결되고, event: 줄은 이벤트 유형을 설정하며, 빈 줄이 이벤트를 끝냅니다. LF, CRLF, CR 줄 끝을 모두 받아들이며, 하나의 이벤트가 여러 청크에 걸쳐 나뉘어도 됩니다. 각 바인딩의 EventName은 읽을 이벤트 유형을 선택합니다. event: 줄이 없는 이벤트의 유형은 message이고, EventName이 비어 있으면 모든 이벤트를 읽습니다. 모든 이벤트의 데이터는 JSON으로 읽습니다.

메서드	설명
ResetStream	스트림이 반쯤 읽다 남긴 이벤트를 버립니다. 재연결 후 새 스트림을 시작하기 전에 호출하십시오.
ProcessChunk(aText)	클라이언트 없이 스트림의 청크를 적용하고 쓰인 값의 개수를 반환합니다.

예제

메트릭 스트림의 cpu와 queue 이벤트로 값을 공급받는 진행 링과 LED 막대:

참고 항목

- 브리지 개요
- MQTT 소스
- WebSocket 소스

.NET 개요

sgcGauges .NET은 Delphi sgcGauges 라이브러리를 관리 코드 C#으로 포팅한 것이며 래퍼가 아닙니다. 클래스는 **esegece.sgcGauges** 네임스페이스에 있으며 Delphi 이름을 유지합니다: **TsgcCustomGauge**, **TsgcRadialGauge**와 **TsgcLEDBarGauge** 같은 10가지 게이지 모델, **TsgcGaugeScale**, **TsgcGaugeRange**, **TsgcGaugePointer**, **TsgcGaugeThreshold**, **TsgcGaugeAnimation**, 게이지 접두사가 붙은 열거형(**grsHalf**, **gtdBoth**, **glsBullet**)입니다. 컨트롤은 모든 호스트에서 Delphi 명명 패턴을 따릅니다: **TsgcVCLRadialGauge**는 **TsgcWinFormsRadialGauge**, **TsgcWpfRadialGauge**, **TsgcAvaloniaRadialGauge**에 해당합니다. 게이지는 **sgcUICore**의 .NET 포트로 그려지며, 캔버스, 글꼴, 밝은 테마와 어두운 테마는 eSeGeCe 비주얼 제품군의 다른 제품과 같습니다.
Delphi와 마찬가지로 모든 컨트롤은 프레임워크에 독립적인 모델을 소유하며 **Gauge**로 노출합니다. 모델은 스케일, 범위, 포인터, 한계값, 애니메이션을 보유하고 임의의 sgcUICore 캔버스에 스스로 그리므로, 같은 모델 코드가 WinForms 폼, WPF나 Avalonia 창, UI가 없는 콘솔 프로그램에서 동작합니다. .NET 헤드리스 출력을 참조하십시오.

패키지

이 포트는 NuGet 패키지 6개로 나뉘어 있어 프로젝트는 사용하는 것만 가져옵니다.

패키지 ID	포함 내용	대상 프레임워크
esegece.sgcGauges	10가지 게이지 모델, 스케일, 범위, 포인터, 한계값, 애니메이션 클래스, 페인터, SVG 출력 (**SaveToSVG**). 네이티브 종속성이 없습니다. esegece.sgcUICore에 종속됩니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcGauges.Rendering	헤드리스 PNG 출력: **sgcGauges_Raster**의 확장 메서드 **SaveToPNG**와 **RenderToRaster**. esegece.sgcUICore.Rendering의 SkiaSharp 캔버스에 그리며 Windows, Linux, macOS에서 동작합니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcGauges.WinForms	**TsgcWinFormsCustomGauge**를 기반으로 한 **TsgcWinForms<Type>Gauge** 컨트롤 10개.	net472, net8.0-windows, net9.0-windows
esegece.sgcGauges.Wpf	**TsgcWpfCustomGauge**를 기반으로 한 **TsgcWpf<Type>Gauge** 컨트롤 10개.	net472, net8.0-windows, net9.0-windows
esegece.sgcGauges.Avalonia	**TsgcAvaloniaCustomGauge**를 기반으로 한 **TsgcAvalonia<Type>Gauge** 컨트롤 10개이며, Avalonia 11을 사용하여 Windows, Linux, macOS에서 동작합니다.	net8.0, net9.0
esegece.sgcGauges.sgcWebSockets	sgcWebSockets로의 브리지: **TsgcGaugeFeedSource_WebSocket**, **TsgcGaugeFeedSource_MQTT**, **TsgcGaugeFeedSource_SSE**. Registered 에디션 전용.	netstandard2.0, net8.0, net9.0

`PackageReference` 한 줄이면 호스트에 필요한 모든 것이 들어옵니다. WinForms 및 Avalonia 패키지는 esegece.sgcGauges.Rendering에 종속되고, 이 패키지는 esegece.sgcGauges에 종속됩니다. WPF 패키지는 esegece.sgcGauges에 종속되며 esegece.sgcUICore.Wpf를 통해 그립니다. NuGet 설치를 참조하십시오.

에디션

에디션	패키지 ID	포함 항목

Registered	esegece.sgcGauges, .Rendering, .WinForms, .Wpf, .Avalonia and .sgcWebSockets	모든 것: 호스트 세 곳과 헤드리스에서의 10가지 게이지 유형, 게이지당 여러 스케일과 포인터, 14세그먼트 및 도트 매트릭스 표시, sgcWebSockets 브리지.
Community	접미사 .Community가 붙은 같은 ID이며, 예를 들어 esegece.sgcGauges.WinForms.Community입니다. Community 브리지 패키지는 없습니다.	radial, linear, digital, LED, LED 막대, 진행 링 게이지. VU 미터, 시계, 나침반, 어넌시에이터는 Community 어셈블리에 포함되지 않습니다. 게이지의 첫 번째 스케일과 첫 번째 포인터만 그려지고, 애니메이션되고, 한계값이 검사됩니다. **gdt14Segment**와 **gdtDotMatrix**는 7세그먼트로 그려집니다.

이것은 Delphi 에디션과 같은 구분입니다. 에디션을 참조하십시오.

호스트

호스트	Delphi 대응	플랫폼
WinForms	VCL(TsgcVCL 컨트롤)	Windows, .NET Framework 4.7.2, .NET 8 및 9.
WPF	VCL과 FMX	Windows, .NET Framework 4.7.2, .NET 8 및 9. 컨트롤은 XAML에서 선언할 수 있습니다.
Avalonia	FMX(TsgcFMX 컨트롤)	Windows, Linux, macOS, .NET 8 및 9.
헤드리스	게이지 모델	대상 프레임워크의 모든 플랫폼: SVG는 esegece.sgcGauges, PNG는 esegece.sgcGauges.Rendering.

모든 컨트롤은 Delphi 대응 컨트롤과 같은 이름으로 속성을 게시하며, 기본 클래스에 **Gauge**, **Theme**, **OnThreshold**, **MaxFPS**, **SaveToPNG**, **SaveToSVG**가 더 있습니다. 클래스 이름은 .NET 클래스를, Delphi에서 달라지는 점은 Delphi와 .NET의 차이점을 참조하십시오.

참고 항목

- NuGet 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 데이터 바인딩
- .NET 헤드리스 출력
- .NET sgcWebSockets 브리지
- .NET 데모
- 플랫폼

NuGet 설치

패키지는 nuget.org에 있습니다. 호스트의 패키지를 추가하면 코어와 렌더링 패키지가 종속성으로 함께 들어옵니다.

Registered 에디션

```
dotnet add package esegece.sgcGauges.WinForms
dotnet add package esegece.sgcGauges.Wpf
dotnet add package esegece.sgcGauges.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcGauges.WinForms.Community
dotnet add package esegece.sgcGauges.Wpf.Community
dotnet add package esegece.sgcGauges.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcGauges.Wpf" Version="2026.10.0" />
</ItemGroup>
```

프로젝트에서는 Registered 또는 Community 패키지 중 하나만 참조하고, 둘을 함께 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있습니다.

프로젝트 설정

- WinForms: `net8.0-windows` 프로젝트에 `<UseWindowsForms>true</UseWindowsForms>`를 지정하거나 .NET Framework 4.7.2 프로젝트를 사용합니다.
- WPF: `net8.0-windows` 프로젝트에 `<UseWPF>true</UseWPF>`를 지정하거나 .NET Framework 4.7.2 프로젝트를 사용합니다. XAML에서 네임스페이스는 `xmlns:sgc="clr-namespace:esegece.sgcGauges;assembly=esegece.sgcGauges.Wpf"`로 매핑합니다.
- Avalonia: `net8.0` 또는 `net9.0` 프로젝트에 Avalonia 11.3의 **Avalonia.Desktop** 및 **Avalonia.Themes.Fluent** 패키지를 사용하며, 일반적인 Avalonia 데스크톱 애플리케이션과 같습니다. XAML에서 네임스페이스는 `xmlns:sgc="using:esegece.sgcGauges"`로 매핑합니다.
- 헤드리스: 모델과 SVG에는 esegece.sgcGauges만으로 충분합니다. PNG에는 esegece.sgcGauges.Rendering을 추가하며, SkiaSharp와 Windows 및 Linux용 네이티브 에셋이 함께 들어옵니다.
- sgcWebSockets의 라이브 데이터: **esegece.sgcGauges.sgcWebSockets**(Registered 에디션)를 추가합니다. 이 패키지는 esegece.sgcGauges와 esegece.sgcWebSockets에 종속됩니다. .NET sgcWebSockets 브리지를 참조하십시오.

C#에서는 모든 호스트에 `using esegece.sgcGauges;`가 필요합니다. `using esegece.sgcUICore;`는 **TsgcUITheme**, **TsgcUITextStyle** 같은 sgcUICore의 형식에 필요합니다.

참고 항목

- .NET 개요
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 데모

빠른 시작 WinForms

이 빠른 시작에서는 WinForms 게이지 3개가 있는 폼을 코드로 만듭니다. 빨간 띠와 시속 200 km의 한계값이 있는 속도계, 연료 막대, 한계값이 켜고 끄는 과속 램프입니다.

단계

1. `net8.0-windows` Windows Forms 프로젝트를 만들고 **esegece.sgcGauges.WinForms** 패키지(Community 에디션에서는 **esegece.sgcGauges.WinForms.Community**)를 추가합니다.
2. `using esegece.sgcGauges;`와 `using esegece.sgcUICore;`를 추가합니다.
3. 컨트롤을 만들어 폼에 추가합니다. 일반 WinForms 컨트롤이므로 Bounds, Dock, Anchor, Enabled가 평소처럼 동작합니다.
4. 먼저 스케일 범위를 설정한 다음 범위, 한계값, 값을 설정합니다. 값은 기록될 때 스케일에 맞게 클램프됩니다.
5. 이벤트를 할당합니다. 이벤트는 Delphi와 같은 이름이며 `=`로 할당합니다.

예제

```
using System;
using System.Drawing;
using System.Windows.Forms;
using esegece.sgcGauges;
using esegece.sgcUICore;

public class GaugesForm : Form
{
    public readonly TsgcWinFormsRadialGauge Speed;
    public readonly TsgcWinFormsLinearGauge Fuel;
    public readonly TsgcWinFormsLEDGauge Alarm;
    private readonly Timer FTimer;
    private readonly Random FRandom = new Random();

    public GaugesForm()
    {
        Text = "sgcGauges";
        ClientSize = new Size(620, 260);

        // a speedometer from 0 to 240 km/h with a red band and a threshold at 200
        Speed = new TsgcWinFormsRadialGauge { Bounds = new Rectangle(10, 10, 240, 240) };
        Speed.Title = "Speed";
        Speed.Units = "km/h";
        Speed.ValueFormat = "0";
        Speed.Scales[0].Maximum = 240;
        Speed.Scales[0].MajorInterval = 40;
        TsgcGaugeRange oRed = Speed.Scales[0].Ranges.Add();
        oRed.StartValue = 200;
        oRed.EndValue = 240;
        oRed.Color = 0xFFE34948;
        TsgcGaugeThreshold oLimit = Speed.Thresholds.Add();
        oLimit.Value = 200;
        oLimit.Direction = TsgcGaugeThresholdDirection.gtdBoth;
        oLimit.Hysteresis = 10;
        Speed.Animation.Duration = 400;
        Speed.Animation.Easing = TsgcGaugeEasing.geCubicOut;

        // a horizontal bar from 0 to 100
        Fuel = new TsgcWinFormsLinearGauge { Bounds = new Rectangle(260, 10, 350, 110) };
        Fuel.Title = "Fuel";
        Fuel.Units = "%";
        Fuel.LinearStyle = TsgcGaugeLinearStyle.glsBar;
        Fuel.Value = 70;
```

```
        // a blinking red lamp
        Alarm = new TsgcWinFormsLEDGauge { Bounds = new Rectangle(260, 130, 160, 110) };
        Alarm.Title = "Overspeed";
        Alarm.OnColor = 0xFFE34948;
        Alarm.Blink = true;

        // OnThreshold runs in the UI thread, whatever thread wrote the value
        Speed.OnThreshold = (Sender, aThreshold, aValue, aRising) => Alarm.Active = aRising;

        Controls.AddRange(new Control[] { Speed, Fuel, Alarm });

        // a new value moves the needle with the animation
        FTimer = new Timer { Interval = 1000 };
        FTimer.Tick += (s, e) => Speed.Value = FRandom.Next(0, 240);
        FTimer.Start();
    }

    public void SetDark(bool aDark)
    {
        TsgcUITheme vTheme = aDark ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        Speed.Theme = vTheme;
        Fuel.Theme = vTheme;
        Alarm.Theme = vTheme;
    }

    protected override void Dispose(bool disposing)
    {
        if (disposing)
            FTimer.Dispose();
        base.Dispose(disposing);
    }
}
```

참고

- **Value**는 **Pointers[0].Value**입니다. 어떤 스레드에서든 기록할 수 있으며, 컨트롤은 UI 스레드에서 다시 그리고 그곳에서 **OnThreshold**를 발생시킵니다.
- **Animation.Enabled**(기본값)에서는 바늘이 **Animation.Duration** 밀리초 동안 새 값으로 움직입니다. 값, 한계값, 이벤트는 즉시 바뀝니다. 움직이는 포인터만 초당 최대 **MaxFPS** 프레임으로 다시 그려집니다.
- **Theme**은 컨트롤마다 따로이며, 0으로 둔 색 속성은 테마의 색을 사용합니다.
- **SaveToPNG**와 **SaveToSVG**는 컨트롤이 표시하는 내용을 컨트롤 크기로 기록하며, 배율을 지정할 수도 있습니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- .NET 데이터 바인딩
- 빠른 시작 VCL

빠른 시작 WPF

이 빠른 시작에서는 속도계, 디지털 rpm 표시, 과속 램프를 XAML로 선언하고 두 값을 뷰 모델에 바인딩합니다.

단계

1. `net8.0-windows` WPF 프로젝트를 만들고 **esegece.sgcGauges.Wpf** 패키지(Community 에디션에서는 **esegece.sgcGauges.Wpf.Community**)를 추가합니다.
2. 창에서 네임스페이스를 매핑합니다: `xmlns:sgc="clr-namespace:esegece.sgcGauges;assembly=esegece.sgcGauges.Wpf"`.
3. 컨트롤을 선언합니다. **Title**, **Units**, **ValueFormat**, **Digits** 같은 단순한 속성은 특성으로 설정하며, **Value**는 종속성 속성이므로 바인딩을 받습니다.
4. 컬렉션(스케일, 범위, 한계값)을 채우고 코드 비하인드에서 이벤트를 할당합니다.

XAML

```
<Window x:Class="QuickStart.QuickStartWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcGauges;assembly=esegece.sgcGauges.Wpf"
        Title="sgcGauges" Width="680" Height="320">
  <StackPanel Orientation="Horizontal" Margin="8">
    <sgc:TsgcWpfRadialGauge x:Name="gauSpeed" Width="260" Height="260"
                            Title="Speed" Units="km/h" ValueFormat="0" Value="{Binding Speed}" />
    <StackPanel Margin="8,0,0,0">
      <sgc:TsgcWpfDigitalGauge x:Name="gauRpm" Width="340" Height="110"
                               Title="Engine" Units="rpm" ValueFormat="0" Digits="5" Value="{Binding Rpm}" />
      <sgc:TsgcWpfLEDGauge x:Name="ledAlarm" Width="160" Height="110" Title="Overspeed" Blink="True" />
    </StackPanel>
  </StackPanel>
</Window>
```

코드 비하인드

```
using System.ComponentModel;
using System.Windows;
using esegece.sgcGauges;

namespace QuickStart
{
    public class DashboardModel : INotifyPropertyChanged
    {
        private double FSpeed;
        private double FRpm;

        public event PropertyChangedEventHandler PropertyChanged;

        public double Speed
        {
            get { return FSpeed; }
            set { FSpeed = value; PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Speed))); }
        }

        public double Rpm
        {
            get { return FRpm; }
            set { FRpm = value; PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Rpm))); }
        }
    }
```

```
    public partial class QuickStartWindow : Window
    {
        public readonly DashboardModel Model = new DashboardModel();

        public QuickStartWindow()
        {
            InitializeComponent();
            // collections are filled in code: scales, ranges, pointers, thresholds
            gauSpeed.Scales[0].Maximum = 240;
            gauSpeed.Scales[0].MajorInterval = 40;
            TsgcGaugeRange oRed = gauSpeed.Scales[0].Ranges.Add();
            oRed.StartValue = 200;
            oRed.EndValue = 240;
            oRed.Color = 0xFFE34948;
            TsgcGaugeThreshold oLimit = gauSpeed.Thresholds.Add();
            oLimit.Value = 200;
            oLimit.Hysteresis = 10;
            oLimit.Direction = TsgcGaugeThresholdDirection.gtdBoth;
            gauRpm.Scales[0].Maximum = 99999;
            ledAlarm.OnColor = 0xFFE34948;
            // in the UI thread, also when the value comes from another thread
            gauSpeed.OnThreshold = (Sender, aThreshold, aValue, aRising) => ledAlarm.Active = aRising;
            DataContext = Model;
        }
    }
}
```

참고

- 각 컨트롤에는 종속성 속성이 하나 있습니다. radial, linear, digital, LED, LED 막대, 진행 링, VU 미터 게이지에는 **ValueProperty**, 나침반에는 **HeadingProperty**입니다. 시계와 어넌시에이터에는 없습니다. 나머지 속성은 CLR 속성입니다. .NET 데이터 바인딩을 참조하십시오.
- **OnThreshold**는 바인딩된 값이 다른 스레드에서 설정될 때도 UI 스레드에서 실행됩니다.
- **Theme**은 컨트롤마다 따로입니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- .NET 데이터 바인딩
- 빠른 시작 FMX

빠른 시작 Avalonia

이 빠른 시작에서는 Avalonia 창을 코드로 만듭니다. 압력용 반원 다이얼, 진행 링, 녹색, 호박색, 빨간색 띠가 있는 LED 막대를 모두 어두운 테마로 배치합니다. 같은 코드가 Windows, Linux, macOS에서 동작합니다.

단계

1. `net8.0` 프로젝트를 만들고 Avalonia 11.3의 **Avalonia.Desktop** 및 **Avalonia.Themes.Fluent** 패키지와 **esegece.sgcGauges.Avalonia**(Community 에디션에서는 **esegece.sgcGauges.Avalonia.Community**)를 추가합니다.
2. `using esegece.sgcGauges;`와 `using esegece.sgcUICore;`를 추가합니다.
3. 컨트롤을 만들고 스케일과 값을 설정한 다음 창의 패널에 추가합니다.

예제

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;
using esegece.sgcGauges;
using esegece.sgcUICore;

public class App : Application
{
    public override void Initialize()
    {
        Styles.Add(new FluentTheme());
    }

    public override void OnFrameworkInitializationCompleted()
    {
        if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime vDesktop)
            vDesktop.MainWindow = new GaugesWindow();
        base.OnFrameworkInitializationCompleted();
    }
}

public class GaugesWindow : Window
{
    public readonly TsgcAvaloniaRadialGauge Pressure;
    public readonly TsgcAvaloniaProgressRingGauge Upload;
    public readonly TsgcAvaloniaLEDBarGauge Signal;

    public GaugesWindow()
    {
        Title = "sgcGauges";
        Width = 700;
        Height = 300;

        // a half dial from 0 to 10 bar
        Pressure = new TsgcAvaloniaRadialGauge { Width = 260, Height = 260 };
        Pressure.Title = "Pressure";
        Pressure.Units = "bar";
        Pressure.ValueFormat = "0.0";
        Pressure.RadialStyle = TsgcGaugeRadialStyle.grsHalf;
        Pressure.Scales[0].Maximum = 10;
        Pressure.Scales[0].MajorInterval = 2;
        Pressure.Value = 6.5;
```

```
        Upload = new TsgcAvaloniaProgressRingGauge { Width = 180, Height = 180 };
        Upload.Title = "Upload";
        Upload.Value = 72;

        // twenty segments, green, amber and red bands
        Signal = new TsgcAvaloniaLEDBarGauge { Width = 220, Height = 90 };
        Signal.Title = "Signal";
        Signal.Segments = 20;
        AddBand(Signal.Scales[0], 0, 70, 0xFF2EAD6B);
        AddBand(Signal.Scales[0], 70, 90, 0xFFF0A030);
        AddBand(Signal.Scales[0], 90, 100, 0xFFE34948);
        Signal.Value = 65;

        StackPanel pnlMain = new StackPanel { Orientation = Avalonia.Layout.Orientation.Horizontal };
        pnlMain.Children.Add(Pressure);
        pnlMain.Children.Add(Upload);
        pnlMain.Children.Add(Signal);
        Content = pnlMain;
        foreach (TsgcAvaloniaCustomGauge oGauge in new TsgcAvaloniaCustomGauge[] { Pressure, Upload, Signal })
            oGauge.Theme = TsgcUITheme.utDark;
    }

    private static void AddBand(TsgcGaugeScale aScale, double aStart, double aEnd, uint aColor)
    {
        TsgcGaugeRange oRange = aScale.Ranges.Add();
        oRange.StartValue = aStart;
        oRange.EndValue = aEnd;
        oRange.Color = aColor;
    }
}

public static class Program
{
    [STAThread]
    public static void Start(string[] args)
    {
        AppBuilder.Configure<App>().UsePlatformDetect().StartWithClassicDesktopLifetime(args);
    }
}
```

참고

- 컨트롤은 `xmlns:sgc="using:esegece.sgcGauges"`를 사용하여 XAML에서 선언할 수도 있습니다. 예를 들어 `<sgc:TsgcAvaloniaRadialGauge Title="Pressure" Value="{Binding Pressure}" />`입니다.
- **ValueProperty**(모든 컨트롤), **ActiveProperty**(LED), **HeadingProperty**(나침반)는 기본적으로 양방향으로 바인딩되는 스타일 속성입니다. .NET 데이터 바인딩을 참조하십시오.
- 컨트롤의 애니메이션 타이머는 비주얼 트리를 벗어나면 멈추므로, 컨트롤에는 Dispose가 필요하지 않습니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- .NET 데이터 바인딩
- 빠른 시작 FMX

.NET 클래스

모델 클래스는 Delphi와 이름이 같으며 **esegece.sgcGauges** 네임스페이스에 있습니다. 컨트롤은 VCL 및 FMX 접두사를 호스트 이름으로 바꾼 것입니다.

게이지와 컨트롤

모델	Delphi 컨트롤	WinForms	WPF
TsgcRadialGauge	TsgcVCLRadialGauge, TsgcFMXRadialGauge	TsgcWinFormsRadialGauge	TsgcWpfRadialGau
TsgcLinearGauge	TsgcVCLLinearGauge, TsgcFMXLinearGauge	TsgcWinFormsLinearGauge	TsgcWpfLinearGau
TsgcDigitalGauge	TsgcVCLDigitalGauge, TsgcFMXDigitalGauge	TsgcWinFormsDigitalGauge	TsgcWpfDigitalGau
TsgcLEDGauge	TsgcVCLLEDGauge, TsgcFMXLEDGauge	TsgcWinFormsLEDGauge	TsgcWpfLEDGauge
TsgcLEDBarGauge	TsgcVCLLEDBarGauge, TsgcFMXLEDBarGauge	TsgcWinFormsLEDBarGauge	TsgcWpfLEDBarGa
TsgcProgressRingGauge	TsgcVCLProgressRingGauge, TsgcFMXProgressRingGauge	TsgcWinFormsProgressRingGauge	TsgcWpfProgressR
TsgcVUMeterGauge (Registered)	TsgcVCLVUMeterGauge, TsgcFMXVUMeterGauge	TsgcWinFormsVUMeterGauge	TsgcWpfVUMeterG
TsgcClockGauge (Registered)	TsgcVCLClockGauge, TsgcFMXClockGauge	TsgcWinFormsClockGauge	TsgcWpfClockGaug
TsgcCompassGauge (Registered)	TsgcVCLCompassGauge, TsgcFMXCompassGauge	TsgcWinFormsCompassGauge	TsgcWpfCompassG
TsgcAnnunciatorGauge (Registered)	TsgcVCLAnnunciatorGauge, TsgcFMXAnnunciatorGauge	TsgcWinFormsAnnunciatorGauge	TsgcWpfAnnunciato

Registered의 네 가지 유형과 해당 컨트롤은 Community 어셈블리에 포함되지 않습니다. 각 유형이 그리는 내용은 게이지 유형을 참조하십시오.

기본 클래스

클래스	패키지	설명
TsgcCustomGauge	esegece.sgcGauges	모델의 기본 클래스: **Scales**, **Pointers**, **Thresholds**, **Animation**, **Theme**, **Title**, **Units**, **ValueFormat**, **FormatSettings**, **Value**, **SetValue**, **ValueToText**, **BeginUpdate**와 **EndUpdate**, **Render**, **RenderStatic**, **RenderDynamic**, **SaveToSVG**, 그리고 이벤트 **OnChange**, **OnThreshold**, **OnFormatValue**.

TsgcWinFormsCustomGauge	esegece.sgcGauges.WinForms	**TsgcUIWinFormsControl**을 기반으로 한 WinForms 컨트롤의 기본 클래스.
TsgcWpfCustomGauge	esegece.sgcGauges.Wpf	**TsgcUIWpfControl**을 기반으로 한 WPF 컨트롤의 기본 클래스.
TsgcAvaloniaCustomGauge	esegece.sgcGauges.Avalonia	**TsgcUIAvaloniaControl**을 기반으로 한 Avalonia 컨트롤의 기본 클래스로, **ValueProperty**, **LogicalWidth()**, **LogicalHeight()**가 있습니다.
IsgcGaugeHost	esegece.sgcGauges	모든 컨트롤이 구현합니다. **GetGauge()**는 모델을 반환합니다. 브리지는 이를 구현하는 모든 개체를 구동합니다.
sgcGauges_Raster	esegece.sgcGauges.Rendering	**TsgcCustomGauge**의 확장 메서드 **SaveToPNG**와 **RenderToRaster**.

모델 구성 요소

클래스	설명
TsgcGaugeScale, TsgcGaugeScales	스케일: **Minimum**, **Maximum**, **Mapping**, **MajorInterval**, **MinorCount**, **TickPlacement**, **LabelFormat**, **Ranges**.
TsgcGaugeRange, TsgcGaugeRanges	스케일의 색 띠: **StartValue**, **EndValue**, **Color**, **Thickness**.
TsgcGaugePointer, TsgcGaugePointers	바늘, 마커, 막대 또는 범위 포인터: **Kind**, **ScaleIndex**, **Value**, **EndValue**, **Origin**, **Color**, **NeedleStyle**, **MarkerShape**.
TsgcGaugeThreshold, TsgcGaugeThresholds	**OnThreshold**를 발생시키는 한계값: **Value**, **PointerIndex**, **Direction**, **Hysteresis**, **Color**, **ValueColor**, **Enabled**, **Active**.
TsgcGaugeAnimation	포인터 애니메이션: **Enabled**, 밀리초 단위의 **Duration**, **Easing**.
TsgcGaugeAnnunciatorTile, TsgcGaugeAnnunciatorTiles	어넌시에이터의 타일: **Caption**, **State**, **Acknowledged**.
TsgcGaugeFormatSettings	레이블의 **DecimalSeparator**와 **ThousandSeparator**를 담은 구조체이며, 기본값은 '.'와 ','입니다.

델리게이트

델리게이트	시그니처
TsgcGaugeChangeEvent	`(object Sender, TsgcGaugeChangeKind aKind)`
TsgcGaugeThresholdEvent	`(object Sender, TsgcGaugeThreshold aThreshold, double aValue, bool aRising)`
TsgcGaugeFormatValueEvent	`(object Sender, int aPointer, double aValue, ref string aText)`
TsgcGaugeTileClickEvent	`(object Sender, int aIndex)`, 어넌시에이터 컨트롤의 **OnTileClick**

브리지의 클래스는 .NET sgcWebSockets 브리지에 나와 있습니다.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- TsgcCustomGauge
- 게이지 유형

Delphi와 .NET의 차이점

이 포트는 Delphi의 이름, 기본값, 동작을 유지하므로 이 도움말의 Delphi 항목은 .NET에도 적용됩니다. 다음은 언어와 프레임워크에서 비롯되는 차이점입니다.

언어

- **이벤트**는 Delphi와 같은 이름의 델리게이트 속성입니다(**OnThreshold**, **OnFormatValue**, **OnChange**, **OnTileClick**). Delphi와 마찬가지로 이벤트마다 핸들러는 하나이며 `=`로 할당합니다. `var` 매개변수는 **OnFormatValue**에서 `ref` 매개변수가 됩니다. 컨트롤의 이벤트에서 Sender는 컨트롤입니다.
- **매개변수가 없는 함수**는 메서드로 남습니다: **ActiveScaleCount()**, **ActivePointerCount()**, **NowMs()**, **GetGauge()**, **TsgcGaugeAnnunciatorTile.Flashing()**.
- **FormatSettings**는 구조체 **TsgcGaugeFormatSettings**입니다. 읽고, 변경한 다음 다시 할당합니다.
- **색**은 ARGB 순서의 `uint` 값입니다(`0xFFE34948`). 0은 Delphi와 마찬가지로 테마의 색을 뜻합니다.
- **컬렉션**은 **TsgcGaugeCollection<T>**에서 파생되며 인덱서(`Scales[0]`), **Add**, **Delete**, **Clear**, **Count**, **BeginUpdate**와 **EndUpdate**, `foreach`를 제공합니다. **Assign(object)**은 컬렉션이나 항목을 복사합니다.
- Delphi 유닛의 독립 함수는 정적 메서드입니다: **sgcGauges_Types**(수학과 이징), **sgcGauges_Render**(지오메트리), **sgcGauges_Raster**(PNG).
- **Free**는 없습니다. 모델은 일반 클래스입니다. WinForms와 WPF 컨트롤은 각 프레임워크를 따르고, Avalonia 컨트롤은 비주얼 트리를 벗어나면 타이머를 멈춥니다.

예제

```
using System;
using esegece.sgcGauges;

public static class DifferencesSample
{
    public static TsgcRadialGauge Create()
    {
        TsgcRadialGauge oOven = new TsgcRadialGauge { Title = "Oven", Units = "C", ValueFormat = "0.0" };
        oOven.Scales[0].Maximum = 300;
        TsgcGaugeThreshold oHot = oOven.Thresholds.Add();
        oHot.Value = 250;
        oHot.Direction = TsgcGaugeThresholdDirection.gtdRising;
        oHot.Hysteresis = 5;
        oHot.Color = 0xFFE34948;

        // events are delegate properties: assign them with =, as in Delphi
        oOven.OnThreshold = (Sender, aThreshold, aValue, aRising) =>
            Console.WriteLine("oven above " + aThreshold.Value + ": " + aValue);
        // the var parameter of the Delphi event is a ref parameter
        oOven.OnFormatValue = (object Sender, int aPointer, double aValue, ref string aText) =>
        {
            if (aValue >= 250)
                aText = aText + " HOT";
        };

        // TsgcGaugeFormatSettings is a struct: read it, change it, assign it back
        TsgcGaugeFormatSettings vFormat = oOven.FormatSettings;
        vFormat.DecimalSeparator = ',';
        oOven.FormatSettings = vFormat;

        // Delphi functions without parameters stay methods
        Console.WriteLine(oOven.ActiveScaleCount() + " scale, " + oOven.ActivePointerCount() + " pointer");
        return oOven;
    }
}
```

출력

- **SaveToSVG**는 esegece.sgcGauges에 있는 모델의 메서드입니다. 모델의 **SaveToPNG**는 PNG에 SkiaSharp 래스터가 필요하므로 esegece.sgcGauges.Rendering에 있는 확장 메서드입니다. 컨트롤에는 둘 다 있으며, 각자의 크기로 기록합니다.
- 게이지의 SVG는 같은 게이지의 Delphi SVG와 바이트 단위로 같습니다. PNG는 SkiaSharp가 그리므로 가장자리의 안티앨리어싱이 Delphi PNG와 다를 수 있습니다.

스레드와 시간

- **SetValue**와 **Value**는 Delphi와 마찬가지로 스레드에 안전합니다. 컨트롤은 **TThread.Queue** 대신 생성 시 가져온 **SynchronizationContext**를 통해 UI 스레드에서 다시 그리고 **OnThreshold**를 발생시킵니다. 컨트롤이 없는 모델에서는 **OnThreshold**가 값을 기록한 스레드에서 실행됩니다.
- 애니메이션 시계는 **sgcGaugeNowMs** 대신 **Stopwatch**이며, **Clock**으로 어떤 **TsgcGaugeClockFunc**로도 바꿀 수 있습니다.

포팅되지 않은 항목

- .dfm 스트리밍 도우미(**Is*Stored** 함수, ZeroValues 정의 속성, **GetNamePath**)는 .NET에 대응하는 것이 없습니다. 컬렉션은 코드에서 채웁니다.

참고 항목

- .NET 개요
- .NET 클래스
- .NET 헤드리스 출력
- TsgcCustomGauge
- 에디션

.NET 데이터 바인딩

게이지는 값을 표시하므로, 데이터 바인딩은 애플리케이션 개체의 값을 컨트롤로 가져오는 것을 뜻합니다. 각 호스트는 컨트롤의 값 속성에 대해 자체 바인딩 시스템으로 이를 수행합니다. 스케일, 범위, 포인터, 한계값은 컬렉션이며 코드에서 설정합니다.

호스트	바인딩 가능한 속성	기본 방향
WinForms	**Control.DataBindings**를 통한 **Value**(및 모든 공개 속성)	소스에서 컨트롤로 (**DataSourceUpdateMode.Never**). 컨트롤은 변경 이벤트를 발생시키지 않습니다.
WPF	종속성 속성 **ValueProperty**(radial, linear, digital, LED, LED 막대, 진행 링, VU 미터)와 **HeadingProperty**(나침반)	단방향. **BindingMode.TwoWay**에서는 모델이 클램프하거나 순환시킨 값이 소스에 다시 기록됩니다.
Avalonia	스타일 속성 **ValueProperty**(모든 컨트롤), **ActiveProperty**(LED), **HeadingProperty**(나침반)	양방향: 모델이 클램프하거나 순환시킨 값은 소스에 다시 기록됩니다.

소스는 어떤 스레드에서든 바뀔 수 있습니다. 모델은 스레드에 안전하며 컨트롤은 UI 스레드에서 다시 그려집니다. WinForms 바인딩 자체는 다른 WinForms 컨트롤과 마찬가지로 UI 스레드에서 발생시켜야 합니다.

WinForms

```
using System.ComponentModel;
using System.Windows.Forms;
using esegece.sgcGauges;

// any object with INotifyPropertyChanged can feed a gauge
public class EngineData : INotifyPropertyChanged
{
    private double FRpm;

    public event PropertyChangedEventHandler PropertyChanged;

    public double Rpm
    {
        get { return FRpm; }
        set
        {
            if (FRpm == value)
                return;
            FRpm = value;
            PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Rpm)));
        }
    }
}

public static class DataBindingSample
{
    public static void Bind(TsgcWinFormsRadialGauge aGauge, EngineData aEngine)
    {
        aGauge.Title = "Engine";
        aGauge.Units = "rpm";
        aGauge.ValueFormat = "0";
        aGauge.Scales[0].Maximum = 8000;
        aGauge.Scales[0].MajorInterval = 1000;
        // a gauge only shows the value: from the source to the control, never back
        aGauge.DataBindings.Add(nameof(aGauge.Value), aEngine, nameof(aEngine.Rpm), false,
            DataSourceUpdateMode.Never);
```

```
    }
}
```

WPF

XAML에서는: `<sgc:TsgcWpfRadialGauge Value="{Binding Speed}" />`. 빠른 시작 WPF를 참조하십시오. 코드에서는:

```
using System.ComponentModel;
using System.Windows.Data;
using esegece.sgcGauges;

public class TankModel : INotifyPropertyChanged
{
    private double FLevel;

    public event PropertyChangedEventHandler PropertyChanged;

    public double Level
    {
        get { return FLevel; }
        set { FLevel = value; PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Level))); }
    }
}

public static class DataBindingWpfSample
{
    public static void Bind(TsgcWpfLinearGauge aGauge, TankModel aTank)
    {
        aGauge.Title = "Tank";
        aGauge.Units = "%";
        aGauge.Orientation = TsgcGaugeOrientation.gloVertical;
        aGauge.Scales[0].Maximum = 100;
        // ValueProperty is one way by default; TwoWay also writes the clamped value back
        BindingOperations.SetBinding(aGauge, TsgcWpfLinearGauge.ValueProperty,
            new Binding(nameof(TankModel.Level)) { Source = aTank, Mode = BindingMode.TwoWay });
    }
}
```

Avalonia

```
using System.ComponentModel;
using Avalonia.Data;
using esegece.sgcGauges;

public class PumpModel : INotifyPropertyChanged
{
    private double FFlow;
    private bool FRunning;

    public event PropertyChangedEventHandler PropertyChanged;

    public double Flow
    {
        get { return FFlow; }
        set { FFlow = value; PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Flow))); }
    }

    public bool Running
    {
        get { return FRunning; }
        set { FRunning = value; PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(nameof(Running))); }
    }
}
```

```
public static class DataBindingAvaloniaSample
{
    public static void Bind(TsgcAvaloniaDigitalGauge aFlow, TsgcAvaloniaLEDGauge aPump, PumpModel aModel)
    {
        aFlow.Title = "Flow";
        aFlow.Units = "m3/h";
        aFlow.ValueFormat = "0.0";
        aFlow.Scales[0].Maximum = 500;
        // ValueProperty and ActiveProperty are styled properties, two way by default
        aFlow.Bind(TsgcAvaloniaCustomGauge.ValueProperty, new Binding(nameof(PumpModel.Flow)) { Source = aModel });
        aPump.Title = "Pump";
        aPump.Bind(TsgcAvaloniaLEDGauge.ActiveProperty, new Binding(nameof(PumpModel.Running)) { Source = aModel });
    }
}
```

피드

뷰 모델 없이 WebSocket, MQTT 또는 서버 전송 이벤트 메시지로 게이지를 구동하려면, 게이지를 브리지의 피드 소스에 바인딩합니다. .NET sgcWebSockets 브리지를 참조하십시오.

참고 항목

- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET sgcWebSockets 브리지

.NET 헤드리스 출력

모든 게이지 모델은 컨트롤 없이 스스로 그릴 수 있으므로, 콘솔 프로그램, 서비스, 웹 서버가 Windows, Linux, macOS에서 게이지의 PNG와 SVG 이미지를 기록할 수 있습니다. 이것은 헤드리스 출력의 .NET 형태입니다.

메서드

메서드	패키지	설명
SaveToSVG(Stream or string, int aWidth, int aHeight, double aScale = 1)	esegece.sgcGauges	게이지의 SVG 문서를 기록합니다. 네이티브 종속성이 없습니다.
SaveToPNG(Stream or string, int aWidth, int aHeight, double aScale = 1)	esegece.sgcGauges.Rendering	**sgcGauges_Raster**의 확장 메서드: 지워진 SkiaSharp 래스터에 Round(aWidth * aScale) x Round(aHeight * aScale) 픽셀로 그리고 PNG를 기록합니다.
RenderToRaster(int aWidth, int aHeight, double aScale = 1, double aNowMs = -1)	esegece.sgcGauges.Rendering	게이지가 그려진 **TsgcUICanvas_Raster**를 반환하므로, 더 그리거나 픽셀을 읽을 수 있습니다. 호출자가 해제합니다.
Render(TsgcUICanvas aCanvas, int aWidth, int aHeight, double aScale = 1, double aNowMs = -1)	esegece.sgcGauges	임의의 sgcUICore 캔버스에 그립니다.

예제

```
using System.IO;
using esegece.sgcGauges;
using esegece.sgcUICore;

public static class HeadlessSample
{
    public static void Export(string aFolder)
    {
        // a gauge model needs no control, no form and no UI thread
        TsgcRadialGauge oSpeed = new TsgcRadialGauge();
        oSpeed.Title = "Speed";
        oSpeed.Units = "km/h";
        oSpeed.ValueFormat = "0";
        oSpeed.Theme = TsgcUITheme.utDark;
        // set the scale range first: a value is clamped when it is written
        oSpeed.Scales[0].Maximum = 240;
        oSpeed.Scales[0].MajorInterval = 40;
        TsgcGaugeRange oRed = oSpeed.Scales[0].Ranges.Add();
        oRed.StartValue = 200;
        oRed.EndValue = 240;
        oRed.Color = 0xFFE34948;
        // false: no animation, a headless render always draws the target value
        oSpeed.SetValue(0, 88, false);

        // SVG comes with esegece.sgcGauges, PNG with esegece.sgcGauges.Rendering
        oSpeed.SaveToSVG(Path.Combine(aFolder, "speed.svg"), 200, 200);
        oSpeed.SaveToPNG(Path.Combine(aFolder, "speed.png"), 200, 200);
        // the same layout with twice the pixels, for a high DPI screen or a report
```

```
        oSpeed.SaveToPNG(Path.Combine(aFolder, "speed@2x.png"), 200, 200, 2);

        // any gauge type, to any stream
        TsgcDigitalGauge oMains = new TsgcDigitalGauge { Title = "Mains", Units = "Hz", ValueFormat = "0.00", Digits = 6 };
        oMains.SetValue(0, 50, false);
        using (FileStream oStream = File.Create(Path.Combine(aFolder, "mains.png")))
            oMains.SaveToPNG(oStream, 400, 110);
    }
}
```

참고

- 헤드리스 렌더링은 애니메이션의 프레임이 아니라 목표값을 그리므로, 같은 게이지의 출력은 항상 같은 바이트입니다. **Live**를 false로 설정한 시계는 **Time**을 표시합니다.
- aScale은 픽셀 수를 곱하고 레이아웃은 유지합니다. 200 x 200을 배율 2로 하면 같은 그림의 400 x 400 PNG가 됩니다.
- Linux에서는 esegece.sgcGauges.Rendering이 네이티브 SkiaSharp 에셋을 가져옵니다. X 서버는 필요하지 않습니다.

ConsoleRender 데모는 게이지 유형마다 PNG와 SVG 파일을 기록합니다. .NET 데모를 참조하십시오.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- .NET 데모
- 헤드리스 출력
- 빠른 시작 헤드리스

.NET sgcWebSockets 브리지

패키지 **esegece.sgcGauges.sgcWebSockets**(Registered 에디션)는 sgcGaugesWS 브리지의 .NET 포트입니다. 피드 소스는 sgcWebSockets .NET의 클라이언트에 연결되어 모든 메시지를 JSON으로 읽고, 바인딩의 값을 게이지에 기록합니다. 동작은 브리지 개요에서 설명한 것과 같습니다.

클래스	클라이언트	읽는 대상
TsgcGaugeFeedSource_WebSocket	**Client**: **TsgcWebSocketClient**와 해당 **OnMessage** 이벤트	모든 텍스트 메시지. **ProcessMessage(string)**은 메시지를 직접 적용합니다.
TsgcGaugeFeedSource_MQTT	**Client**: **TsgcWSProtocol_MQTT_Client**와 해당 publish 이벤트	바인딩의 **Topic** 필터와 토픽이 일치하는 publish의 페이로드(`+` 및 `#` 와일드카드). **ProcessPublish(topic, text)**는 하나를 직접 적용합니다.
TsgcGaugeFeedSource_SSE	**Client**: **TsgcHTTP1Client**와 해당 **OnSSEMessage** 이벤트	유형이 바인딩의 **EventName**과 같은 이벤트의 데이터. **ProcessChunk(string)**과 **ResetStream()**은 스트림을 직접 적용합니다.

모든 소스에는 **Active**, **Bindings**, **OnValue**, **OnError**가 있습니다. **TsgcGaugeFeedBinding**에는 **Gauge**(모든 컨트롤 또는 **IsgcGaugeHost**를 구현하는 모든 개체), **GaugeModel**(컨트롤이 없는 모델), **PointerIndex**, **ValuePath**, **FilterPath**, **FilterValue**, **Topic**, **EventName**, **Active**가 있습니다.

예제

```
using System;
using esegece.sgcGauges;
using esegece.sgcWebSockets;

public class MetricsFeed : IDisposable
{
    public readonly TsgcWebSocketClient Client = new TsgcWebSocketClient();
    public readonly TsgcGaugeFeedSource_WebSocket Feed = new TsgcGaugeFeedSource_WebSocket();

    public MetricsFeed(TsgcWinFormsRadialGauge aCpu, TsgcWinFormsLinearGauge aQueue)
    {
        Feed.Client = Client;
        // {"type":"cpu","value":42.5} moves the CPU gauge
        TsgcGaugeFeedBinding oCpu = Feed.Bindings.Add();
        oCpu.Gauge = aCpu;
        oCpu.FilterPath = "type";
        oCpu.FilterValue = "cpu";
        oCpu.ValuePath = "value";
        // {"type":"queue","data":{"length":120}} moves the queue gauge
        TsgcGaugeFeedBinding oQueue = Feed.Bindings.Add();
        oQueue.Gauge = aQueue;
        oQueue.FilterPath = "type";
        oQueue.FilterValue = "queue";
        oQueue.ValuePath = "data.length";
        // OnValue and OnError run in the thread of the client, not in the UI thread
        Feed.OnError = (Sender, aError) => Console.WriteLine("feed: " + aError);
    }

    // the source never connects: the application owns the client
    public bool Connect(string aHost, int aPort)
```

```
    {
        Client.Host = aHost;
        Client.Port = aPort;
        return Client.Connect(5000);
    }

    public void Dispose()
    {
        Feed.Dispose();
        if (Client.Active)
            Client.Active = false;
    }
}

public static class MQTTFeedSample
{
    public static TsgcGaugeFeedSource_MQTT Create(TsgcWSProtocol_MQTT_Client aMQTT, TsgcWinFormsDigitalGauge aOven)
    {
        TsgcGaugeFeedSource_MQTT oFeed = new TsgcGaugeFeedSource_MQTT();
        oFeed.Client = aMQTT;
        // a publish of {"value":180.5} on plant/oven1/temperature
        TsgcGaugeFeedBinding oBinding = oFeed.Bindings.Add();
        oBinding.Gauge = aOven;
        oBinding.Topic = "plant/+/temperature";
        oBinding.ValuePath = "value";
        // the source does not subscribe: subscribe when the session is up
        aMQTT.OnMQTTConnect += (Connection, aSession, aReasonCode, aReasonName, aProperties) =>
            aMQTT.Subscribe("plant/#");
        return oFeed;
    }
}
```

참고

- 소스는 연결하지도 구독하지도 않습니다. 클라이언트 연결, MQTT 토픽 구독, HTTP 클라이언트의 **GetSSE** 호출은 애플리케이션이 담당합니다.
- 클라이언트 이벤트는 .NET 이벤트입니다. 소스는 `+=`로 자체 핸들러를 추가하고 그 핸들러만 제거하므로, 애플리케이션의 핸들러는 계속 동작하며 추가된 순서대로 실행됩니다.
- 값은 클라이언트의 스레드에서 기록됩니다. 게이지는 스레드에 안전하며 UI 스레드에서 다시 그려집니다. **OnValue**와 **OnError**는 클라이언트 스레드에서 실행되므로, 컨트롤을 다루기 전에 UI 스레드로 전달하십시오.
- 바인딩은 WinForms 컨트롤의 **Disposed** 이벤트를 감시하고, 컨트롤이 사라지면 **Gauge**를 비웁니다. WPF 및 Avalonia 컨트롤과 **GaugeModel**의 경우에는 게이지가 사라지기 전에 바인딩을 비우십시오.
- 숫자 문자열은 문화권과 관계없이 소수점을 '.'으로 읽습니다. true와 false는 1과 0으로 읽습니다.

MQTTLive 데모는 작은 프로세스 내 브로커를 사용해 게이지 6개를 MQTT로 구동합니다. .NET 데모를 참조하십시오.

참고 항목

- 브리지 개요
- MQTT 소스
- WebSocket 소스
- 서버 전송 이벤트 소스
- .NET 데이터 바인딩
- .NET 데모

.NET 데모

.NET 데모는 sgcUI .NET 폴더의 `demos\Gauges`에 있으며, 데모마다 프로젝트 파일, Program.cs, README.md가 들어 있는 폴더가 하나씩 있습니다. 사용자 인터페이스는 코드로 만듭니다.

데모	호스트	보여 주는 내용
Gallery	WinForms	하나의 폼에 10가지 게이지 컨트롤: 테마 전환, 애니메이션 켜기와 끄기, 값 시뮬레이션, 2배 크기의 PNG 내보내기. 시속 200 km의 한계값에서 경보 LED와 HIGH SPEED 타일이 전환되며, 타일을 클릭하면 확인 처리됩니다.
GalleryWpf	WPF	TsgcWpf 컨트롤로 만든 같은 갤러리.
GalleryAvalonia	Avalonia	TsgcAvalonia 컨트롤로 만든 같은 갤러리.
HMIPanel	WinForms	탱크 2개와 펌프 2개가 있는 모의 공정: 선형 수위 게이지, 원형 압력 게이지, 디지털 유량 표시, 펌프 LED, 공유된 하나의 **OnThreshold** 핸들러가 켜는 어넌시에이터.
MQTTLive	WinForms	**TsgcGaugeFeedSource_MQTT**가 구동하는 게이지 6개. 루프백 옵션을 쓰면 Connect가 프로세스 안에서 작은 MQTT 브로커를 시작하므로, 데모에는 브로커도 네트워크도 필요하지 않습니다. Registered 에디션 전용.
ConsoleRender	Console	UI 프레임워크 없이 각 유형의 모델 하나를 PNG와 SVG 파일로 기록합니다. Delphi Headless 데모의 짝입니다.

빌드와 실행

```
dotnet run --project Gallery\Gallery.csproj
```

sgcUI 소스 트리 안에서는 데모가 Registered 에디션의 라이브러리 프로젝트에 대해 컴파일됩니다. Community 에디션에는 `-p:SgcCommunity=true`를 추가합니다. 데모 zip에서는 데모가 nuget.org에서 패키지를 복원하며, 기본값은 Community 패키지입니다. Registered 패키지를 쓰려면 `-p:SgcGaugesEdition=Registered`를 설정합니다.
Community 에디션에서 갤러리는 게이지 6개를 보여 주고, HMIPanel은 어넌시에이터 대신 목록 상자에 경보를 나열하며, ConsoleRender는 Community 유형 6개를 기록합니다. MQTTLive에는 Registered 에디션과 esegece.sgcWebSockets 패키지가 필요합니다.

자체 테스트

모든 데모는 `--test result.txt`를 받습니다. 화면 없이 자체 점검을 실행하고, 점검마다 PASS 또는 FAIL 한 줄을 기록하며, 실패한 점검의 수를 종료 코드로 하여 끝납니다. `--snapshot file.png`는 PNG 이미지를 저장하고 끝납니다.

참고 항목

- .NET 개요
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 헤드리스 출력
- .NET sgcWebSockets 브리지

문제 해결

값이 100에서 멈춥니다

값은 쓸 때 해당 스케일의 범위로 제한되며 기본 스케일은 0부터 100까지입니다. 값보다 먼저 스케일의 Minimum과 Maximum을 설정하십시오. 나중에 범위를 바꾸면 저장된 값이 다시 제한되지만, 잘려 나간 값이 복원되지는 않습니다.

두 번째 스케일이나 포인터가 그려지지 않습니다

Community 에디션은 첫 번째 스케일과 첫 번째 포인터만 그립니다. 나머지는 컬렉션에 보관됩니다. 에디션을 참조하십시오.

TsgcVCLClockGauge 또는 TsgcCompassGauge가 선언되어 있지 않습니다

VU 미터, 시계, 나침반, 애넌시에이터는 Registered 에디션에 포함됩니다. Community에서는 그 유닛이 비어 있고 컨트롤이 등록되지 않습니다.

14 세그먼트 디스플레이가 7 세그먼트로 표시됩니다

14 세그먼트와 도트 매트릭스 디스플레이는 Registered 에디션에 포함됩니다. Community에서는 7 세그먼트로 그려집니다.

바늘이 움직이지 않고 튑니다

Animation.Enabled와 Animation.Duration을 확인하십시오. 디자인 시점, 폼 파일에서 불러온 값, `SetValue(aPointer, aValue, False)`, 그리고 목표 값을 그리는 헤드리스 렌더링에서는 애니메이션이 없습니다.

OnThreshold가 발생하지 않습니다

- gtdRising과 gtdFalling에서는 임계값이 활성이 될 때만 이벤트가 발생합니다. 해제 이벤트도 받으려면 gtdBoth를 사용하십시오.
- 이벤트 후 임계값이 다시 발생하려면 값이 Hysteresis로 정해지는 띠를 벗어나야 합니다.
- 임계값 자체를 바꾸면 이벤트 없이 Active가 갱신됩니다.
- Enabled와 PointerIndex를 확인하십시오. Community에서는 첫 번째 이후 포인터의 임계값이 평가되지 않습니다.

스케일에 임계값 표시가 없습니다

임계값은 Color가 0이 아닐 때만 표시를 그립니다.

소수점이 쉼표인 로캘에서 레이블에 '.'이 표시됩니다

게이지는 자체 FormatSettings를 사용하며 기본값은 '.'이므로 출력이 컴퓨터에 따라 달라지지 않습니다. 이를 바꾸려면 컨트롤의 Gauge 속성을 통해 모델의 FormatSettings를 지정하십시오. 테마를 참조하십시오.

텍스트가 너무 작거나 너무 큽니다

글꼴 크기는 게이지 크기에 대한 상대값이므로 텍스트는 컨트롤의 크기를 따릅니다. 비율을 바꾸려면 TitleFont, ValueFont, LabelFont의 Size를 변경하십시오.

작업자 스레드에서 값 쓰기

Value, SetValue, 포인터의 Value와 EndValue는 스레드 안전하며, 컨트롤은 메인 스레드에서 다시 그리고 OnThreshold를 호출합니다. 그 밖의 모든 속성은 메인 스레드의 것입니다.

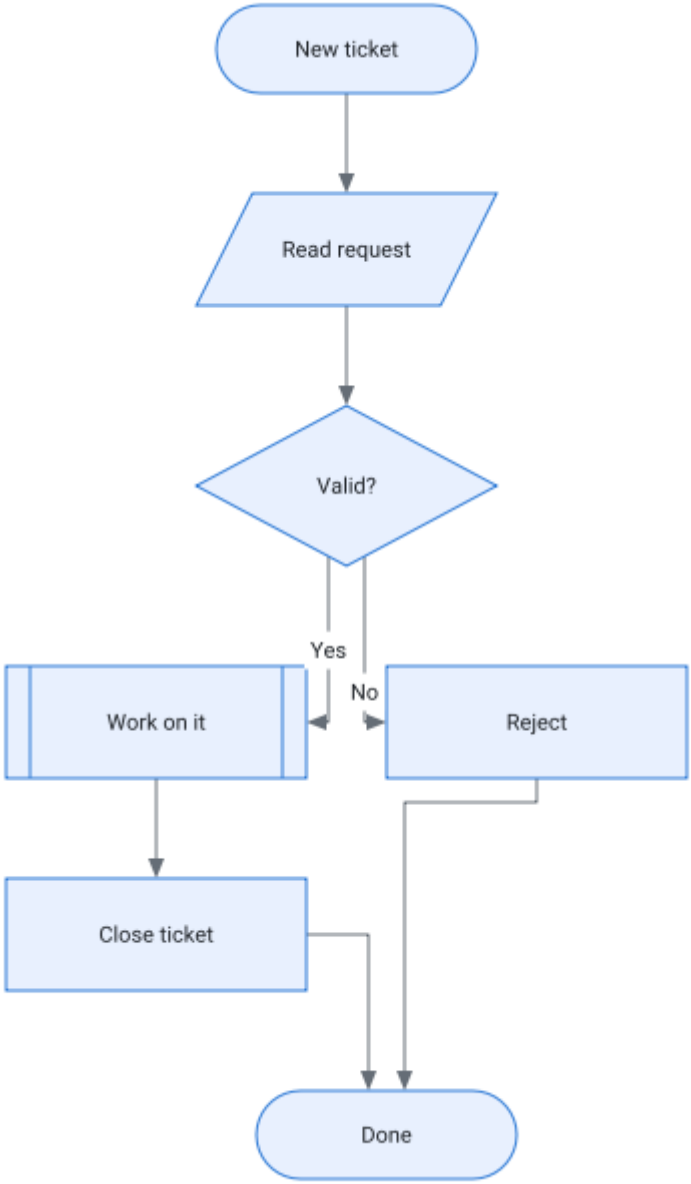
예제

참고 항목

- 에디션
- 애니메이션
- 임계값

소개

sgcDiagram은(는) Delphi 및 C++Builder 애플리케이션에 VCL과 FireMonkey용 다이어그램 편집기와 뷰어를 추가합니다. 순서도, 조직도, 노드 그래프를 다룰 수 있습니다. 사용자는 팔레트에서 도형을 끌어 놓고, 도형 사이에 커넥터를 그리고, 노드를 이동하고 크기를 조정하고 회전하며, 텍스트를 그 자리에서 편집하고, 모든 단계를 실행 취소할 수 있습니다. 같은 다이어그램을 창 없이 서버에서나 Linux64에서도 코드로 만들고, 정렬하고, 내보낼 수 있습니다.
모든 내용은 하나의 문서 객체인 **TsgcDiagramDocument**에 들어 있습니다. 노드, 포트와 레이블이 있는 커넥터, 레이어, 이미지, 기본 스타일 입니다. **TsgcVCLDiagramView** 와 **TsgcFMXDiagramView** 는 문서를 표시하고 편집하며, **TsgcVCLDiagramPalette**와 **TsgcFMXDiagramPalette**는 뷰로 끌어 놓을 도형을 제공하고, **TsgcDiagramRenderer**는 뷰 없이 문서를 PNG와 SVG로 기록합니다. 문서는 JSON으로 저장하고 불러오며, 뷰는 다이어그램을 폼 파일에 저장합니다.
sgcDiagram은 eSeGeCe 비주얼 스위트의 일부입니다. 캔버스, 래스터라이저, 글꼴, SVG 작성기, 라이트 및 다크 테마를 갖춘 공유 런타임 패키지인 **sgcUICore**로 그려지므로, 다이어그램은 VCL, FireMonkey, 내보낸 파일에서 모두 같은 모양입니다. sgcDiagram에는 sgcUICore가 필요하며, 함께 설치됩니다.



에디션

sgcDiagram은 Registered, Trial, Community 에디션으로 제공됩니다. Community 에디션은 완전한 수동 편집기입니다. 모든 도형, 사용자 지정 경로, 이미지, 세 가지 커넥터 종류, 포트와 접착, 그룹과 컨테이너, 실행 취소, 클립보드, JSON, PNG와

SVG, 테마, 스냅과 가이드, 팔레트, 읽기 전용 뷰어를 사용할 수 있습니다. 자동 레이아웃, 장애물을 피하는 직교 라우팅, 데이터 바인딩, 둘 이상의 레이어는 포함되지 않습니다. 에디션을 참조하십시오.

시작하는 곳

- sgcDiagram이란: 그릴 수 있는 다이어그램의 종류와 사용자가 할 수 있는 작업.
- 아키텍처: 문서, 뷰, 컨트롤러, 유닛.
- 빠른 시작 VCL, 빠른 시작 FMX, 빠른 시작 C++Builder, 빠른 시작 헤드리스: 몇 분 만에 만드는 첫 다이어그램.
- 컴포넌트: TsgcVCLDiagramView, TsgcFMXDiagramView, TsgcVCLDiagramPalette, TsgcFMXDiagramPalette, TsgcDiagramDataBinding 및 이들의 이벤트.
- 클래스: TsgcDiagramDocument, TsgcDiagramNode, TsgcDiagramConnector, TsgcDiagramPort, TsgcDiagramStyle, TsgcDiagramLayer, TsgcDiagramController, TsgcDiagramRenderer 및 JSON 형식.
- 계층형 레이아웃, 트리 레이아웃, 데이터 바인딩: 다이어그램을 자동으로 정렬하거나 데이터셋에서 만듭니다.
- 기능: 도형, 커넥터와 라우팅, 포트와 접착, 편집과 키보드, 스냅과 가이드, 실행 취소와 클립보드, 레이어, 직렬화, 내보내기, 테마, 뷰어 모드.
- 플랫폼: 지원되는 IDE, 프레임워크, 운영 체제.
- 문제 해결: 흔한 문제와 그 원인.

sgcDiagram이란

sgcDiagram은 다이어그램 컴포넌트 집합입니다. 노드와 커넥터의 문서, 이를 편집하는 대화형 뷰, 도형 팔레트, 내보내기로 구성됩니다. 업무용 애플리케이션이 표시하고 사용자가 편집할 수 있는 다이어그램을 위해 만들어졌습니다.

그릴 수 있는 것

- **순서도.** 프로세스, 판단, 터미네이터, 데이터, 문서, 미리 정의된 프로세스, 커넥터 원 도형, 그리고 화살표와 Yes 및 No 레이블이 있는 커넥터.
- **조직도와 트리.** 이름과 역할이 있는 상자를 직교 커넥터로 연결합니다. Registered 에디션에서는 트리 레이아웃으로 정렬하고 데이터셋에서 만들 수 있습니다.
- **노드 그래프.** 형식이 지정된 입력 및 출력 포트가 있는 노드, 곡선 커넥터, OnDrawNode에서의 사용자 지정 그리기. 노드 편집기나 파이프라인 디자이너 같은 것입니다.
- **자유 스케치.** 사각형, 둥근 사각형, 타원, 자유 텍스트, 이미지, SVG 경로 데이터로 만든 사용자 지정 도형, 다른 노드를 보유하는 컨테이너.

사용자가 할 수 있는 것

- 팔레트에서 도형을 뷰로 끌어 놓거나, 도형을 선택하여 도형 도구로 그립니다.
- 클릭, Shift 또는 Ctrl 클릭, 또는 선택 사각형으로 선택한 다음 이동, 크기 조정, 회전, 화살표 키로 미세 이동, 그룹화, 삭제, 복사, 붙여넣기를 합니다.
- 포트에서 커넥터를 그리고, 그 끝을 포트 또는 노드의 외곽선에 접착하고, 레이블과 Waypoints를 이동합니다.
- 노드의 텍스트를 그 자리에서 편집합니다. 더블 클릭, F2를 누르거나 그냥 입력을 시작합니다.
- 모든 동작을 실행 취소하고 다시 실행하며, 마우스 휠, 가운데 버튼 또는 Space로 이동하고, Ctrl과 휠 또는 핀치로 확대/축소합니다.

코드로 할 수 있는 것

- AddNode와 AddConnector로 다이어그램을 만들고, 임의의 속성을 변경하고, BeginTransaction과 EndTransaction으로 변경을 하나의 실행 취소 단계로 묶습니다.
- JSON으로 파일, 스트림 또는 문자열에 저장하고 불러옵니다.
- TsgcDiagramRenderer로 PNG와 SVG로 내보냅니다. 콘솔 애플리케이션이나 Linux64에서도 사용할 수 있습니다.
- OnChange, OnSelectionChange, OnItemClick, OnConnecting 등의 이벤트를 받아 사용자 동작에 반응합니다.

하지 않는 것

버전 1에는 BPMN, UML, 네트워크 스텐실, 수영 레인 풀, 힘 기반이나 원형 레이아웃, PDF 내보내기, 다른 애플리케이션과의 클립보드 교환, 미니맵, 눈금자, 스크롤 막대, 인쇄, SVG나 Visio 가져오기, 노드 안의 서식 있는 텍스트가 없습니다.

관련 항목

- 아키텍처
- 에디션
- 빠른 시작 VCL

아키텍처

sgcDiagram은 문서, 상호작용, 그리기를 분리하므로 같은 코드가 VCL, FireMonkey, 창 없는 테스트에서 다이어그램을 편집합니다.

구성 요소	역할
TsgcDiagramDocument	모델: Nodes, Connectors, Layers, Images, Selection, 기본 NodeStyle과 ConnectorStyle, Routing, GridSize, Theme, 실행 취소 스택. 모든 변경은 BeginUpdate와 EndUpdate 쌍마다 한 번 OnChange를 통해 알립니다.
TsgcDiagramNode	Shape, Left, Top, Width, Height, Angle, Text, Style, Ports, Layer, ContainerId, GroupId, ImageKey, PathData, Locked, Visible, Data, Tag를 가진 상자. z 순서는 Nodes 안의 인덱스입니다.
TsgcDiagramConnector	두 노드 또는 두 자유 점 사이의 링크: Kind, SourceNodeId, SourcePortId, TargetNodeId, TargetPortId, SourcePoint, TargetPoint, Waypoints, StartArrow, EndArrow, ArrowSize, Labels, Style. Points는 계산된 경로를 보유합니다.
Router	Routing.AutoRoute가 True이면 각 변경 집합이 끝날 때 끝, 노드 또는 종류가 변경된 모든 커넥터의 경로를 계산합니다.
TsgcDiagramController	상호작용 상태 기계: 히트 테스트, 선택, 끌기, 크기 조정, 회전, 연결, 스냅, 가이드, 텍스트 편집, 확대/축소, 이동. 프레임워크 중립이며 외부에서는 디바이스 픽셀, 내부에서는 월드 단위로 동작합니다.
TsgcDiagramPainter와 뷰 캐시	sgcUICore의 TsgcUICanvas에 문서를 그립니다. 뷰는 콘텐츠의 캐시된 이미지를 유지하므로 이동은 픽셀 복사로 끝나고, 편집은 변경된 사각형만 다시 그립니다.
TsgcVCLDiagramView, TsgcFMXDiagramView	컨트롤. 문서(Diagram 속성)와 Controller를 소유하며 마우스, 키보드, 제스처, 끌어서 놓기를 컨트롤러 호출로 변환합니다.
TsgcVCLDiagramPalette, TsgcFMXDiagramPalette	범주별로 묶인 도형 항목의 격자. 클릭하면 연결된 뷰의 도형 도구가 선택되고, 끌면 도형이 뷰에 놓입니다.
TsgcDiagramRenderer	뷰 없이 문서를 PNG와 SVG로 내보냅니다.

좌표

월드 단위는 확대/축소 1, 96 PPI에서의 픽셀이며 y는 아래쪽으로 증가합니다. 노드 각도는 노드 중심을 기준으로 시계 방향의 도 단위입니다. 색상은 TsgcUIColor이며 0은 테마의 색상을 뜻합니다. 뷰의 ScreenToDiagram과 DiagramToScreen은 컨트롤의 디바이스 픽셀과 월드 단위를 서로 변환합니다.

변경 알림

문서는 가장 바깥쪽 BeginUpdate와 EndUpdate 사이의 변경을 모아 한 번만 알립니다. 먼저 이를 소유한 뷰에 알려 변경된 사각형을 다시 그리게 하고, 그다음 TsgcDiagramChange 값의 집합(dgkGeometry, dgkStyle, dgkStructure, dgkSelection, dgkView)과 함께 OnChange를 통해, 그리고 OnSelectionChange로 알립니다. BeginUpdate와 EndUpdate 밖에서 코드로 한 변경은 즉시 알립니다.

유닛

유닛	내용

sgcDiagram_Types	열거형, TsgcDiagramPoint, TsgcDiagramRect, sgcDiagramPoint와 sgcDiagramBounds 같은 기하 도우미, EsgcDiagramError.
sgcDiagram_Shapes	모든 도형의 윤곽선, 텍스트 사각형, 기본 포트, 그리고 SVG 경로 데이터 파서.
sgcDiagram_Model	TsgcDiagramDocument, 노드, 커넥터, 포트, 레이블, 레이어, 이미지, 스타일, 선택 항목.
sgcDiagram_Undo, sgcDiagram_JSON, sgcDiagram_Router	실행 취소 스택, JSON 형식과 클립보드, 커넥터 라우팅.
sgcDiagram_Render	페인터, 뷰 캐시, TsgcDiagramRenderer.
sgcDiagram_Palette, sgcDiagram_Controller	팔레트 모델과 상호작용 컨트롤러.
sgcDiagram_Layout, sgcDiagram_Data	계층형 및 트리 레이아웃과 데이터 바인딩. Registered 에디션용이며 Community에서는 빈 유닛입니다.
sgcDiagram_VCL, sgcDiagram_FMX	VCL과 FireMonkey 뷰와 팔레트.

관련 항목

- sgcDiagram이란
- 플랫폼
- 실행 취소와 클립보드
- TsgcDiagramDocument
- TsgcDiagramController
- TsgcDiagramRenderer

에디션

sgcDiagram은 모든 에디션에서 같은 소스로 빌드됩니다.

에디션	포함 내용
Registered	모든 것: VCL과 FireMonkey용 뷰와 팔레트, 모든 도형, 세 가지 커넥터 종류, 포트, 그룹과 컨테이너, 실행 취소, 클립보드, JSON, PNG와 SVG, 테마, 스냅, 가이드, 그리고 아래에 나열된 고급 기능. SGC Diagram 팔레트 페이지에는 5개의 컴포넌트가 표시됩니다.
Trial	Registered와 같은 기능입니다. SGC_TRIAL로 컴파일된 빌드는 빌드에 지정된 릴리스 기간이 지난 후 컴포넌트가 만들어지면 평가판이라는 메시지를 표시합니다. 그 외에는 달라지는 것이 없습니다.
Community	고급 기능을 제외한 모든 것: VCL, FireMonkey, 헤드리스 용도의 완전한 수동 순서도 및 조직도 편집기. SGC Diagram 팔레트 페이지에는 4개의 컴포넌트(뷰 2개와 팔레트 2개)가 표시됩니다.

고급 기능

고급 기능은 SGC_DIAGRAM_ADVANCED 정의 아래에서 컴파일되며, sgcVer.inc는 Community 에디션에서 이를 끕니다. 패키지의 유닛 목록은 모든 에디션에서 같습니다. Community에서는 sgcDiagram_Layout과 sgcDiagram_Data가 빈 유닛으로 컴파일됩니다.

기능	Community 에디션에서
자동 레이아웃: TsgcDiagramLayeredLayout과 TsgcDiagramTreeLayout	사용할 수 없습니다. 노드는 사용자 또는 코드가 놓은 위치에 그대로 남습니다.
장애물을 피하는 직교 라우팅과 평행한 세그먼트의 미세 조정	dgcOrthogonal 커넥터는 노드를 피하지 않는 단순한 엘보로 라우팅되며 RouteFailed는 False로 유지됩니다.
데이터 바인딩: TsgcDiagramDataBinding은 데이터셋에서 다이어그램을 만듭니다	사용할 수 없습니다.
첫 번째 레이어를 넘는 레이어	레이어가 이미 있으면 Layers.Add는 EsgcDiagramError를 발생시킵니다. Registered 에디션에서 저장한 파일은 레이어 이름을 유지하며 그대로 불러오고 저장할 수 있지만, 모든 레이어가 표시 및 잠금 해제 상태로 그려지고 내보내지고 편집됩니다.

다이어그램 파일은 모든 에디션에서 같으므로 Registered 에디션에서 만든 문서를 Community 빌드에서 열 수 있고 그 반대도 가능합니다.

관련 항목

- 커넥터와 라우팅
- 레이어
- 플랫폼
- 계층형 레이아웃
- 트리 레이아웃
- TsgcDiagramDataBinding

플랫폼

sgcDiagram은 모든 IDE와 플랫폼에 대해 하나의 소스 트리입니다. 문서, 라우터, 컨트롤러, 페인터, 렌더러는 RTL과 sgcUICore만 사용하므로 모든 대상에서 같은 코드입니다. 프레임워크에 묶여 있는 것은 뷰와 팔레트뿐입니다.

대상	지원
Delphi VCL	Delphi 7부터 Delphi 13까지, Win32와 Win64: TsgcVCLDiagramView와 TsgcVCLDiagramPalette.
Delphi FireMonkey	Windows, macOS, iOS, Android용 Delphi XE6부터 Delphi 13까지: TsgcFMXDiagramView와 TsgcFMXDiagramPalette. SGC_DIAGRAM_FMX로 컴파일됩니다.
Linux64	Delphi 10.2 이상, SGC_DIAGRAM_LINUX로 컴파일됩니다. 문서, JSON, 라우팅, 레이아웃, TsgcDiagramRenderer를 사용할 수 있으므로 서버에서 다이어그램을 만들고 PNG와 SVG 파일을 생성할 수 있습니다. 빠른 시작 헤드리스를 참조하십시오.
C++Builder	생성된 .hpp 헤더를 사용하는 같은 컴포넌트. 빠른 시작 C++Builder를 참조하십시오.
.NET	동일한 API를 제공하는 C# 이식판으로, WinForms, WPF, Avalonia 각각에 뷰와 팔레트를 제공하고 헤드리스 JSON, PNG, SVG 출력도 지원합니다. .NET Framework 4.7.2와 .NET 8 및 9에서 동작합니다. .NET 개요를 참조하십시오.

프레임워크별 유닛

유닛	프레임워크
sgcDiagram_Const, sgcDiagram_Types, sgcDiagram_Shapes, sgcDiagram_Model, sgcDiagram_Undo, sgcDiagram_JSON, sgcDiagram_Router, sgcDiagram_Layout, sgcDiagram_Render, sgcDiagram_Palette, sgcDiagram_Controller	없음. 모델, 엔진, 페인터, 파일 형식.
sgcDiagram_Data	DB 유닛만 사용(dbrtl). VCL이나 FireMonkey 유닛은 사용하지 않습니다.
sgcDiagram_VCL	VCL, Windows 전용.
sgcDiagram_FMX	FireMonkey, Delphi XE6 이상.

패키지

sgcDiagram에는 프레임워크에 독립적인 런타임 패키지와 프레임워크마다 하나씩의 런타임 패키지가 들어 있으므로, Android, iOS, macOS용 FireMonkey 애플리케이션은 VCL을 링크하지 않습니다. 각 패키지 이름은 IDE 접미사로 끝납니다. 예를 들어 Delphi 13에서는 sgcDiagramD13.dpk, C++Builder 13에서는 sgcDiagramC13.cbproj입니다.

패키지	종류	내용	필요한 패키지
sgcDiagram	런타임, 프레임워크 독립	VCL이나 FireMonkey를 사용하지 않는 제품의 모든 유닛. 문서, 도형, 실행 취소, JSON, 라우터, 레이아웃, 렌더러, 팔레트 모델, 컨트롤러, Community 에디션이 아닌 경우 TsgcDiagramDataBinding(sgcDiagram_Data).	rtl, dbrtl, sgcUICore

sgcDiagramVCL	런타임, VCL	sgcDiagram_VCL (TsgcVCLDiagramView, TsgcVCLDiagramPalette)	rtl, vcl, sgcUICore, sgcUICoreVCL, sgcDiagram
sgcDiagramFMX	런타임, FireMonkey, XE6부터	sgcDiagram_FMX (TsgcFMXDiagramView, TsgcFMXDiagramPalette)	rtl, fmx, sgcUICore, sgcUICoreFMX, sgcDiagram
dclsgcDiagram	디자인 타임, VCL	**SGC Diagram** 팔레트 페이지에 등록하는 컴포넌트는 TsgcVCLDiagramView, TsgcVCLDiagramPalette이며, Community 에디션이 아닌 경우 TsgcDiagramDataBinding도 등록합니다. 또한 뷰의 컴포넌트 에디터에 "Load Diagram...", "Save Diagram...", "Clear Diagram" 명령을 추가합니다.	designide, rtl, vcl, sgcDiagram, sgcDiagramVCL
dclsgcDiagramFMX	디자인 타임, FireMonkey, XE6부터	**SGC Diagram** 팔레트 페이지에 등록하는 컴포넌트는 TsgcFMXDiagramView 및 TsgcFMXDiagramPalette이며, 같은 명령을 추가합니다.	designide, rtl, fmx, sgcDiagram, sgcDiagramFMX, dclsgcDiagram

sgcUICore도 같은 구성입니다. sgcUICore, sgcUICoreVCL, sgcUICoreFMX가 있고 디자인 타임 패키지는 없습니다. dclsgcDiagramFMX 에는 dclsgcDiagram 가 필요하므로 dclsgcDiagram 를 먼저 설치하십시오. sgcDiagram 에는 sgcWebSockets용 브리지 패키지가 없습니다.

IDE 버전과 대상 플랫폼

Delphi	sgcDiagram	sgcDiagramVCL	sgcDiagramFMX	dclsgcDiagram, dclsgcDiagramFMX
7, 2007, 2009, 2010, XE	Win32	Win32	없음	Win32
XE2부터 XE5까지	Win32, Win64	Win32, Win64	없음	Win32
XE6부터 10.2까지	Win32, Win64	Win32, Win64	Win32, Win64	Win32
10.3	Win32, Win64, Linux64, OSX64	Win32, Win64	Win32, Win64, OSX64	Win32
10.4	Win32, Win64, Linux64, OSX64, Android, Android64, iOSDevice64	Win32, Win64	Win32, Win64, OSX64, Android, Android64, iOSDevice64	Win32
11	Win32, Win64, Linux64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32, Win64	Win32, Win64, OSX64, OSXARM64, Android, Android64, iOSDevice64, iOSSimARM64	Win32
12, 13	11과 같음	Win32, Win64	11과 같음	Win32, Win64

XE6부터 10.2까지는 FireMonkey 패키지가 Windows용으로만 빌드됩니다. 디자인 타임 패키지는 IDE가 로드하므로 IDE 플랫폼용으로만 빌드됩니다.

C++Builder	sgcDiagram, sgcDiagramVCL	sgcDiagramFMX	dclsgcDiagram, dclsgcDiagramFMX
2007, 2010, XE, XE2	Win32	없음	Win32 (dclsgcDiagram만)
XE3부터 XE5까지	Win32, Win64	없음	Win32 (dclsgcDiagram만)
XE6부터 12까지	Win32, Win64	Win32, Win64	Win32
13	Win32, Win64, Win64x	Win32, Win64, Win64x	Win32, Win64x

C++Builder 프로젝트는 같은 이름에 C 접미사가 붙습니다. 예를 들어 sgcDiagramC13.cbproj, sgcDiagramVCLC13.cbproj, sgcDiagramFMXC13.cbproj, dclsgcDiagramC13.cbproj, dclsgcDiagramFMXC13.cbproj입니다. C++Builder 패키지는 Windows용으로만 빌드됩니다. 모든 sgcUI 제품에 적용되는 C++Builder 관련 참고 사항(Win64의 런타임 패키지, Win64의 C++Builder 10.3, Win64x 패키지 이름)은 sgcUI 제품군 도움말의 "플랫폼과 패키지" 항목에 있습니다.

Android, iOS, macOS에서의 FireMonkey

FireMonkey 컨트롤(TsgcFMXDiagramView 및 TsgcFMXDiagramPalette)은 macOS는 Delphi 10.3부터, Android와 iOS는 Delphi 10.4부터, macOS ARM과 iOS 시뮬레이터는 Delphi 11부터 지원합니다. 해당 플랫폼용 FireMonkey 프로젝트에서는 다음과 같이 합니다.

- FireMonkey 컨트롤을 사용합니다. VCL 컨트롤과 sgcDiagramVCL은 Windows 전용입니다.
- 컨트롤을 폼에 놓거나 해당 유닛(sgcDiagram_FMX)을 uses 절에 추가합니다.
- 프로젝트 창에서 대상 플랫폼을 선택합니다. 설치 프로그램은 빌드한 모든 플랫폼의 라이브러리 경로에 sgcUI Source 폴더와 각 플랫폼의 컴파일된 유닛(LibD<version>\<platform>)을 추가합니다. 설치 프로그램을 사용하지 않으면 대상 플랫폼의 라이브러리 경로에 sgcUI Source 폴더를 추가하십시오(Tools, Options, Language, Delphi, Library).
- Android와 iOS 애플리케이션은 정적으로 링크되므로 런타임 패키지를 배포하지 않습니다.

Linux64

프레임워크에 독립적인 패키지 sgcDiagram는 Delphi 10.3부터 Linux64용으로 빌드됩니다. Linux64 애플리케이션(예: 콘솔 프로그램이나 서버)은 문서, JSON, 라우팅, 레이아웃, TsgcDiagramRenderer를 사용하여 다이어그램을 만들고 PNG 및 SVG 파일을 생성합니다. Linux64용 VCL 또는 FireMonkey 패키지는 빌드되지 않습니다. Linux의 FireMonkey에는 서드파티 애드온인 FMX for Linux(FMXLinux)가 필요하기 때문입니다.

플랫폼별 참고 사항

- **고 DPI.** VCL 뷰는 모니터의 PPI에 맞게 배율이 조정되고, FireMonkey 뷰는 폼의 배율에 맞춥니다. 확대/축소 1에서는 장치 독립 픽셀당 1 월드 단위가 표시됩니다.
- **터치.** FireMonkey에서는 핀치가 중심을 기준으로 확대/축소하고 빈 공간의 손가락은 이동합니다. 노드 위의 손가락은 노드를 끌며, Windows 터치 화면에서도 마찬가지입니다.
- **팔레트에서 끌어서 놓기.** iOS와 Android에는 드래그 서비스가 없습니다. 팔레트 항목을 탭한 다음 뷰를 탭하여 도형을 배치합니다.
- **Tab 키.** VCL에서는 Tab과 Shift+Tab으로 선택이 항목을 순환합니다. FireMonkey 폼은 컨트롤보다 먼저 Tab을 처리하므로 그곳에서는 키가 포커스를 이동합니다.
- **Delphi 7과 Delphi 2007.** 이 버전들은 Ansi 문자열을 사용하므로 JSON 파일을 불러올 때 시스템 코드 페이지 밖의 텍스트는 손실됩니다.
- **에디션.** sgcVer.inc가 Community 에디션에서 끄는 SGC_DIAGRAM_ADVANCED가 모든 플랫폼에서 고급 기능을 결정합니다. 에디션을 참조하십시오.

관련 항목

- 아키텍처
- 내보내기
- 문제 해결

빠른 시작 VCL

사용 중인 Delphi 버전용 sgcDiagram 패키지를 설치합니다. 컴포넌트는 **SGC Diagram** 팔레트 페이지에 나타납니다.

디자인 타임

- 폼에 **TsgcVCLDiagramPalette**를 놓고 Align을 alLeft로 설정합니다.
- **TsgcVCLDiagramView**를 놓고 Align을 alClient로 설정합니다.
- 팔레트의 View 속성을 뷰로 설정합니다.
- 애플리케이션을 실행합니다. 팔레트에서 도형을 끌고, 포인터를 노드 위에 두면 나타나는 포트에서 커넥터를 그리고, 노드를 더블 클릭하여 텍스트를 편집합니다.

디자인 타임에 뷰의 오른쪽 클릭 메뉴에는 JSON 파일에서 다이어그램을 불러오고, 저장하고, 지우는 동사가 있습니다. 다이어그램은 dfm에 DiagramJSON 속성으로 저장됩니다.

코드에서

다음 단계

- FView.Diagram.SaveToFile로 다이어그램을 저장합니다. 직렬화를 참조하십시오.
- FView.SaveToPNG와 FView.SaveToSVG로 내보냅니다. 내보내기를 참조하십시오.
- ReadOnly로 뷰를 읽기 전용 뷰어로 전환합니다. 뷰어 모드를 참조하십시오.

관련 항목

- 빠른 시작 FMX
- 빠른 시작 헤드리스
- 편집과 키보드
- TsgcVCLDiagramView
- TsgcVCLDiagramPalette

빠른 시작 FMX

FireMonkey 컴포넌트는 **TsgcFMXDiagramView**와 **TsgcFMXDiagramPalette**이며, Delphi XE6 이상의 **SGC Diagram** 팔레트 페이지에 있습니다. VCL 컴포넌트와 같은 속성, 메서드, 이벤트와 같은 문서 클래스를 가지므로 Diagram 속성에 대한 코드는 공유할 수 있습니다.

디자인 타임

- Align을 Left로 한 TsgcFMXDiagramPalette와 Align을 Client로 한 TsgcFMXDiagramView를 놓습니다.
- 팔레트의 View 속성을 뷰로 설정합니다.
- Windows 또는 macOS에서 애플리케이션을 실행하고 도형을 뷰로 끕니다. iOS와 Android에서는 팔레트 항목을 탭한 다음 뷰를 탭합니다.

코드에서

터치

- 핀치는 중심을 기준으로 확대/축소합니다.
- 빈 공간의 손가락은 이동하고, 노드 위의 손가락은 노드를 끕니다.

관련 항목

- 빠른 시작 VCL
- 플랫폼
- TsgcFMXDiagramView
- TsgcFMXDiagramPalette

빠른 시작 C++Builder

사용 중인 C++Builder 버전용 sgcDiagram 패키지를 설치하고, `sgcUI\Source`와 컴파일된 유닛의 폴더(예: `sgcUI\LibD13\win32`, 64비트는 `sgcUI\LibD13\win64`)를 프로젝트의 포함 경로와 라이브러리 경로에 추가한 다음, **SGC Diagram** 팔레트 페이지의 컴포넌트를 Delphi와 똑같이 사용합니다.

C++에서 알아둘 점

- 열거형에는 범위가 없습니다. `dgsDecision`, `dgcCurved`, `dgtSelect`처럼 작성합니다.
- 이 도움말의 샘플은 예를 들어 `AddConnector(Start, Check, "", "")`처럼 메서드의 모든 인수를 명시적으로 전달하므로, 생성된 헤더의 어느 버전으로도 컴파일됩니다.
- 이벤트는 클로저이므로 그 핸들러는 클래스의 __fastcall 멤버 함수입니다. aAccept 같은 var 매개변수는 bool 참조입니다.
- TsgcDiagramRenderer는 일반 클래스입니다. `new TsgcDiagramRenderer()`로 만들고 delete로 삭제합니다.

코드에서

관련 항목

- 빠른 시작 VCL
- 플랫폼

빠른 시작 헤드리스

다이어그램에는 뷰가 필요하지 않습니다. TsgcDiagramDocument와 TsgcDiagramRenderer는 RTL과 sgcUICore만 사용하므로 콘솔 애플리케이션, 서비스, Linux64 서버가 다이어그램을 만들고, 라우팅하고, PNG, SVG, JSON 파일을 기록할 수 있습니다.

- `TsgcDiagramDocument.Create`(소유자 없음)로 문서를 만들고 직접 해제합니다.
- Routing.AutoRoute가 True(기본값)인 동안 커넥터는 각 변경 집합이 끝날 때 라우팅됩니다. 모든 커넥터를 다시 라우팅하려면 RouteAll을 호출합니다.
- 렌더러는 콘텐츠 경계에 여백(기본값은 20 월드 단위)을 더한 범위를 지정한 배율로 내보냅니다.

예제

관련 항목

- 내보내기
- 직렬화
- 플랫폼
- TsgcDiagramDocument
- TsgcDiagramRenderer
- 계층형 레이아웃

TsgcVCLDiagramView

TsgcVCLDiagramView(유닛 sgcDiagram_VCL)는 다이어그램을 표시하고 편집하는 VCL 컨트롤입니다. Diagram으로 노출되는 TsgcDiagramDocument와, 마우스, 키보드, 끌어서 놓기를 처리하는 TsgcDiagramController를 소유합니다.

속성

속성	의미
Diagram	뷰가 표시하고 편집하는 TsgcDiagramDocument. 폼 파일에 DiagramJSON 속성으로 스트리밍되며, 다이어그램이 비어 있지 않을 때만 기록됩니다.
Tool	포인터 도구: dgtSelect(기본값), dgtPan, dgtConnector, dgtShape.
ShapeTool	dgtShape 도구가 그리는 도형. 기본값은 dgsRectangle입니다.
ReadOnly	True로 하면 뷰가 뷰어가 됩니다. 선택, 호버, 확대/축소, 이동은 작동하고 편집은 작동하지 않습니다.
Zoom, MinZoom, MaxZoom	확대/축소 배율과 그 한계(기본값은 0.05와 8).
ShowGrid	Diagram.GridSize의 격자를 그립니다(기본값은 True).
SnapToGrid, SnapToGuides	이동하거나 크기를 조정한 노드를 격자와 주변 노드의 가장자리, 중심, 같은 간격에 맞춥니다(둘 다 기본값은 True).
ShowPorts	노드의 포트를 그리는 시점: dgvNever, dgvHover(기본값), dgvAlways.
Theme	다이어그램의 테마. Diagram.Theme과 같은 값입니다(별도로 저장되지 않음).
FontFile	텍스트를 그리는 데 사용하는 글꼴 파일. 뷰와 내보내기에서 같은 글꼴이 사용됩니다.
MaxFPS	컨트롤의 최대 다시 그리기 속도.
Controller	Public. 포인터와 키보드를 편집으로 바꾸는 TsgcDiagramController.

메서드

메서드	효과
ZoomToFit, ZoomToSelection	다이어그램 전체 또는 선택 항목이 뷰를 가득 채우도록 확대/축소합니다.
CenterOn(aPoint)	월드 상의 점을 뷰의 중심으로 스크롤합니다.
ScreenToDiagram, DiagramToScreen	뷰의 디바이스 픽셀과 월드 단위를 서로 변환합니다.
Undo, Redo	문서의 마지막 단계를 실행 취소하거나 다시 실행합니다.
CopyToClipboard, CutToClipboard, PasteFromClipboard	다이어그램의 클립보드. 실행 취소와 클립보드를 참조하십시오.
DeleteSelection	선택한 항목을 하나의 실행 취소 단계로 삭제합니다.

EditText(aNode)	노드의 텍스트에 대해 인플레이스 편집기를 엽니다. 편집이 허용되지 않으면 False를 반환합니다.
SaveToPNG, SaveToSVG	파일 또는 스트림으로 내보냅니다. 배율은 선택 사항입니다(기본값은 1).

이벤트

뷰는 이벤트에서 설명하는 OnChange, OnSelectionChange, OnItemClick, OnItemDblClick, OnHover, OnConnecting, OnConnected, OnDrawNode, OnTextEditing, OnTextEdited, OnZoomChange를 노출합니다.

참고 사항

- 뷰에는 스크롤 막대가 없습니다. 휠로 스크롤하고, Shift와 휠로 가로 스크롤하며, Ctrl과 휠로 포인터를 기준으로 확대/축소하고, 가운데 버튼, Space 또는 dgtPan 도구로 이동합니다. 편집과 키보드를 참조하십시오.
- 텍스트 편집기는 인플레이스 메모입니다. Enter를 누르거나 메모를 벗어나면 확정되고, Esc는 취소하며, Shift+Enter는 줄을 추가합니다.
- TsgcVCLDiagramPalette 항목을 놓으면 놓은 위치에 노드(또는 프래그먼트)가 만들어집니다.

예제

관련 항목

- TsgcFMXDiagramView
- TsgcVCLDiagramPalette
- TsgcDiagramDocument
- 빠른 시작 VCL

TsgcFMXDiagramView

TsgcFMXDiagramView(유닛 sgcDiagram_FMX)는 Windows, macOS, Linux, iOS, Android에서 다이어그램을 표시하고 편집하는 FireMonkey 컨트롤입니다. TsgcVCLDiagramView와 같은 속성, 메서드, 이벤트를 가집니다. Diagram, Tool, ShapeTool, ReadOnly, Zoom, MinZoom, MaxZoom, ShowGrid, SnapToGrid, SnapToGuides, ShowPorts, Theme, FontFile, MaxFPS, Controller, 확대/축소, 클립보드, 실행 취소, 내보내기 메서드, 그리고 이벤트에서 설명하는 OnChange, OnSelectionChange, OnItemClick, OnItemDblClick, OnHover, OnConnecting, OnConnected, OnDrawNode, OnTextEditing, OnTextEdited, OnZoomChange입니다.

VCL 뷰와의 차이점

- 다이어그램은 fmx 파일에 DiagramJSON 속성으로 스트리밍되며, 다이어그램이 비어 있지 않을 때만 기록됩니다.
- 위치는 디바이스 픽셀 단위입니다. 컨트롤의 로컬 좌표에 그 배율을 곱한 값입니다.
- 핀치는 핀치의 중심을 기준으로 확대/축소합니다. 항목 위에서 시작한 손가락은 그것을 끌고, 빈 공간에서 시작한 손가락은 이동합니다.
- FMX 마우스 휠에는 위치가 없으므로 휠은 마지막 포인터 위치를 기준으로 확대/축소합니다.
- iOS와 Android에는 끌어서 놓기 서비스가 없습니다. 팔레트 항목을 탭하여 준비한 다음 뷰를 탭하여 도형을 배치합니다.
- FMX 폼은 포커스된 컨트롤보다 먼저 Tab을 처리하므로 Tab과 Shift+Tab으로 항목을 순환할 수 없습니다.
- 텍스트 편집기는 폼의 스타일을 따르는 일반 FMX TMemo입니다.

예제

관련 항목

- TsgcVCLDiagramView
- TsgcFMXDiagramPalette
- 빠른 시작 FMX
- 뷰어 모드

TsgcVCLDiagramPalette

TsgcVCLDiagramPalette(유닛 sgcDiagram_VCL)는 사용자가 TsgcVCLDiagramView에 클릭 또는 끌어서 놓기로 추가할 수 있는 도형을 표시합니다.

속성

속성	의미
View	클릭하거나 끈 항목을 받는 다이어그램 뷰.
Items	항목. TsgcDiagramPaletteItem의 컬렉션이며 Category별로 묶여 격자에 표시됩니다.
IconSize	항목 아이콘의 크기(픽셀). 기본값은 32입니다.
ShowCaptions	각 아이콘 아래에 캡션을 그립니다(기본값은 True).
Theme	utLight(기본값), utDark 또는 utCustom.
MaxFPS	컨트롤의 최대 다시 그리기 속도.
Model	Public. 컨트롤의 배후에 있는 팔레트 모델. 레이아웃, 히트 테스트, 드래그 페이로드를 처리합니다.
DragPayload	Public. 진행 중인 끌기의 페이로드. 끌고 있는 항목의 JSON입니다.

항목

TsgcDiagramPaletteItem	의미
Category, Caption	그룹 머리글과 아이콘 아래의 텍스트.
Shape, Width, Height, Text, Style, PathData	항목이 만드는 노드(dgsPath용 PathData는 0..100 상자 안).
Fragment	Diagram.CopySelection이 반환하는 것과 같은 JSON 프래그먼트. 프래그먼트가 있는 항목은 프래그먼트 전체를 놓습니다.
Payload	읽기 전용. 항목이 놓는 JSON.

이벤트

이벤트	발생 시점
OnItemClick(Sender; aItem)	항목을 클릭했습니다.

참고 사항

- 항목을 클릭하면 뷰의 Tool이 dgtShape으로, ShapeTool이 해당 항목의 도형으로 설정됩니다. 이후 뷰를 클릭하거나 끌면 노드가 만들어지고, Esc를 누르면 dgtSelect로 돌아갑니다. 프래그먼트 항목은 끌어서 놓기로만 배치됩니다.
- 항목을 뷰로 끌면 미리 보기가 표시되고, 포인터 위치에 노드(또는 프래그먼트)가 하나의 실행 취소 단계로 놓입니다. 페이로드는 시스템 클립보드를 거치지 않습니다.

- Items.AddShape(aCategory, aCaption, aShape, aWidth, aHeight, aText)와 Items.AddFragment(aCategory, aCaption, aFragment)는 코드로 항목을 추가합니다. LoadDefaultItems는 기본 항목을 복원합니다.

예제

관련 항목

- TsgcFMXDiagramPalette
- TsgcVCLDiagramView
- 도형

TsgcFMXDiagramPalette

TsgcFMXDiagramPalette(유닛 sgcDiagram_FMX)는 TsgcFMXDiagramView 용 FireMonkey 팔레트입니다. TsgcVCLDiagramPalette 와 같은 속성, 항목, 이벤트를 가집니다. View, Items, IconSize, ShowCaptions, Theme, MaxFPS, Model, DragPayload, OnItemClick, LoadDefaultItems입니다.

VCL 팔레트와의 차이점

- 데스크톱 플랫폼에서는 끌면 폼의 FMX 끌어서 놓기가 시작되며, 포인터가 뷰 위에 있는 동안 뷰가 미리 보기를 표시합니다.
- iOS와 Android에는 드래그 서비스가 없습니다. 항목을 탭하면 뷰의 도형 도구가 준비되고, 뷰를 탭하면 도형이 배치됩니다.

참고 사항

- 항목을 클릭하면 뷰의 Tool이 dgtShape으로, ShapeTool이 해당 항목의 도형으로 설정됩니다. 이후 뷰를 클릭하거나 끌면 노드가 만들어지고, Esc를 누르면 dgtSelect로 돌아갑니다. 프래그먼트 항목은 끌어서 놓기로만 배치됩니다.
- 항목을 뷰로 끌면 미리 보기가 표시되고, 포인터 위치에 노드(또는 프래그먼트)가 하나의 실행 취소 단계로 놓입니다. 페이로드는 시스템 클립보드를 거치지 않습니다.
- Items.AddShape(aCategory, aCaption, aShape, aWidth, aHeight, aText)와 Items.AddFragment(aCategory, aCaption, aFragment)는 코드로 항목을 추가합니다. LoadDefaultItems는 기본 항목을 복원합니다.

예제

관련 항목

- TsgcFMXDiagramView
- TsgcVCLDiagramPalette

TsgcDiagramDataBinding

TsgcDiagramDataBinding(유닛 sgcDiagram_Data)는 행에서 다이어그램을 만듭니다. 대표적으로 조직도나 순서도입니다. 각 행이 노드가 되고, 부모 키마다 부모에서 자식으로 가는 커넥터가 만들어집니다. 프레임워크 중립이며 VCL 뷰, FMX 뷰 또는 문서 단독으로 동작합니다. 필드 이름과 열 이름은 대소문자를 구분하지 않습니다.

속성

속성	의미
DataSource	DataSet이 행을 제공하는 TDataSource.
Table	Public. 행을 제공하는 IsgcUITable. 할당하면 DataSource 대신 이것을 읽습니다. 테이블 객체를 해제하기 전에 nil로 설정하십시오.
KeyField	행의 고유 키를 가진 필드. 키가 노드 Id가 되므로 Document.FindNode(key)로 노드를 찾을 수 있습니다.
ParentField	부모의 키를 가진 필드 또는 ParentSeparator로 구분한 여러 키.
ParentSeparator	여러 부모 키의 구분 문자. 기본값은 세미콜론입니다. 부모가 여러 개이면 트리가 아니라 순서도가 됩니다.
LabelField	노드 텍스트로 표시되는 필드. 비어 있으면 키.
Target	다이어그램을 표시하는 뷰. IsgcDiagramDocumentOwner를 구현하는 모든 컴포넌트이며, 예를 들면 TsgcVCLDiagramView와 TsgcFMXDiagramView입니다.
Document	Public. 직접 채울 문서. 할당하면 Target의 문서 대신 사용됩니다. 소유하지 않으므로 문서를 해제하기 전에 nil로 설정하십시오.
Layout	dgbAuto(기본값)는 부모가 둘인 노드가 없으면 트리 레이아웃을, 있으면 계층형 레이아웃을 적용합니다. dgbTree와 dgbLayered는 둘 중 하나를 강제하고, dgbNone은 노드를 행 순서대로 행으로 유지합니다.
Direction, TreeStyle	레이아웃에 전달됩니다. 기본값은 dglTopToBottom과 dgoStandard입니다.
NodeShape, NodeWidth, NodeHeight	노드의 도형과 크기. 기본값은 dgsRoundRect, 140 x 60입니다.
ConnectorKind	커넥터의 종류. 기본값은 dgcOrthogonal입니다.
LiveUpdates	True(기본값)이면 데이터가 변경된 후 다이어그램을 다시 만듭니다.
Active	True로 하면 다이어그램을 만들고 데이터 추적을 시작합니다.

메서드와 이벤트

멤버	의미
Refresh	다이어그램을 지금 다시 만듭니다. 문서의 트랜잭션 안에서는 EsgcDiagramError를 발생시킵니다.

OnCreateNode(Sender; aNode; aRow)	텍스트, 도형, 크기가 설정된 후 각 노드에 대해 실행됩니다. aRow는 0부터 시작하는 행이며, DataSet이 그 행에 위치해 있습니다.
OnCreateConnector(Sender; aConnector; aRow)	모든 노드가 만들어진 후 각 커넥터에 대해 실행됩니다. aRow는 자식 노드의 행이며, DataSet은 그 행에 위치해 있지 않습니다.

규칙

- 키가 비어 있거나 앞선 행이 사용한 키인 행은 노드가 되지 않습니다. 부모 키가 비어 있거나, 알 수 없거나, 반복되거나, 행 자신의 키이면 커넥터가 되지 않으므로, 부모가 모두 알 수 없는 행은 루트가 됩니다.
- 다시 만들기는 문서(항목, 레이어, 이미지)를 지우고, 항목을 만들고, 레이아웃을 적용하고, 커넥터를 라우팅하는 작업을 한 번의 BeginUpdate 안에서 수행하므로 뷰는 한 번만 다시 그립니다. 이것은 실행 취소 단계가 아니며 두 실행 취소 스택이 모두 지워집니다.
- 바인딩은 단방향입니다. 데이터에 쓰지 않으며, 다이어그램에 가한 변경은 다음 다시 만들기에서 대체됩니다.
- 디자인 타임에는 다시 만들기가 실행되지 않으므로 뷰가 생성된 다이어그램을 폼 파일에 저장하는 일은 없습니다.
- 실시간 업데이트와 전체 예제는 데이터 바인딩을 참조하십시오.

Community 에디션

TsgcDiagramDataBinding 에는 SGC_DIAGRAM_ADVANCED 가 필요합니다. Community 에디션에서는 유닛 sgcDiagram_Data가 비어 있으며 컴포넌트가 등록되지 않습니다.

예제

관련 항목

- 데이터 바인딩
- 트리 레이아웃
- 계층형 레이아웃
- 에디션

이벤트

뷰의 이벤트는 해당 컨트롤러에서 옵니다. 문서와 바인딩에는 자체 이벤트가 있습니다. VCL 뷰와 FMX 뷰는 같은 매개변수로 같은 이벤트를 노출합니다.

뷰 이벤트

이벤트	발생 시점
OnChange(Sender; aChanges)	문서가 변경되었습니다. aChanges는 dgkGeometry, dgkStyle, dgkStructure, dgkSelection, dgkView의 집합입니다.
OnSelectionChange(Sender)	선택이 변경되었습니다.
OnItemClick(Sender; aItem), OnItemDblClick(Sender; aItem)	노드 또는 커넥터를 클릭하거나 더블 클릭했습니다.
OnHover(Sender; aItem)	포인터가 항목에 들어가거나 항목에서 나갑니다(aItem은 nil).
OnConnecting(Sender; aSource, aSourcePort, aTarget, aTargetPort; var aAccept)	커넥터를 끌 때 끝을 접착하기 전에 확인을 요청합니다. 빈 포트는 노드의 외곽선, nil 노드는 자유 끝을 뜻합니다. 거부하려면 aAccept를 False로 설정합니다.
OnConnected(Sender; aItem)	사용자가 커넥터를 만들거나 다시 연결했습니다.
OnDrawNode(Sender; aCanvas; aNode; aRect; aZoom; aScale; var aDefault)	노드가 그려지기 전. aRect는 회전하지 않은 상자이며 월드 단위입니다. 캔버스 변환은 월드 단위를 디바이스 픽셀에 대응시키며 노드 회전이 적용됩니다. 1 월드 단위는 aZoom x aScale 디바이스 픽셀입니다. aCanvas(TsgcUICanvas)에 그리고, 도형, 이미지, 텍스트의 기본 그리기를 건너뛰려면 aDefault를 False로 설정합니다. 포트 캡션은 어느 경우에나 그려집니다.
OnTextEditing(Sender; aNode; var aAllow)	텍스트 편집기가 열리려고 합니다. 텍스트를 읽기 전용으로 유지하려면 aAllow를 False로 설정합니다.
OnTextEdited(Sender; aNode)	편집기에서 노드의 텍스트를 변경했습니다.
OnZoomChange(Sender)	확대/축소가 변경되었습니다.

기타 이벤트

이벤트	발생 시점
TsgcDiagramDocument.OnChange, OnSelectionChange	뷰가 없는 문서를 위한 같은 알림.
TsgcDiagramDocument.OnDecodeImage(Sender; aKey; aStream; var aImage; var aHandled)	PNG가 아닌 형식의 이미지를 불러왔습니다. aStream을 aImage로 디코딩하고 aHandled를 설정합니다.
TsgcDiagramDataBinding.OnCreateNode, OnCreateConnector	TsgcDiagramDataBinding을 참조하십시오.
TsgcVCLDiagramPalette.OnItemClick, TsgcFMXDiagramPalette.OnItemClick	팔레트 항목을 클릭했습니다.

예제

관련 항목

- TsgcVCLDiagramView
- TsgcDiagramController
- 포트와 접착

TsgcDiagramDocument

TsgcDiagramDocument(유닛 sgcDiagram_Model)는 다이어그램을 보유합니다. 노드, 커넥터, 레이어, 이미지, 선택 항목, 실행 취소 기록입니다. 각 뷰는 하나의 문서를 표시하며 이는 Diagram 속성으로 노출됩니다. 문서는 뷰 없이 단독으로도 콘솔 애플리케이션이나 서비스에서 동작합니다.

속성

속성	의미
Nodes, Connectors	항목(z 순서, 마지막 항목이 맨 위).
Layers, Images	레이어와 이미지 노드가 표시하는 이미지(키별, Images.Add, AddPNG, Find, Delete).
Selection	선택한 항목: Add, Remove, Toggle, Clear, SelectAll, Contains, Count, Items.
Theme	utLight(기본값), utDark 또는 utCustom.
GridSize	월드 단위의 격자 간격. 기본값은 10입니다.
NodeStyle, ConnectorStyle	항목이 값을 0으로 두는 곳에서 사용되는 스타일.
DefaultConnectorKind	새 커넥터의 종류. 기본값은 dgcOrthogonal입니다.
Routing	라우터 설정: Margin(12), BendPenalty(40), Spacing(6), AutoRoute(True).
UndoLimit	보관하는 실행 취소 단계 수. 기본값은 100입니다.
InTransaction, Loading	읽기 전용 상태 플래그.

메서드

메서드	효과
AddNode(aShape, aX, aY, aWidth, aHeight, aText)	노드를 추가하고 반환합니다.
AddConnector(aSource, aTarget, aSourcePort, aTargetPort)	두 노드 사이에 커넥터를 추가합니다. 포트가 비어 있으면 끝이 노드의 외곽선에 접착됩니다.
FindNode(aId), FindConnector(aId)	Id로 항목을 찾습니다. 찾지 못하면 nil입니다.
ConnectorsOf(aNode, aList), ChildrenOf(aContainer, aList)	노드의 커넥터 또는 컨테이너의 자식을 TList에 채웁니다.
ItemsInRect(aRect, aList, aFullyInside)	사각형과 만나거나 사각형 안에 있는 항목을 TList에 채웁니다.
ContentBounds	보이는 모든 항목의 경계.
DeleteSelection, Clear	선택한 항목 또는 전체를 삭제합니다.
Group, Ungroup, BringToFront, SendToBack	선택 항목에 대해 작동합니다. Group은 새 그룹 Id를 반환합니다.

BeginUpdate, EndUpdate	많은 변경에 대해 다시 그리기와 변경 알림을 한 번만 수행합니다.
BeginTransaction(aCaption), EndTransaction, Rollback	많은 변경을 하나의 실행 취소 단계로 처리하거나, 아무것도 적용하지 않습니다.
BeginUndoSuspend, EndUndoSuspend	그 사이의 변경은 기록되지 않으며 두 실행 취소 스택이 모두 지워집니다. 대량 불러오기용입니다.
Undo, Redo, CanUndo, CanRedo, UndoCaption, RedoCaption	실행 취소 기록.
RouteAll	모든 커넥터를 다시 라우팅합니다.
SaveToJSON, LoadFromJSON, SaveToFile, LoadFromFile, SaveToStream, LoadFromStream	JSON 형식.
CopySelection, Paste(aJSON)	JSON으로 만든 문서의 일부분과 그 붙여넣기.
ResolveNodeStyle(aNode), ResolveConnectorStyle(aConnector)	항목을 그리는 데 사용되는 스타일. 테마 색상은 해석된 상태입니다.
LayerVisible(aName), LayerLocked(aName)	레이어를 처리하는 방식.

이벤트

OnChange(Sender; aChanges), OnSelectionChange, OnDecodeImage는 이벤트를 참조하십시오.

스레딩

문서는 그것을 소유한 스레드에서만 변경합니다. 뷰에 표시되는 경우에는 메인 스레드입니다.

예제

관련 항목

- TsgcDiagramNode
- TsgcDiagramConnector
- 실행 취소와 클립보드
- 아키텍처

TsgcDiagramNode

노드는 텍스트가 있는 도형입니다. 처리 단계, 판단, 조직도의 인물 또는 다른 노드의 컨테이너 등입니다. 노드는 Document.AddNode로 만들며 Document.Nodes에 들어 있습니다.

속성

속성	의미
Shape	도형. 기본값은 dgsRectangle입니다.
Left, Top, Width, Height	월드 단위의 상자(기본값은 120 x 60).
Angle	중심을 기준으로 시계 방향으로 회전한 각도(도).
Text, TextAlign, TextVAlign	텍스트와 그 정렬(기본값은 가운데).
Ports	포트. 비어 있으면 도형의 기본 포트를 사용합니다.
ContainerId	이 노드를 보유하는 컨테이너 노드의 Id.
GroupId	노드의 그룹. 그룹의 노드는 함께 선택되고 이동됩니다.
ImageKey	dgsImage 노드가 그리는 이미지의 Document.Images 내 키.
PathData	dgsPath 노드의 윤곽선. 0..100 상자 안의 SVG 경로 구문으로 지정합니다.
Visible	False로 하면 노드가 숨겨집니다.
Id	문서에서 항목의 고유한 Id. AddNode와 AddConnector가 생성합니다.
Style	항목의 스타일. 0인 값은 문서 스타일에서 가져옵니다.
Layer	레이어의 이름. 기본 레이어에서는 비어 있습니다.
Locked	True로 하면 항목을 선택하거나 편집할 수 없습니다.
Data, Tag	애플리케이션 데이터: 문자열(파일에 저장됨)과 정수.
Selected, Document	public, 읽기 전용.

메서드

메서드	효과
SetBounds(aLeft, aTop, aWidth, aHeight), MoveBy(aDX, aDY)	한 번의 변경으로 노드를 이동하거나 크기를 조정합니다.
BoundsRect, Center	상자와 그 중심.
PortPoint(aPortId)	포트의 월드 위치.

예제

관련 항목

- TsgcDiagramPort
- TsgcDiagramConnector
- 도형

TsgcDiagramConnector

커넥터는 두 노드 또는 노드와 자유 점을 연결합니다. 경로는 Kind에 따라 라우터가 계산합니다. 커넥터는 Document.AddConnector로 만들며 Document.Connectors에 들어 있습니다.

속성

속성	의미
Kind	dgcStraight, dgcOrthogonal 또는 dgcCurved.
SourceNodeId, SourcePortId, TargetNodeId, TargetPortId	접착된 끝. 포트가 비어 있으면 끝이 노드의 외곽선에 접착됩니다.
SourcePoint, TargetPoint	자유 끝. 노드 Id가 비어 있을 때 사용됩니다.
Waypoints	직선 또는 곡선 커넥터가 지나가는 점.
StartArrow, EndArrow, ArrowSize	화살촉(기본값은 dgaNone과 dgaTriangle)과 그 크기(10).
Labels	레이블: Text, 경로를 따른 0에서 1까지의 Position(기본값 0.5), 경로로부터의 Offset.
Points	public, 읽기 전용. 현재 경로.
RouteFailed	public, 읽기 전용. 라우터가 노드를 돌아가는 경로를 찾지 못한 경우 True.
Id	문서에서 항목의 고유한 Id. AddNode와 AddConnector가 생성합니다.
Style	항목의 스타일. 0인 값은 문서 스타일에서 가져옵니다.
Layer	레이어의 이름. 기본 레이어에서는 비어 있습니다.
Locked	True로 하면 항목을 선택하거나 편집할 수 없습니다.
Data, Tag	애플리케이션 데이터: 문자열(파일에 저장됨)과 정수.
Selected, Document	public, 읽기 전용.

메서드

메서드	효과
SetRoute(aPoints, aFailed)	경로를 설정합니다. 라우터와 레이아웃이 사용합니다.

Community 에디션

SGC_DIAGRAM_ADVANCED가 없으면 dgcOrthogonal 커넥터는 엘보로 라우팅됩니다. 장애물 회피도 미세 조정도 없으며 RouteFailed는 False로 유지됩니다.

예제

관련 항목

- 커넥터와 라우팅
- TsgcDiagramPort
- TsgcDiagramNode

TsgcDiagramPort

포트는 커넥터의 끝이 접착되는 노드 위의 이름 있는 점입니다. Ports 컬렉션이 비어 있는 노드는 도형의 기본 포트를 사용합니다. 각 변의 중점이며 이름은 left, top, right, bottom입니다.

속성

속성	의미
Id	SourcePortId와 TargetPortId가 사용하는 이름.
X, Y	노드 상자에 대한 비율로 나타낸 위치. 0에서 1(기본값은 0.5).
Side	커넥터가 나오는 변: dgpAuto(기본값), dgpLeft, dgpTop, dgpRight, dgpBottom.
Flow	dgfBoth(기본값), dgfIn, dgfOut: 포트에 접착할 수 있는 끝.
Caption	포트 옆에 그려지는 텍스트.

Node.Ports.Add는 포트를 추가하고 Node.Ports.Find(aId)는 포트를 찾습니다. Node.PortPoint(aId)는 그 월드 위치를 반환합니다.

예제

관련 항목

- 포트와 접착
- TsgcDiagramNode

TsgcDiagramStyle

모든 노드와 커넥터에는 Style이 있고, 문서에는 NodeStyle과 ConnectorStyle이 있습니다. 0으로 남아 있는 색상이나 크기는 문서 스타일에서 가져오고, 거기서도 0인 색상은 테마에서 가져오므로, 색상을 설정하지 않은 다이어그램은 테마 변경을 따릅니다.

속성

속성	의미
Fill	채우기 색상. TsgcUIColor입니다(sgcUIARGB로 만듭니다).
Stroke	외곽선 또는 선의 색상.
StrokeWidth	월드 단위의 외곽선 또는 선의 너비. 기본값은 1입니다.
Dash	dgdSolid(기본값), dgdDash, dgdDot.
CornerRadius	사각형과 둥근 사각형의 모서리 반지름.
Text	텍스트 스타일: Size, Bold, Color.

Document.ResolveNodeStyle(aNode)와 ResolveConnectorStyle(aConnector)는 항목을 그리는 데 사용되는 값을 반환합니다.

예제

관련 항목

- 테마
- TsgcDiagramDocument

TsgcDiagramLayer

레이어는 항목을 이름으로 묶습니다. Document.Layers에 레이어가 들어 있으며, 각 노드와 커넥터는 Layer 속성에서 레이어를 이름으로 지정합니다. 이름이 비어 있으면 기본 레이어입니다.

속성

속성	의미
Name	항목의 Layer 속성이 사용하는 이름.
Visible	False로 하면 뷰와 내보내기에서 해당 레이어의 항목이 숨겨집니다.
Locked	True로 하면 해당 레이어의 항목을 선택하거나 편집할 수 없습니다.

Layers.Add는 레이어를 추가하고 Layers.Find(aName)은 레이어를 찾습니다. 레이어의 추가, 삭제, 이름 변경은 실행 취소 단계가 되지만 Visible과 Locked는 그렇지 않습니다.

Community 에디션

SGC_DIAGRAM_ADVANCED가 없으면 두 번째 Layers.Add는 EsgcDiagramError를 발생시킵니다. 레이어가 여러 개인 파일은 그대로 불러오고 저장할 수 있지만, 모든 레이어가 표시 및 잠금 해제 상태로 그려지고 편집됩니다.

예제

관련 항목

- 레이어
- 에디션

TsgcDiagramController

TsgcDiagramController(유닛 sgcDiagram_Controller)는 뷰의 프레임워크 중립 편집 엔진입니다. 포인터, 키보드, 휠, 끌어서 놓기 입력을 문서의 변경으로 바꾸고, 확대/축소와 스크롤 위치를 유지하며, TsgcDiagramPainter를 통해 그립니다. 각 뷰가 하나를 만들고 Controller 속성으로 노출합니다. 위치는 디바이스 픽셀 단위이며, 컨트롤러는 내부에서 월드 단위로 작업합니다.

속성

속성	의미
Document, Host, Painter	문서, 호스팅하는 컨트롤(IsgcDiagramHost), 페인터.
Tool, ShapeTool, ShapeWidth, ShapeHeight	활성 도구와, 도형 도구가 그리는 도형 및 기본 크기.
ReadOnly, ShowGrid, SnapToGrid, SnapToGuides, ShowPorts	뷰가 노출하는 옵션.
Zoom, MinZoom, MaxZoom, OffsetX, OffsetY, Viewport	뷰 변환.
State	진행 중인 상호작용: dgiIdle, dgiMoving, dgiResizing 등.
HoverItem, EditingNode	포인터 아래에 있는 항목과 텍스트를 편집 중인 노드.

메서드

메서드	효과
ZoomAt(aZoom, aX, aY), ScrollBy(aDX, aDY)	디바이스 상의 점을 중심으로 확대/축소하고 디바이스 픽셀 단위로 스크롤합니다.
ZoomToFit, ZoomToSelection, CenterOn	뷰와 같습니다.
SelectAll, Duplicate, Nudge(aDX, aDY)	모두 선택하고, 선택 항목을 복제하고, 월드 단위로 이동합니다.
GroupSelection, UngroupSelection	선택한 노드를 그룹화하거나 그룹 해제합니다.
Undo, Redo, CopyToClipboard, CutToClipboard, PasteFromClipboard, DeleteSelection	편집 명령.
EditText(aNode), EndTextEdit	텍스트 편집기를 열고 닫습니다.
Cancel	Esc와 마찬가지로 진행 중인 상호작용을 취소합니다.

컨트롤러의 이벤트는 뷰가 노출합니다. 이벤트를 참조하십시오. 키보드는 편집과 키보드에서 설명합니다.

예제

관련 항목

- TsgcVCLDiagramView
- 편집과 키보드

TsgcDiagramRenderer

TsgcDiagramRenderer(유닛 sgcDiagram_Render)는 뷰 없이 문서를 그립니다. PNG 및 SVG 파일이나 스트림으로, 또는 임의의 TsgcUICanvas로 그립니다. 뷰의 SaveToPNG와 SaveToSVG가 사용하며, 콘솔 애플리케이션, 서비스, Linux에서 동작합니다.

메서드

메서드	효과
SaveToPNGFile(aDocument, aFileName, aScale, aMargin), SaveToPNGStream	다이어그램 전체의 PNG. aScale의 기본값은 1, aMargin은 20 월드 단위입니다.
SaveToSVGFile(aDocument, aFileName, aScale, aMargin), SaveToSVGStream	SVG로 같은 작업을 합니다.
ExportViewport(aDocument, aScale, aMargin)	내보내기가 사용하는 뷰포트: 오프셋, 확대/축소, 배율, 이미지 크기.
RenderTo(aCanvas, aDocument, aViewport)	뷰포트를 통해 캔버스에 문서를 그립니다.
Painter	그리기를 수행하는 TsgcDiagramPainter. 그 OnDrawNode 이벤트는 내보내기에서도 노드를 사용자 지정합니다.

예제

관련 항목

- 내보내기
- 빠른 시작 헤드리스

JSON 형식

SaveToJSON은 JSON 객체 하나를 기록합니다. 출력은 정규 형식입니다. 키 순서가 고정되어 있고, 들여쓰기가 없으며, 기본값인 값은 모두 생략되므로 키가 없으면 항상 기본값을 의미합니다. 열거형은 이름으로, 색상은 16진수 문자열로, 점은 [x, y] 배열로 기록됩니다. 리더는 알 수 없는 키와 이름을 무시합니다.

문서

키	값
`format`	항상 "sgcDiagram"입니다.
`version`	형식 버전, 1. 더 새로운 버전의 파일은 EsgcDiagramError를 발생시킵니다.
`gridSize`	GridSize (10).
`theme`	"light"(기본값), "dark" 또는 "custom".
`layers`	{ name, visible (true), locked (false) } 배열.
`nodes`	노드 배열(z 순서).
`connectors`	커넥터 배열(z 순서).
`images`	{ key, png } 배열. PNG 데이터는 Base64입니다.

NodeStyle, ConnectorStyle, DefaultConnectorKind, Routing, UndoLimit은 문서 객체의 속성이며 파일에 기록되지 않습니다.

노드

키	속성
`id`	Id
`shape`	Shape: "rectangle"(기본값), "roundRect", "ellipse", "text", "image", "path", "process", "decision", "terminator", "data", "document", "predefinedProcess", "connectorCircle", "container"
`x` , `y` , `w` , `h`	Left, Top, Width (120), Height (60)
`angle`	Angle
`text`	Text
`textAlign` , `textVAlign`	TextAlign ("center"), TextVAlign ("middle")
`layer` , `container` , `group`	Layer, ContainerId, GroupId
`image` , `path`	ImageKey, PathData

`locked`, `visible`	Locked (false), Visible (true)
`data`, `tag`	Data, Tag
`style`	Style, 아래 참조
`ports`	{ id, x (0.5), y (0.5), side ("auto"), flow ("both"), caption } 배열

커넥터

키	속성
`id`	Id
`kind`	Kind: "straight", "orthogonal"(기본값) 또는 "curved"
`source`, `target`	{ node, port }: SourceNodeId와 SourcePortId, TargetNodeId와 TargetPortId
`sourcePoint`, `targetPoint`	자유 끝의 SourcePoint, TargetPoint
`waypoints`	Waypoints
`startArrow`, `endArrow`	StartArrow ("none"), EndArrow ("triangle"): "none", "triangle", "openTriangle", "diamond", "openDiamond", "circle"
`arrowSize`	ArrowSize (10)
`labels`	{ text, pos (0.5), offset (0) } 배열
`layer`, `locked`, `data`, `tag`	Layer, Locked, Data, Tag
`style`	Style, 아래 참조
`points`	저장된 경로. 불러올 때 그대로 유지됩니다

스타일

키	속성
`fill`, `stroke`	Fill, Stroke
`strokeWidth`	StrokeWidth (1)
`dash`	Dash: "solid"(기본값), "dash" 또는 "dot"
`radius`	CornerRadius
`text`	{ size, bold, color }: 텍스트 스타일

작고 완전한 파일의 예는 직렬화에 있습니다.

예제

관련 항목

- 직렬화
- TsgcDiagramDocument

계층형 레이아웃

TsgcDiagramLayeredLayout(유닛 sgcDiagram_Layout)는 방향 그래프를 Sugiyama 방법으로 계층 형태로 배치합니다. 순서도, 프로세스 다이어그램, 상태 기계, 종속성 그래프 등입니다. 커넥터는 한 계층에서 다음 계층으로 이어지며, 레이아웃은 교차를 적게 유지합니다.

속성

속성	의미
Direction	dglTopToBottom(기본값), dglLeftToRight, dglBottomToTop 또는 dglRightToLeft.
LayerSpacing	계층 사이의 간격. 기본값은 60 월드 단위이며 해당 계층에서 가장 깊은 노드에 더해집니다.
NodeSpacing	인접한 노드 사이의 간격. 기본값은 30입니다.
SelectionOnly	선택한 노드만 배치합니다.
Crossings	읽기 전용. Execute 후에 센 커넥터 교차 수.

작동 방식

- 사이클은 컬렉션 순서로 깊이 우선 탐색을 한 후 후진 간선을 뒤집어 끊습니다.
- 각 노드는 소스로부터의 가장 긴 경로에 해당하는 계층을 받습니다. 긴 커넥터는 지나가는 각 계층에 더미 노드를 가집니다.
- 무게 중심법 스윕으로 교차를 줄이고 가장 좋은 순서를 유지합니다.
- 좌표는 인접 노드의 중앙값 쪽으로 배치되며, 긴 커넥터는 직선이 되고 겹침은 없습니다.
- 더미 위치는 직선 및 곡선 커넥터의 Waypoints가 됩니다. 직교 커넥터는 라우터가 라우팅합니다.
- 연결되지 않은 부분은 NodeSpacing 간격으로 나란히 배치됩니다.

참고 사항

- 레이아웃은 최상위 노드에 대해, 또는 SelectionOnly가 True이면 선택한 노드(예: 한 컨테이너의 자식)에 대해 작동합니다. 컨테이너는 그 크기의 노드 하나로 취급되며 자식을 함께 가집니다.
- 노드는 경계의 왼쪽 위 모서리를 유지합니다. 결과는 하나의 트랜잭션으로 적용되므로 레이아웃은 하나의 실행 취소 단계가 되며, 이동한 커넥터는 Routing.AutoRoute가 False일 때도 다시 라우팅됩니다.
- 모든 좌표는 정수 100분의 1 단위로 계산되고 모든 정렬은 안정적이므로, 레이아웃은 Win32, Win64, Linux64에서 같은 결과를 냅니다.

Community 에디션

레이아웃에는 SGC_DIAGRAM_ADVANCED가 필요합니다. Community 에디션에서는 유닛 sgcDiagram_Layout가 비어 있습니다.

예제

관련 항목

- 트리 레이아웃
- TsgcDiagramDataBinding
- 에디션

트리 레이아웃

TsgcDiagramTreeLayout(유닛 sgcDiagram_Layout)는 조직도, 범주 트리, 마인드맵 같은 계층 구조를 배치합니다. 부모는 자식 위에 가운데 정렬되고, 인접한 하위 트리는 NodeSpacing 간격으로 선형 시간에 채워집니다.

속성

속성	의미
Direction	dglTopToBottom(기본값), dglLeftToRight, dglBottomToTop 또는 dglRightToLeft.
LayerSpacing	수준 사이의 간격. 기본값은 60 월드 단위이며 해당 수준에서 가장 깊은 노드에 더해집니다.
NodeSpacing	인접한 하위 트리 사이 및 쌓은 리프 사이의 간격. 기본값은 30입니다.
TreeStyle	dgoStandard(기본값) 또는 dgoCompactLeaves.
SelectionOnly	선택한 노드만 배치합니다.

작동 방식

- 루트는 들어오는 커넥터가 없는 노드이며 컬렉션 순서입니다. 커넥터는 부모에서 자식으로 향합니다. 트리 간선이 아닌 커넥터는 그려지지만 배치에서는 무시됩니다.
- 노드의 자식은 컬렉션 순서를 유지합니다.
- dgoCompactLeaves는 노드에 자식이 둘 이상이고 모두 리프인 경우, 자식을 부모 중심의 오른쪽에 한 열로 쌓습니다. 전형적인 조직도 형태입니다. 직선 및 곡선 커넥터에는 부모 아래에 엘보 Waypoints가 붙습니다.
- 여러 트리는 루트 순서로 NodeSpacing 간격으로 나란히 배치됩니다.

참고 사항

- 레이아웃은 최상위 노드에 대해, 또는 SelectionOnly가 True이면 선택한 노드(예: 한 컨테이너의 자식)에 대해 작동합니다. 컨테이너는 그 크기의 노드 하나로 취급되며 자식을 함께 가집니다.
- 노드는 경계의 왼쪽 위 모서리를 유지합니다. 결과는 하나의 트랜잭션으로 적용되므로 레이아웃은 하나의 실행 취소 단계가 되며, 이동한 커넥터는 Routing.AutoRoute가 False일 때도 다시 라우팅됩니다.
- 모든 좌표는 정수 100분의 1 단위로 계산되고 모든 정렬은 안정적이므로, 레이아웃은 Win32, Win64, Linux64에서 같은 결과를 냅니다.

Community 에디션

레이아웃에는 SGC_DIAGRAM_ADVANCED가 필요합니다. Community 에디션에서는 유닛 sgcDiagram_Layout가 비어 있습니다.

예제

관련 항목

- 계층형 레이아웃
- 데이터 바인딩

데이터 바인딩

TsgcDiagramDataBinding은 데이터셋의 행을 다이어그램으로 바꾸고 최신 상태로 유지합니다. 이 페이지에서는 일반적인 조직도 설정을 단계별로 보여 줍니다.

단계

1. 데이터에 키 필드와 부모 필드를 준비합니다. 각 행이 노드가 되고, 부모 키가 부모에서 오는 커넥터를 결정합니다. 부모를 알 수 없는 행은 루트가 됩니다.
2. 폼에 TsgcDiagramDataBinding을 놓고 DataSource, KeyField, ParentField, LabelField를 설정한 다음 Target을 다이어그램 뷰로 설정합니다.
3. 레이아웃을 선택합니다. dgbAuto는 트리에는 트리 레이아웃을, 행에 부모가 여러 개이면 계층형 레이아웃을 선택합니다(키는 ParentSeparator로 구분). 필요에 따라 Direction과 TreeStyle을 설정합니다.
4. OnCreateNode에서 노드의 스타일을 지정합니다. DataSet이 노드의 행에 위치해 있으므로 어떤 필드로든 노드의 텍스트, 색상, Data를 설정할 수 있습니다.
5. Active를 True로 설정합니다. 다이어그램이 즉시 만들어지고 데이터가 변경되면 다시 만들어집니다.

실시간 업데이트

- LiveUpdates가 True이면 DataSet을 열거나, 편집, 삽입, 삭제를 게시하거나, 행 수 또는 필터가 바뀌거나, 테이블이 알림을 보낼 때 다이어그램이 다시 만들어집니다. 요청은 메시지 루프 한 턴에 한 번의 다시 만들기로 합쳐집니다.
- 탐색은 다시 만들기를 하지 않으므로 선택이 유지됩니다. 현재 행이 아닌 행만 바꾸고 행 수는 바꾸지 않는 새로 고침은 감지되지 않으므로, 그 후에는 Refresh를 호출하십시오.
- 다시 만들기는 DataSet을 편집하는 동안과 사용자가 뷰에서 끄는 동안에는 대기합니다. DataSet을 닫아도 다이어그램은 유지됩니다.
- Delphi 10.2 이상에서는 다시 만들기가 TThread.ForceQueue로 큐에 들어갑니다. 콘솔 애플리케이션과 서비스는 CheckSynchronize를 호출해야 합니다.
- 다시 만들기는 실행 취소 기록을 지웁니다. 다이어그램에서 한 변경은 데이터에 다시 기록되지 않습니다.

Community 에디션

데이터 바인딩에는 SGC_DIAGRAM_ADVANCED가 필요합니다. Community 에디션에서는 AddNode와 AddConnector로 코드에서 다이어그램을 만듭니다.

예제

관련 항목

- TsgcDiagramDataBinding
- 트리 레이아웃
- 계층형 레이아웃

도형

모든 노드는 TsgcDiagramShape 형식의 Shape를 가집니다. 도형이 윤곽선, 텍스트를 받는 영역, 기본 포트, 테마 색상을 결정합니다. 노드는 월드 단위의 Left, Top, Width, Height로 배치되며, 중심을 기준으로 시계 방향의 도 단위 Angle로 회전합니다.

도형	용도
dgsRectangle, dgsRoundRect, dgsEllipse	기본 상자. dgsRoundRect는 Style.CornerRadius가 0이면 반지름 8의 모서리를 사용하며, 다른 상자는 CornerRadius가 0보다 클 때만 모서리를 둥글게 합니다.
dgsProcess, dgsDecision, dgsTerminator, dgsData, dgsDocument, dgsPredefinedProcess, dgsConnectorCircle	순서도 기호. 터미네이터는 항상 짧은 변의 절반으로 둥글게 처리되며, 판단은 마름모이고 기본 포트는 그 꼭짓점입니다.
dgsText	채우기도 테두리도 없는 자유 텍스트.
dgsImage	ImageKey로 지정되는 문서의 이미지. 아래를 참조하십시오.
dgsPath	PathData에 의한 사용자 지정 윤곽선. 노드에 맞게 늘어나는 0에서 100 상자 안의 SVG 경로 구문으로 지정합니다.
dgsContainer	다른 노드를 보유하는 머리글 띠가 있는 상자. 노드는 ContainerId를 통해 여기에 속하며, 컨테이너를 이동하면 자식도 이동합니다.

텍스트

텍스트는 도형의 텍스트 사각형 안에 줄 바꿈되어 TextAlign(기본값은 thaCenter)과 TextVAlign(기본값은 tvaMiddle)으로 그려집니다. 글꼴 크기, 굵기, 색상은 Style.Text에서 가져옵니다. 뷰에서는 더블 클릭, F2 또는 문자 입력으로 텍스트를 그 자리에서 편집합니다.

이미지

Images.AddPNG(aKey, aStream)는 스트림의 PNG 바이트를 저장하고 디코딩합니다. Images.Add(aKey, aImage)는 디코딩된 TsgcUIImage를 추가합니다. Shape가 dgsImage이고 ImageKey가 그 키로 설정된 노드는 상자에 맞게 늘린 이미지를 그립니다. 다른 형식은 문서의 OnDecodeImage 이벤트 핸들러에서 디코딩됩니다. 노드가 아직 사용하는 이미지를 삭제하면 EsgcDiagramError가 발생합니다.

그룹과 컨테이너

Document.Group은 선택한 노드를 하나의 그룹으로 묶고 그 GroupId를 반환합니다. 이후 클릭하면 그룹 전체가 선택되고, Alt와 클릭은 한 구성원을 선택합니다. Ungroup은 선택 항목의 그룹을 해제합니다. 컨테이너는 자식을 소유합니다. 자식은 컨테이너와 함께 이동하고, 컨테이너를 삭제하면 자식도 삭제되며, 뷰에서 컨테이너 위에 놓은 노드는 그 자식이 됩니다.

예제

관련 항목

- 포트와 접착
- 테마
- 커넥터와 라우팅

- TsgcDiagramNode

커넥터와 라우팅

TsgcDiagramConnector는 원본과 대상을 연결합니다. 각 끝은 SourceNodeId와 TargetNodeId를 통해 노드에 접착되거나, SourcePoint 와 TargetPoint 에 서 자 유 롭 습 니 다. Document.AddConnector(aSource, aTarget)는 DefaultConnectorKind(기본값은 dgcOrthogonal) 종류의 커넥터를 만들고 변경 집합이 끝날 때 라우팅합니다.

종류	경로
dgcStraight	직선 또는 Waypoints를 지나는 폴리라인.
dgcOrthogonal	수평 및 수직 세그먼트. Registered 에디션에서는 라우터가 여백을 두고 노드를 피하며 평행한 세그먼트를 떨어뜨려 배치합니다. Community에서는 경로가 단순한 엘보입니다.
dgcCurved	부드러운 곡선. Waypoints가 있으면 그것을 지납니다.

화살표, 선, 레이블

- StartArrow(기본값은 dgaNone)와 EndArrow(기본값은 dgaTriangle)는 dgaNone, dgaTriangle, dgaOpenTriangle, dgaDiamond, dgaOpenDiamond, dgaCircle을 사용할 수 있습니다. ArrowSize로 크기를 설정합니다.
- Style.Stroke, Style.StrokeWidth, Style.Dash(dgdSolid, dgdDash, dgdDot)로 선을 설정합니다.
- Labels에는 임의의 수의 TsgcDiagramLabel을 담을 수 있습니다. Position은 원본의 0에서 대상의 1까지이며, Offset은 레이블을 선에서 떨어뜨립니다. 사용자는 레이블을 커넥터를 따라 끌 수 있습니다.

라우팅

문서는 Routing.AutoRoute가 True인 동안 가장 바깥쪽 EndUpdate가 실행될 때 노드, 끝, 종류 또는 Waypoints가 변경된 모든 커넥터를 라우팅합니다. RouteAll은 모든 커넥터를 다시 라우팅합니다. 경로는 Points에 있으며, 직교 라우터가 노드를 돌아가는 경로를 찾지 못하고 대체 경로를 그린 경우 RouteFailed는 True입니다.

라우팅 속성	기본값	의미
Margin	12	직교 경로가 노드로부터 유지하는 거리.
BendPenalty	40	꺾임 한 번의 비용. 그래서 라우터는 약간 더 짧은 경로보다 꺾임이 적은 경로를 선택합니다.
Spacing	6	같은 통로를 공유하는 평행한 세그먼트 사이의 거리.
AutoRoute	True	변경 집합이 끝날 때 라우팅합니다. 코드에서 큰 다이어그램을 만드는 동안에는 끄고, 그 후 RouteAll을 호출합니다.

예제

관련 항목

- 포트와 접착
- 에디션
- 도형
- TsgcDiagramConnector

포트와 접착

포트는 커넥터의 끝이 붙는 노드 위의 점입니다. Ports 컬렉션이 비어 있는 노드는 도형의 기본 포트를 사용합니다. `left`, `top`, `right`, `bottom`이며 각 변의 중점입니다(판단 도형은 꼭짓점). 직접 포트를 추가하면 노드에 형식이 지정된 입력과 출력을 줄 수 있습니다.

TsgcDiagramPort	의미
Id	포트의 이름. 노드 안에서 고유합니다. 커넥터는 SourcePortId 또는 TargetPortId에서 이를 참조합니다.
X, Y	노드 상자에 대한 비율로 나타낸 위치. 0에서 1.
Side	dgpAuto, dgpLeft, dgpTop, dgpRight, dgpBottom: 경로가 포트에서 나가는 방향.
Flow	dgfBoth, dgfIn, dgfOut.
Caption	포트 옆의 노드 안에 그려지는 텍스트. 캡션은 확대/축소가 0.4 미만이면 숨겨집니다.

접착

- 포트 Id가 있는 커넥터의 끝은 그 포트에 접착됩니다.
- 포트 Id가 비어 있는 커넥터의 끝은 노드의 외곽선에 접착됩니다. 끝은 노드를 따라가며 중심을 향하는 선 위에서 외곽선과 만납니다.
- 노드가 없는 끝은 SourcePoint 또는 TargetPoint에서 자유롭습니다.
- Node.PortPoint(aPortId)는 포트의 월드 위치를 반환합니다.

뷰에서

ShowPorts는 포트를 그리는 시점을 결정합니다. dgvNever, dgvHover(기본값, 포인터 아래 노드의 포트), dgvAlways입니다. 포트에서 끌면 새 커넥터를 그리며, 그리는 동안 모든 포트가 대상이 됩니다. 선택한 노드에서는 크기 조정 핸들이 변의 포트를 덮으므로, 거기서 그리려면 Tool을 dgtConnector로 설정하십시오. 끝이 접착되기 전에 뷰는 OnConnecting을 발생시키며 여기서 거부할 수 있고, 커넥터가 만들어진 후에는 OnConnected를 발생시킵니다.

예제

관련 항목

- 커넥터와 라우팅
- 편집과 키보드
- TsgcDiagramPort

편집과 키보드

뷰는 컨트롤러를 통해 문서를 편집합니다. 문서를 변경하는 모든 사용자 동작은 하나의 실행 취소 단계가 되며, 끄는 동안 Esc를 누르면 취소됩니다.

도구

도구	효과
dgtSelect	기본값입니다. 클릭하면 선택하고, 끌면 이동하며, 핸들로 크기 조정과 회전을 합니다. 빈 공간에서 끌면 선택 사각형을 그리고, 포트에서 끌면 커넥터를 그립니다.
dgtPan	왼쪽 버튼으로 끌면 이동합니다.
dgtConnector	임의의 노드에서 끌면 커넥터를 그립니다.
dgtShape	클릭하거나 끌면 ShapeTool의 도형으로 노드를 만듭니다. 각 노드를 만든 후에도 도구는 활성 상태로 남으며, Esc를 누르면 dgtSelect로 돌아갑니다. 팔레트 항목을 클릭하면 이 도구가 설정됩니다.

마우스

- Shift와 클릭은 선택에 추가하고, Ctrl과 클릭은 항목의 선택을 전환하며, Alt와 클릭은 그룹의 한 구성원을 선택합니다.
- 크기를 조정하는 동안 Shift를 누르면 가로세로 비율이 유지되고, 회전하는 동안 Shift를 누르면 각도가 15도 단위로 맞춰집니다.
- 휠은 세로로 스크롤하고, Shift와 휠은 가로로 스크롤하며, Ctrl과 휠은 포인터를 기준으로 확대/축소합니다.
- 가운데 버튼 또는 Space를 누른 채 왼쪽 버튼으로 이동합니다.
- 오른쪽 버튼은 포인터 아래의 항목을 선택하여 팝업 메뉴를 준비합니다.
- 노드를 더블 클릭하면 텍스트를 편집하고 OnItemDblClick이 발생합니다.

키보드

키	동작
Ctrl+Plus, Ctrl+Minus	뷰의 중심을 기준으로 확대하고 축소합니다.
Ctrl+0, Ctrl+1	확대/축소 1, 화면에 맞춤.
Ctrl+A, Ctrl+C	모두 선택, 복사.
Ctrl+X, Ctrl+V, Ctrl+D	잘라내기, 붙여넣기, 복제.
Ctrl+Z, Ctrl+Y 또는 Ctrl+Shift+Z	실행 취소, 다시 실행.
Ctrl+G, Ctrl+Shift+G	그룹화, 그룹 해제.
Delete	선택 항목을 삭제합니다(잠긴 항목은 남습니다).
화살표 키	선택 항목을 GridSize만큼 이동합니다. Alt를 누르면 1 월드 단위입니다.
F2 또는 아무 문자	선택한 노드의 텍스트를 편집합니다.
Tab, Shift+Tab	다음 또는 이전 항목을 선택합니다.

Esc	끌기를 취소하거나, 현재 도구를 종료하거나, 선택을 해제합니다.

ReadOnly 모드에서는 확대/축소 키, Ctrl+A, Ctrl+C, Tab, Esc만 작동합니다.

텍스트 편집

텍스트는 노드 위의 메모에서 편집합니다. Enter를 누르거나 다른 곳을 클릭하면 확정되고, Esc는 취소하며, Shift+Enter는 새 줄을 시작합니다. OnTextEditing은 aAllow를 False로 설정하여 편집을 거부할 수 있고, OnTextEdited는 확정 후에 발생합니다. EditText(aNode)는 코드에서 편집기를 시작합니다.

예제

관련 항목

- 스냅과 가이드
- 실행 취소와 클립보드
- 뷰어 모드
- TsgcDiagramController
- 이벤트

스냅과 가이드

사용자가 노드를 이동하거나 크기를 조정하거나 만드는 동안 두 가지 보조 기능이 노드를 정렬합니다.

속성	효과
Diagram.GridSize	월드 단위의 격자 간격. 기본값은 10입니다. 화살표 키는 선택 항목을 이 값만큼 이동합니다.
ShowGrid	뷰에 격자를 그립니다.
SnapToGrid	이동, 크기 조정, 생성된 노드와 놓은 도형을 격자에 맞춥니다.
SnapToGuides	정렬 가이드를 표시합니다. 노드를 이동하거나 크기를 조정하는 동안 그 가장자리와 중심선이 근처 노드의 가장자리와 중심선에 맞춰지고 가이드 선이 그려집니다.

- 가이드는 400 디바이스 픽셀 이내의 노드에 대해 5 디바이스 픽셀 이내의 가장자리 또는 중심을 잡습니다.
- 가이드는 회전하지 않은 노드에 적용됩니다.
- 크기를 조정하는 동안 Shift를 누르면 해당 끌기 동안 가이드가 꺼집니다.
- Alt와 화살표 키는 격자와 무관하게 선택 항목을 1 월드 단위씩 이동합니다.

예제

관련 항목

- 편집과 키보드
- 도형

실행 취소와 클립보드

문서는 실행 취소 스택과 다시 실행 스택을 유지합니다. 뷰에서의 모든 사용자 동작은 한 단계입니다. 코드에서는 BeginTransaction(aCaption)과 EndTransaction이 임의의 수의 변경을 한 단계로 묶고, Rollback은 열려 있는 트랜잭션의 변경을 버립니다.

- Undo, Redo, CanUndo, CanRedo, UndoCaption, RedoCaption은 스택에 대해 작동합니다. 뷰와 컨트롤러에도 Undo와 Redo가 있습니다.
- UndoLimit(기본값은 100)은 보관하는 단계 수입니다.
- 트랜잭션 밖에서 코드로 한 변경은 두 스택을 모두 지우므로, 실행 취소가 오래된 상태를 복원하는 일은 없습니다.
- BeginUndoSuspend와 EndUndoSuspend는 기록하지 않아야 하는 대량 다시 만들기를 감쌉니다. 스택을 지우고 단계를 만들지 않으며, 뷰에는 계속 알립니다. 중첩할 수 있습니다.
- Document.Clear는 실행 취소 단계가 아닙니다.

클립보드

클립보드는 애플리케이션 내부용입니다. CopySelection은 선택한 노드, 선택한 컨테이너의 자식, 양 끝이 복사되는 커넥터, 그리고 이들이 사용하는 이미지를 JSON 프래그먼트로 반환하고 앱 내 클립보드에 보관합니다. Paste(aJSON)는 프래그먼트를 하나의 실행 취소 단계로 삽입하고 붙여넣은 항목을 선택합니다. 인수 없는 Paste는 앱 내 클립보드를 붙여넣으며, 반복할 때마다 20월드 단위씩 옮겨 놓습니다. 뷰에는 CopyToClipboard, CutToClipboard, PasteFromClipboard, DeleteSelection이 있으며, Ctrl+D는 선택 항목을 복제합니다.

예제

관련 항목

- 직렬화
- 편집과 키보드

레이어

레이어는 다이어그램의 항목을 정리합니다. 예를 들어 작업 레이어 아래에 프레임과 메모를 두는 배경 레이어를 둘 수 있습니다. 각 노드와 커넥터는 Layer 속성에서 레이어를 이름으로 지정하며, 이름이 비어 있으면 기본 레이어입니다.

TsgcDiagramLayer	의미
Name	항목의 Layer 속성이 사용하는 이름.
Visible	False로 하면 뷰와 내보내기에서 해당 레이어의 항목이 숨겨집니다.
Locked	True로 하면 해당 레이어의 항목을 선택하거나 편집할 수 없습니다.

- Layers.Add, Layers.Find(aName), 컬렉션의 Items로 레이어를 관리합니다. 레이어의 추가, 삭제, 이름 변경은 실행 취소 단계가 되지만 Visible과 Locked는 그렇지 않습니다.
- Document.LayerVisible(aName)와 LayerLocked(aName)은 레이어를 처리하는 방식을 알려 줍니다.
- 개별 항목도 자체 Locked 속성으로 잠글 수 있습니다.

Community 에디션

SGC_DIAGRAM_ADVANCED가 없으면 레이어를 하나만 추가할 수 있으며, 두 번째 Layers.Add는 EsgcDiagramError를 발생시킵니다. 레이어가 여러 개인 파일도 그대로 불러오고 저장할 수 있지만, 모든 레이어가 표시 및 잠금 해제 상태로 그려지고 내보내지고 편집됩니다.

예제

관련 항목

- 에디션
- 뷰어 모드
- TsgcDiagramLayer

직렬화

문서는 JSON으로 저장하고 불러옵니다. 형식은 모든 플랫폼과 모든 에디션에서 같습니다.

메서드	효과
SaveToJSON, LoadFromJSON	문자열과의 상호 변환.
SaveToFile, LoadFromFile	파일과의 상호 변환.
SaveToStream, LoadFromStream	스트림과의 상호 변환.
CopySelection, Paste	문서의 일부분. 실행 취소와 클립보드를 참조하십시오.

- 출력은 정규 형식입니다. 키 순서가 고정되어 있고, 기본값인 속성은 생략되며, 들여쓰기가 없습니다. 같은 문서에서는 항상 같은 텍스트가 나옵니다.
- 문서의 이미지는 파일 안에 PNG 데이터로 저장됩니다.
- 리더는 관대합니다. 알 수 없는 키와 알 수 없는 열거형 이름은 무시됩니다. 더 새로운 형식 버전으로 기록된 파일은 EsgcDiagramError를 발생시키며, 구문 분석 오류는 그 위치를 보고합니다.
- 불러오기는 문서의 내용을 대체합니다. 파일에 저장된 커넥터 경로는 유지됩니다.
- 뷰는 다이어그램을 dfm 또는 fmx 파일에 DiagramJSON 속성으로 저장하며, 다이어그램이 비어 있지 않을 때만 기록합니다.

예제

작은 파일

노드 두 개와 커넥터 하나입니다. 읽기 쉽도록 여기서는 들여쓰기를 했습니다.

```
{
  "format": "sgcDiagram",
  "version": 1,
  "nodes": [
    {
      "id": "n1",
      "shape": "terminator",
      "x": 40,
      "y": 40,
      "w": 140,
      "h": 50,
      "text": "Start"
    },
    {
      "id": "n2",
      "shape": "process",
      "x": 40,
      "y": 140,
      "w": 140,
      "text": "Save"
    }
  ],
  "connectors": [
    {
      "id": "c3",
      "source": {
        "node": "n1"
      },
      "target": {
        "node": "n2"
```

```
      },
      "labels": [
        {
          "text": "Go"
        }
      ],
      "points": [[110, 90], [110, 140]]
    }
  ]
}
```

관련 항목

- 실행 취소와 클립보드
- 내보내기
- JSON 형식

내보내기

다이어그램은 PNG와 SVG로 내보냅니다. 이미지는 문서의 콘텐츠 경계에 여백을 더한 범위를 포함합니다.

메서드	비고
View.SaveToPNG(aFileName or aStream, aScale)	뷰의 다이어그램을 FontFile과 OnDrawNode를 적용하여 내보냅니다. aScale은 월드 단위당 픽셀 수이며 기본값은 1입니다.
View.SaveToSVG(aFileName or aStream, aScale)	SVG로 같은 작업을 합니다.
TsgcDiagramRenderer.SaveToPNGFile, SaveToPNGStream	(aDocument, aFileName 또는 aStream, aScale = 1, aMargin = 20). 뷰가 필요 없습니다.
TsgcDiagramRenderer.SaveToSVGFile, SaveToSVGStream	SVG로 같은 작업을 합니다.
TsgcDiagramRenderer.RenderTo	(aCanvas, aDocument, aViewport)는 문서의 일부를 sgcUICore의 TsgcUICanvas에 그립니다.

- 내보내기는 문서의 테마를 사용하므로 다크 다이어그램은 다크로 내보내집니다.
- 렌더러는 콘솔 애플리케이션, 서비스, Linux64에서 동작합니다. 빠른 시작 헤드리스를 참조하십시오.
- PDF 내보내기와 인쇄는 버전 1에 포함되지 않습니다.

예제

관련 항목

- 테마
- 직렬화
- TsgcDiagramRenderer

테마

뷰, 팔레트, 내보내기는 sgcUICore의 테마로 그려지므로 다이어그램은 같은 폼의 다른 sgcUI 제품과 어울립니다.

- 뷰, 팔레트, 문서의 **Theme**은 utLight(기본값), utDark 또는 utCustom입니다. 뷰의 Theme은 해당 Diagram의 Theme을 설정합니다.
- 각 노드와 커넥터의 **Style**: Fill, Stroke, StrokeWidth, Dash, CornerRadius, Text(Size, Bold, Color). 색상은 ARGB TsgcUIColor이며, 예를 들면 sgcUIARGB($FF, $00, $96, $88)입니다. 0이면 도형에 따라 달라지는 테마의 색상이 유지됩니다.
- 문서의 **NodeStyle**과 **ConnectorStyle**은 새 노드와 커넥터마다 복사되는 템플릿입니다.
- 뷰의 **FontFile**은 sgcUICore에 내장된 글꼴 대신 TrueType 파일로 텍스트를 그립니다.
- 뷰의 **OnDrawNode**를 사용하면 노드를 직접 그리거나 기본 그리기에 추가할 수 있습니다. 기본 그리기를 건너뛰려면 aDefault를 False로 설정합니다.

예제

관련 항목

- 내보내기
- 도형
- TsgcDiagramStyle

뷰어 모드

ReadOnly를 True로 설정하면 사용자가 살펴볼 수는 있지만 변경할 수는 없는 다이어그램을 표시할 수 있습니다. 대시보드의 프로세스 맵이나 생성된 조직도 등입니다.

- 사용자는 항목을 선택하고, Ctrl과 휠 또는 키로 확대/축소하고, 휠, 가운데 버튼, Space 또는 이동 도구로 이동하고, 선택 항목을 복사할 수 있습니다.
- 이동, 크기 조정, 연결, 삭제, 붙여넣기, 편집은 모두 불가능하며 팔레트에서 도형을 놓을 수도 없습니다.
- 포인터는 항목 위에서 손 모양이 되고, OnItemClick, OnItemDblClick, OnHover가 해당 항목을 알려 주므로 애플리케이션은 상세 페이지를 열거나 힌트를 표시할 수 있습니다.
- ZoomToFit, ZoomToSelection, CenterOn은 코드에서 뷰를 이동합니다. MinZoom(0.05)과 MaxZoom(8)이 확대/축소 범위를 제한합니다.

일부 항목만 보호하려면 ReadOnly를 False로 두고 해당 항목이나 그 레이어에 Locked를 설정합니다.

예제

관련 항목

- 편집과 키보드
- 레이어
- TsgcVCLDiagramView
- TsgcFMXDiagramView

.NET 개요

sgcDiagram .NET은 Delphi sgcDiagram 라이브러리를 관리 코드 C#으로 이식한 것이며 래퍼가 아닙니다. C# 클래스는 **esegece.sgcDiagram** 네임스페이스에 있고 Delphi 이름을 그대로 유지하므로, **TsgcDiagramDocument**, **TsgcDiagramNode**, **TsgcDiagramConnector**, **TsgcDiagramController**, **TsgcDiagramRenderer**와 열거형은 두 언어에서 똑같이 읽힙니다. 뷰와 팔레트는 모든 호스트에서 Delphi 명명 규칙을 따릅니다. **TsgcVCLDiagramView**에 대응하는 것이 **TsgcWinFormsDiagramView**, **TsgcWpfDiagramView**, **TsgcAvaloniaDiagramView** 이고, **TsgcVCLDiagramPalette** 에 대응하는 것이 **TsgcWinFormsDiagramPalette**, **TsgcWpfDiagramPalette**, **TsgcAvaloniaDiagramPalette**입니다. 컨트롤은 **sgcUICore**의 .NET 이식판으로 그려지므로, eSeGeCe 비주얼 스위트의 다른 제품과 같은 캔버스, 글꼴, 밝은 테마와 어두운 테마를 사용합니다.
문서, 실행 취소 스택, JSON 형식, 라우터, 레이아웃, 페인터, 컨트롤러는 Delphi 코드를 한 줄씩 이식한 것입니다. .NET 애플리케이션에서 저장한 다이어그램은 Delphi 애플리케이션에서 열 수 있고 그 반대도 가능합니다. JSON 텍스트와 SVG 내보내기는 Delphi와 같고, 레이아웃과 경로도 같은 좌표를 만듭니다.

패키지

포트는 5개의 NuGet 패키지로 나뉘어 있으므로, 프로젝트는 사용하는 호스트만 가져옵니다.

패키지 ID	포함 내용	대상 프레임워크
esegece.sgcDiagram	노드, 커넥터, 포트, 레이블, 레이어, 이미지, 스타일을 가진 문서, 실행 취소와 다시 실행, JSON 형식과 애플리케이션 내부 클립보드, 라우터, 계층형 레이아웃과 트리 레이아웃, 팔레트 모델, 컨트롤러, 페인터, **TsgcDiagramRenderer**의 SVG 내보내기, **TsgcDiagramDataBinding**을 포함합니다. 네이티브 종속성은 없습니다. esegece.sgcUICore에 의존합니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcDiagram.Rendering	PNG 출력입니다. **TsgcDiagramRenderer**의 **SaveToPNGStream**과 **SaveToPNGFile** 확장 메서드와 뷰 캐시의 래스터를 esegece.sgcUICore.Rendering의 SkiaSharp 캔버스에 그립니다. Windows, Linux, macOS에서 동작합니다.	netstandard2.0, net472, net6.0, net8.0, net9.0
esegece.sgcDiagram.WinForms	**TsgcWinFormsDiagramView**와 **TsgcWinFormsDiagramPalette**.	net472, net8.0-windows, net9.0-windows
esegece.sgcDiagram.Wpf	**TsgcWpfDiagramView**와 **TsgcWpfDiagramPalette**.	net472, net8.0-windows, net9.0-windows
esegece.sgcDiagram.Avalonia	**TsgcAvaloniaDiagramView**와 **TsgcAvaloniaDiagramPalette**. Avalonia 11을 사용하며 Windows, Linux, macOS에서 동작합니다.	net8.0, net9.0

세 호스트 패키지는 esegece.sgcDiagram.Rendering에 의존하고 이것은 다시 esegece.sgcDiagram에 의존하므로, `PackageReference` 한 줄로 호스트에 필요한 모든 것이 들어옵니다. NuGet 설치를 참조하십시오.

에디션

에디션	패키지 ID	포함 항목
Registered	esegece.sgcDiagram, .Rendering, .WinForms, .Wpf, .Avalonia	모든 기능입니다. 도형, 커넥터, 포트와 접착, 편집, 스냅과 가이드, 실행 취소와 클립보드, 레이

		어, JSON, PNG와 SVG 내보내기, 테마, 뷰어 모드, 그리고 고급 기능인 계층형 레이아웃과 트리 레이아웃, 장애물을 피하고 넛징을 적용하는 직교 라우팅, 데이터 바인딩, 둘 이상의 레이어를 포함합니다.
Community	같은 ID에 .Community 접미사가 붙습니다(예: esegece.sgcDiagram.WinForms.Community).	고급 기능을 제외한 모든 기능입니다. **TsgcDiagramLayeredLayout**, **TsgcDiagramTreeLayout**, **TsgcDiagramDataBinding**은 Community 패키지에 없고, **dgcOrthogonal** 커넥터는 단순한 꺾임선으로 라우팅되며, 레이어가 이미 있으면 **Layers.Add**가 **EsgcDiagramError**를 발생시킵니다.

이는 Delphi 에디션과 같은 구분이며, 다이어그램 파일은 두 에디션에서 같습니다. 에디션을 참조하십시오.

호스트

호스트	Delphi 대응	플랫폼
WinForms	VCL (TsgcVCLDiagramView, TsgcVCLDiagramPalette)	Windows, .NET Framework 4.7.2, .NET 8 및 9.
WPF	VCL과 FMX	Windows, .NET Framework 4.7.2, .NET 8 및 9. 컨트롤은 XAML에서 선언할 수 있습니다.
Avalonia	FMX (TsgcFMXDiagramView, TsgcFMXDiagramPalette), 터치 화면에서 핀치 확대와 한 손가락 이동 지원	Windows, Linux, macOS, .NET 8 및 9.
헤드리스	TsgcDiagramDocument와 TsgcDiagramRenderer	대상 프레임워크의 모든 플랫폼에서 동작합니다. JSON과 SVG는 esegece.sgcDiagram, PNG는 esegece.sgcDiagram.Rendering을 사용합니다.

모든 호스트는 같은 API를 가집니다. Delphi 뷰와 팔레트의 속성, 메서드, 이벤트가 같은 이름으로 제공됩니다. 클래스 이름은 .NET 클래스를, Delphi와 달라지는 점은 Delphi와 .NET의 차이점를 참조하십시오.

참고 항목

- NuGet 설치
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 클래스
- Delphi와 .NET의 차이점
- .NET 데이터 바인딩
- .NET 헤드리스 출력
- .NET 데모
- 플랫폼

NuGet 설치

패키지는 nuget.org에 있습니다. 호스트의 패키지를 추가하면 코어와 렌더링 패키지가 종속성으로 함께 들어옵니다.

Registered 에디션

```
dotnet add package esegece.sgcDiagram.WinForms
dotnet add package esegece.sgcDiagram.Wpf
dotnet add package esegece.sgcDiagram.Avalonia
```

Community 에디션

```
dotnet add package esegece.sgcDiagram.WinForms.Community
dotnet add package esegece.sgcDiagram.Wpf.Community
dotnet add package esegece.sgcDiagram.Avalonia.Community
```

또는 프로젝트 파일에 참조를 직접 추가합니다.

```
<ItemGroup>
  <PackageReference Include="esegece.sgcDiagram.Wpf" Version="2026.10.0" />
</ItemGroup>
```

프로젝트에서는 Registered 또는 Community 패키지 중 하나만 참조하고, 둘을 함께 참조하지 마십시오. 같은 네임스페이스에 같은 형식이 들어 있습니다.

프로젝트 설정

- WinForms: `net8.0-windows` 프로젝트에 `<UseWindowsForms>true</UseWindowsForms>`를 지정하거나 .NET Framework 4.7.2 프로젝트를 사용합니다.
- WPF: `<UseWPF>true</UseWPF>`를 지정한 `net8.0-windows` 프로젝트 또는 .NET Framework 4.7.2 프로젝트입니다. XAML에서는 `xmlns:sgc="clr-namespace:esegece.sgcDiagram;assembly=esegece.sgcDiagram.Wpf"`로 네임스페이스를 매핑합니다.
- Avalonia: `net8.0` 또는 `net9.0` 프로젝트이며, Avalonia 11.3의 **Avalonia.Desktop**과 **Avalonia.Themes.Fluent** 패키지를 사용합니다. 일반적인 Avalonia 데스크톱 애플리케이션과 같습니다. 제자리 텍스트 편집기는 Avalonia **TextBox**이며 모양은 Fluent 테마를 따릅니다.
- 헤드리스: 문서, JSON, SVG에는 esegece.sgcDiagram만으로 충분합니다. PNG에는 esegece.sgcDiagram.Rendering을 추가하십시오. SkiaSharp와 Windows 및 Linux용 네이티브 자산이 함께 들어옵니다.

C#에서는 모든 호스트에 `using esegece.sgcDiagram;`이 필요하며, `using esegece.sgcUICore;`는 **TsgcUITheme**, **TsgcUINotifyEvent** 등 sgcUICore의 다른 형식에 필요합니다.

참고 항목

- .NET 개요
- 빠른 시작 WinForms
- 빠른 시작 WPF
- 빠른 시작 Avalonia
- .NET 데모

빠른 시작 WinForms

이 빠른 시작은 왼쪽에 도형 팔레트를, 나머지 영역에 다이어그램 뷰를 배치하고 뷰에 작은 순서도를 표시하는 폼을 코드로 만듭니다. 팔레트에서 도형을 뷰로 끌어 놓거나, 도형을 클릭한 다음 뷰를 클릭하고, 포트에서 두 노드를 연결합니다.

단계

1. `net8.0-windows` Windows Forms 프로젝트를 만들고 **esegece.sgcDiagram.WinForms** 패키지(Community 에디션에서는 **esegece.sgcDiagram.WinForms.Community**)를 추가합니다.
2. `using esegece.sgcDiagram;`과 `using esegece.sgcUICore;`를 추가합니다.
3. **TsgcWinFormsDiagramView**와 **TsgcWinFormsDiagramPalette**를 만들어 폼에 추가합니다. 이들은 일반 WinForms 컨트롤이므로 Dock, Anchor, TabIndex, Enabled가 평소처럼 동작합니다.
4. 팔레트의 **View**를 뷰로 설정합니다. 팔레트는 만들어질 때 Flowchart와 Basic 도형을 불러옵니다.
5. 뷰의 **TsgcDiagramDocument**인 **Diagram**을 트랜잭션 안에서 채우면 전체 설정이 실행 취소 한 단계가 됩니다. 그런 다음 이벤트를 지정합니다.

예제

```
using System;
using System.Windows.Forms;
using esegece.sgcDiagram;
using esegece.sgcUICore;

public class DiagramForm : Form
{
    public readonly TsgcWinFormsDiagramPalette Palette;
    public readonly TsgcWinFormsDiagramView View;

    public DiagramForm()
    {
        Text = "sgcDiagram";
        Width = 900;
        Height = 600;

        View = new TsgcWinFormsDiagramView { Dock = DockStyle.Fill };
        View.ShowGrid = true;
        View.SnapToGrid = true;
        View.ShowPorts = TsgcDiagramShowPorts.dgvHover;
        // the palette has the Flowchart and Basic shapes; a click arms the
        // shape tool of the view, a drag drops the shape on the view
        Palette = new TsgcWinFormsDiagramPalette { Dock = DockStyle.Left, Width = 180 };
        Palette.View = View;
        Controls.Add(View);
        Controls.Add(Palette);

        // three nodes and two connectors, one undo step
        TsgcDiagramDocument oDiagram = View.Diagram;
        oDiagram.BeginTransaction("Create");
        try
        {
            TsgcDiagramNode oStart = oDiagram.AddNode(TsgcDiagramShape.dgsTerminator, 40, 40, 140, 50, "Start");
            TsgcDiagramNode oCheck = oDiagram.AddNode(TsgcDiagramShape.dgsDecision, 30, 140, 160, 90, "Valid?");
            TsgcDiagramNode oDone = oDiagram.AddNode(TsgcDiagramShape.dgsProcess, 40, 290, 140, 60, "Save");
            oDiagram.AddConnector(oStart, oCheck);
            oDiagram.AddConnector(oCheck, oDone).Labels.Add().Text = "Yes";
        }
        finally
        {
            oDiagram.EndTransaction();
        }
```

```
        View.OnSelectionChange += Sender =>
            Text = "sgcDiagram, " + View.Diagram.Selection.Count + " selected";
        Shown += (s, e) => View.ZoomToFit();
    }

    public void SetDark(bool aDark)
    {
        TsgcUITheme vTheme = aDark ? TsgcUITheme.utDark : TsgcUITheme.utLight;
        View.Theme = vTheme;
        Palette.Theme = vTheme;
    }
}

public static class Program
{
    [STAThread]
    public static void Start()
    {
        Application.EnableVisualStyles();
        Application.Run(new DiagramForm());
    }
}
```

참고

- **Tool**은 누름 동작(**dgtSelect**, **dgtPan**, **dgtConnector**, **ShapeTool**과 함께 쓰는 **dgtShape**)을 선택하고, **ReadOnly**는 뷰를 뷰어로 바꿉니다. Delphi와 같습니다.
- 키보드는 Delphi와 같습니다. 편집과 키보드를 참조하십시오. 제자리 편집기는 여러 줄 **TextBox**이며 **TextEditor**로 노출됩니다. Enter는 확정, Shift+Enter는 줄 바꿈, Esc는 취소입니다.
- **Theme**은 컨트롤마다 설정하므로 뷰와 팔레트 모두에 설정하십시오. 뷰는 이를 자신의 문서에 전달합니다.
- 다이어그램은 숨겨진 속성 **DiagramJSON**에 저장되므로, 디자이너로 배치한 뷰는 그 안에 그린 다이어그램을 유지합니다.
- **SaveToPNG**와 **SaveToSVG**는 배율을 적용해 다이어그램 전체를 파일이나 스트림에 저장합니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 VCL

빠른 시작 WPF

WPF 컨트롤은 XAML로 선언할 수 있습니다. 이 창은 왼쪽에 팔레트를, 나머지 영역에 뷰를 배치하며, 코드 비하인드에서 팔레트를 뷰에 연결하고 작은 순서도를 추가합니다.

단계

1. `net8.0-windows` WPF 프로젝트를 만들고 **esegece.sgcDiagram.Wpf** 패키지(Community 에디션에서는 **esegece.sgcDiagram.Wpf.Community**)를 추가합니다.
2. 창에서 네임스페이스를 매핑합니다. `xmlns:sgc="clr-namespace:esegece.sgcDiagram;assembly=esegece.sgcDiagram.Wpf"`
3. **TsgcWpfDiagramPalette**와 **TsgcWpfDiagramView**를 선언하고 속성을 특성으로 설정합니다.
4. 코드 비하인드에서 팔레트의 **View**를 설정하고 **Diagram**을 채운 다음 이벤트를 지정합니다.

XAML

```
<Window x:Class="QuickStart.QuickStartWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sgc="clr-namespace:esegece.sgcDiagram;assembly=esegece.sgcDiagram.Wpf"
        Title="sgcDiagram" Width="900" Height="600">
  <DockPanel>
    <sgc:TsgcWpfDiagramPalette x:Name="palShapes" DockPanel.Dock="Left" Width="180" />
    <sgc:TsgcWpfDiagramView x:Name="dgmFlow" ShowGrid="True" SnapToGrid="True" ShowPorts="dgvHover" />
  </DockPanel>
</Window>
```

코드 비하인드

```
using System.Windows;
using esegece.sgcDiagram;

namespace QuickStart
{
    public partial class QuickStartWindow : Window
    {
        public QuickStartWindow()
        {
            InitializeComponent();
            // the palette drives the view: click a shape, or drag it onto the view
            palShapes.View = dgmFlow;

            TsgcDiagramDocument oDiagram = dgmFlow.Diagram;
            oDiagram.BeginTransaction("Create");
            try
            {
                TsgcDiagramNode oStart = oDiagram.AddNode(TsgcDiagramShape.dgsTerminator, 40, 40, 140, 50, "Start");
                TsgcDiagramNode oCheck = oDiagram.AddNode(TsgcDiagramShape.dgsDecision, 30, 140, 160, 90, "Valid?");
                TsgcDiagramNode oDone = oDiagram.AddNode(TsgcDiagramShape.dgsProcess, 40, 290, 140, 60, "Save");
                oDiagram.AddConnector(oStart, oCheck);
                oDiagram.AddConnector(oCheck, oDone).Labels.Add().Text = "Yes";
            }
            finally
            {
                oDiagram.EndTransaction();
            }
            dgmFlow.OnItemDblClick += (Sender, aItem) => Title = "sgcDiagram, " + aItem.Id;
            Loaded += (s, e) => dgmFlow.ZoomToFit();
            // stops the timer of the view and releases its controller
```

```
            Closed += (s, e) => dgmFlow.Dispose();
        }
    }
}
```

참고

- 속성은 일반 CLR 속성이므로 XAML에서 특성 값으로 설정합니다. 팔레트의 **View**는 뷰 객체를 받으므로 코드에서 설정하십시오.
- 다이어그램은 **DiagramJSON** 특성으로 XAML에서 지정할 수 있습니다. JSON 텍스트는 중괄호로 시작하므로 `{}` 이스케이프 뒤에 쓰십시오.
- API의 좌표는 **ScreenToDiagram**과 **DiagramToScreen**을 포함하여 컨트롤의 장치 픽셀이며, 장치 독립 픽셀에 DPI 배율을 곱한 값입니다.
- 창을 닫을 때 뷰의 **Dispose**를 호출하십시오. 뷰의 타이머를 중지하고 컨트롤러를 해제합니다.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 VCL

빠른 시작 Avalonia

Avalonia 컨트롤은 Windows, Linux, macOS에서 실행됩니다. 이 빠른 시작은 왼쪽에 팔레트를, 나머지 영역에 뷰를 배치한 창을 어두운 테마로 코드에서 만듭니다.

단계

1. Avalonia 11.3과 Fluent 테마를 사용하는 `net8.0` Avalonia 데스크톱 프로젝트를 만들고 **esegece.sgcDiagram.Avalonia** 패키지(Community 에디션에서는 **esegece.sgcDiagram.Avalonia.Community**)를 추가합니다.
2. **TsgcAvaloniaDiagramView**와 **TsgcAvaloniaDiagramPalette**를 만들고, 팔레트의 **View**를 설정한 다음 둘 다 패널에 배치합니다.
3. **Diagram**을 채우고 이벤트를 지정합니다.

예제

```
using System;
using Avalonia;
using Avalonia.Controls;
using Avalonia.Controls.ApplicationLifetimes;
using Avalonia.Themes.Fluent;
using esegece.sgcDiagram;
using esegece.sgcUICore;

public class App : Application
{
    public override void Initialize()
    {
        Styles.Add(new FluentTheme());
    }

    public override void OnFrameworkInitializationCompleted()
    {
        if (ApplicationLifetime is IClassicDesktopStyleApplicationLifetime vDesktop)
            vDesktop.MainWindow = new DiagramWindow();
        base.OnFrameworkInitializationCompleted();
    }
}

public class DiagramWindow : Window
{
    public readonly TsgcAvaloniaDiagramPalette Palette;
    public readonly TsgcAvaloniaDiagramView View;

    public DiagramWindow()
    {
        Title = "sgcDiagram";
        Width = 900;
        Height = 600;

        View = new TsgcAvaloniaDiagramView { ShowGrid = true, SnapToGrid = true };
        View.Theme = TsgcUITheme.utDark;
        Palette = new TsgcAvaloniaDiagramPalette { Width = 180, View = View };
        Palette.Theme = TsgcUITheme.utDark;

        TsgcDiagramDocument oDiagram = View.Diagram;
        oDiagram.BeginTransaction("Create");
        try
        {
            TsgcDiagramNode oStart = oDiagram.AddNode(TsgcDiagramShape.dgsTerminator, 40, 40, 140, 50, "Start");
```

```
            TsgcDiagramNode oCheck = oDiagram.AddNode(TsgcDiagramShape.dgsDecision, 30, 140, 160, 90, "Valid?");
            TsgcDiagramNode oDone = oDiagram.AddNode(TsgcDiagramShape.dgsProcess, 40, 290, 140, 60, "Save");
            oDiagram.AddConnector(oStart, oCheck);
            oDiagram.AddConnector(oCheck, oDone).Labels.Add().Text = "Yes";
        }
        finally
        {
            oDiagram.EndTransaction();
        }
        // Avalonia: the events are delegate properties, assigned with =
        View.OnSelectionChange = Sender => Title = "sgcDiagram, " + View.Diagram.Selection.Count + " selected";

        DockPanel pnlMain = new DockPanel();
        DockPanel.SetDock(Palette, Dock.Left);
        pnlMain.Children.Add(Palette);
        pnlMain.Children.Add(View);
        Content = pnlMain;
        Opened += (s, e) => View.ZoomToFit();
        // Avalonia does not dispose controls: release the view with the window
        Closed += (s, e) => View.Dispose();
    }
}

public static class Program
{
    [STAThread]
    public static void Start(string[] args)
    {
        AppBuilder.Configure<App>().UsePlatformDetect().StartWithClassicDesktopLifetime(args);
    }
}
```

참고

- Avalonia 컨트롤의 이벤트는 델리게이트 속성이며 =로 지정합니다. WinForms와 WPF 컨트롤에는 C# 이벤트가 있으며 +=로 지정합니다.
- Avalonia 컨트롤의 **Theme**은 Avalonia의 **Theme** 속성(컨트롤 테마)을 숨기며 **TsgcUITheme**을 받습니다.
- 터치 화면에서는 핀치로 확대/축소하고 빈 공간에서 한 손가락으로 이동합니다. 노드 위의 한 손가락은 해당 노드를 끕니다. FMX 뷰와 같습니다.
- 팔레트 드래그는 `DragDrop.DoDragDropAsync`를 뷰의 애플리케이션 형식 **DragFormat**(**CS_DIAGRAM_AVALONIA_DRAG_FORMAT**)과 함께 사용합니다. 자체 드래그 원본에서는 **DragPayloadOf**와 **CreateDragData**로 그 데이터를 읽고 만듭니다.
- Avalonia는 컨트롤을 해제하지 않으므로 창을 닫을 때 뷰의 **Dispose**를 호출하십시오.

참고 항목

- .NET 개요
- NuGet 설치
- .NET 클래스
- Delphi와 .NET의 차이점
- 빠른 시작 FMX

.NET 클래스

C# 클래스는 Delphi 클래스와 같은 이름을 가집니다. 모델, 엔진, 페인터의 클래스는 모든 호스트에서 같고, 컨트롤은 호스트마다 클래스가 하나씩 있습니다.

컨트롤

Delphi	WinForms	WPF	Avalonia
TsgcVCLDiagramView, TsgcFMXDiagramView	TsgcWinFormsDiagramView	TsgcWpfDiagramView	TsgcAvaloniaDiagramView
TsgcVCLDiagramPalette, TsgcFMXDiagramPalette	TsgcWinFormsDiagramPalette	TsgcWpfDiagramPalette	TsgcAvaloniaDiagramPalette
기본 클래스	TsgcUIWinFormsControl	TsgcUIWpfControl	TsgcUIAvaloniaControl

뷰에는 Delphi와 같은 속성 **Diagram**, **Controller**, **Tool**, **ShapeTool**, **ReadOnly**, **Zoom**, **MinZoom**, **MaxZoom**, **ShowGrid**, **SnapToGrid**, **SnapToGuides**, **ShowPorts**, **Theme**, **FontFile**이 있고, 여기에 **DiagramJSON**과 **TextEditor**가 더해집니다. 메서드는 **ZoomToFit**, **ZoomToSelection**, **CenterOn**, **ScreenToDiagram**, **DiagramToScreen**, **Undo**, **Redo**, **CopyToClipboard**, **CutToClipboard**, **PasteFromClipboard**, **DeleteSelection**, **EditText**, **SaveToPNG**, **SaveToSVG**이고, 이벤트는 **OnChange**, **OnSelectionChange**, **OnItemClick**, **OnItemDblClick**, **OnHover**, **OnConnecting**, **OnConnected**, **OnDrawNode**, **OnTextEditing**, **OnTextEdited**, **OnZoomChange**입니다. 팔레트에는 **Model**, **View**, **Items**, **IconSize**, **ShowCaptions**, **Theme**, **DragPayload**, **LoadDefaultItems**, **OnItemClick**이 있습니다.

모델과 엔진

esegece.sgcDiagram 패키지:

클래스	설명
TsgcDiagramDocument	다이어그램 자체입니다. **Nodes**, **Connectors**, **Layers**, **Images**, **Selection**, **NodeStyle**, **ConnectorStyle**, **Routing**, **Theme**, **GridSize**, **AddNode**, **AddConnector**, **BeginTransaction**, **EndTransaction**, **Undo**, **Redo**, **RouteAll**, **SaveToJSON**, **LoadFromJSON**, **SaveToFile**, **LoadFromFile**, **CopySelection**, **Paste**가 있습니다. 뷰는 자신의 문서를 만들고, 헤드리스 프로그램은 `new TsgcDiagramDocument()`로 문서를 만듭니다.
TsgcDiagramNode, TsgcDiagramConnector	항목이며 **Style**, **Layer**, **Data**를 가집니다. Delphi와 마찬가지로 노드는 **Ports**를, 커넥터는 **Labels**를 가집니다.
TsgcDiagramPort, TsgcDiagramLabel, TsgcDiagramLayer, TsgcDiagramImage, TsgcDiagramStyle	항목과 문서의 구성 요소입니다. 컬렉션은 **TsgcDiagramNodes**, **TsgcDiagramConnectors**, **TsgcDiagramPorts**, **TsgcDiagramLabels**, **TsgcDiagramLayers**이며 `Add()`, `Count`, 인덱서를 가집니다.
TsgcDiagramUndoStack	문서의 실행 취소 스택으로, **UndoStack**을 통해 접근합니다.
TsgcDiagramController	호스트가 공유하는 편집 로직입니다. 히트 테스트, 드래그 상태, 키 맵, 스냅, 가이드, 드롭을 처리합니다. 뷰는 이를 **Controller**로 노출합니다.

TsgcDiagramPaletteModel, TsgcDiagramPaletteItem	호스트가 없는 팔레트입니다. 항목, 범주, 레이아웃, 그리기, 히트 테스트, 드래그 페이로드를 처리합니다.
TsgcDiagramPainter, TsgcDiagramViewCache	문서의 페인터와 뷰의 캐시된 래스터입니다.
TsgcDiagramRenderer	헤드리스 내보내기입니다. **SaveToSVGStream**과 **SaveToSVGFile**은 esegece.sgcDiagram에 있고, **SaveToPNGStream**과 **SaveToPNGFile**은 esegece.sgcDiagram.Rendering의 확장 메서드입니다.
TsgcDiagramLayeredLayout, TsgcDiagramTreeLayout	자동 레이아웃입니다. Registered 에디션.
TsgcDiagramDataBinding	`DataTable` 로 다이어그램을 만듭니다. .NET 데이터 바인딩을 참조하십시오. Registered 에디션.
EsgcDiagramError	라이브러리의 예외입니다.

정적 클래스

Delphi 의 유닛 수준 상수와 함수는 유닛 이름을 딴 클래스의 정적 멤버입니다. **sgcDiagram_JSON** (**sgcDiagramSaveJSON**, **sgcDiagramLoadJSON**, **sgcDiagramCopy**, **sgcDiagramPaste**), **sgcDiagram_Render**, **sgcDiagram_Controller** (**CS_DIAGRAM_*** 상수와 **CS_DIAGRAM_KEY_*** 키 코드), **sgcDiagram_Palette**, **sgcDiagram_Types**가 있습니다. **sgcDiagram_Render_Rendering.RegisterRaster**는 Skia 래스터를 설치하며, 컨트롤은 생성자에서 이를 호출합니다. 팔레트의 드래그 형식은 **sgcDiagram_WinForms.CS_DIAGRAM_DRAG_FORMAT**, **sgcDiagram_Wpf.CS_DIAGRAM_DRAG_FORMAT**, **TsgcAvaloniaDiagramView.CS_DIAGRAM_AVALONIA_DRAG_FORMAT**이며 모두 `esegece.sgcDiagram.Shape`입니다.

열거형

열거형과 그 값은 Delphi 이름을 그대로 유지합니다. **TsgcDiagramShape** (**dgsRectangle**, **dgsDecision** 및 그 밖의 도형), **TsgcDiagramConnectorKind**, **TsgcDiagramArrow**, **TsgcDiagramTool**, **TsgcDiagramShowPorts**, **TsgcDiagramDirection**, **TsgcDiagramTreeStyle**, **TsgcDiagramBindingLayout**입니다. 값은 형식과 함께 씁니다 (예: `TsgcDiagramShape.dgsProcess`). Delphi 의 집합형은 `[Flags]` 열거형입니다. **TsgcDiagramChanges** 와 **TsgcDiagramShiftState**입니다.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- TsgcDiagramDocument
- TsgcDiagramController
- TsgcVCLDiagramView

Delphi와 .NET의 차이점

C# API는 Delphi API를 이름 하나하나 그대로 따릅니다. 달라지는 부분은 언어와 프레임워크에서 비롯됩니다.

Delphi	.NET
이벤트는 메서드 포인터이며 Sender는 컴포넌트입니다.	WinForms와 WPF 컨트롤에는 C# 이벤트(`+=`)가 있으며 Sender는 해당 컨트롤입니다. Avalonia 컨트롤, 문서, 컨트롤러, 팔레트 모델, 데이터 바인딩에는 델리게이트 속성(`=`)이 있습니다.
이벤트의 `var` 매개변수 (**OnTextEditing**, **OnConnecting**, **OnDrawNode**).	`ref` 매개변수입니다. `ref` 매개변수가 있는 람다는 모든 매개변수의 형식을 명시합니다.
인덱스 속성과 함수로 읽는 값입니다.	메서드: 컨트롤러의 **ContentBounds()**, **UndoCaption()**, **CanUndo()**, **Viewport()**, 팔레트 모델의 **GetCollapsed**, **SetCollapsed**, **GetCategory**, **MaxScroll()**, **Payload(i)**입니다. 컬렉션 항목은 `Add()` 로 추가합니다.
TList 인수입니다.	`List<TsgcDiagramItem>`, `List<TsgcDiagramNode>` and `List<TsgcDiagramConnector>`.
UndoStack은 **TObject**입니다.	여전히 `object` 이므로 **TsgcDiagramUndoStack**으로 캐스팅하십시오.
Free입니다.	가비지 컬렉터가 처리합니다. WPF와 Avalonia 뷰는 `IDisposable` 이므로 창을 닫을 때 **Dispose**를 호출하십시오. WinForms 뷰는 폼과 함께 해제됩니다. **TsgcDiagramController**와 **TsgcDiagramDataBinding**은 `IDisposable` 입니다.
DFM 또는 FMX 파일에 스트리밍되는 다이어그램입니다.	숨겨진 속성 **DiagramJSON**이며, WinForms 디자이너는 다이어그램이 비어 있지 않을 때만 이를 기록하고, XAML 특성으로도 사용할 수 있습니다.
키 코드와 **TShiftState**입니다.	`int` 가상 키 코드(**CS_DIAGRAM_KEY_***; WinForms의 **Keys** 값과 같습니다)와 `[Flags]` 열거형 **TsgcDiagramShiftState**입니다.
색상은 **TsgcUIColor**입니다.	`uint` ARGB 값입니다(예: `0xFFDBEAFE`). 0은 테마 색상입니다.
TsgcDiagramRenderer의 PNG 내보내기입니다.	esegece.sgcDiagram.Rendering의 확장 메서드이며 Skia로 그립니다. PNG 파일은 Delphi의 파일과 바이트 단위로 같지 않지만, SVG와 JSON 텍스트는 같습니다.
TsgcDiagramDataBinding은 **TDataSource**를 가진 컴포넌트입니다.	`DataTable` 을 가진 일반 클래스입니다. .NET 데이터 바인딩을 참조하십시오.

예제

`ref` 매개변수가 있는 이벤트, 플래그 집합, 그리고 Delphi 속성을 대체하는 메서드입니다.

```
using System;
using esegece.sgcDiagram;

public static class EventsSample
{
    public static void Wire(TsgcWinFormsDiagramView aView, Action<string> aLog)
    {
        // WinForms and WPF: C# events, Sender is the view
        aView.OnChange += (Sender, aChanges) =>
```

```
        {
            if ((aChanges & TsgcDiagramChanges.dgkStructure) != 0)
                aLog("structure changed");
        };
        aView.OnItemClick += (Sender, aItem) => aLog("click " + aItem.Id);
        // a Delphi var parameter is a ref parameter
        aView.OnTextEditing += (object Sender, TsgcDiagramNode aNode, ref bool aAllow) =>
            aAllow = aNode.Shape != TsgcDiagramShape.dgsTerminator;
        aView.OnConnecting += (object Sender, TsgcDiagramNode aSource, string aSourcePort,
            TsgcDiagramNode aTarget, string aTargetPort, ref bool aAccept) =>
            aAccept = aTarget != aSource;
    }

    public static string Describe(TsgcWinFormsDiagramView aView)
    {
        // Delphi properties without a setter that compute a value are methods
        TsgcDiagramRect vBounds = aView.Diagram.ContentBounds();
        string vUndo = aView.Diagram.CanUndo() ? aView.Diagram.UndoCaption() : "";
        // UndoStack is an object, cast it to reach the stack itself
        TsgcDiagramUndoStack oStack = (TsgcDiagramUndoStack)aView.Diagram.UndoStack;
        return string.Format("{0} x {1}, undo \"{2}\", {3} steps",
            vBounds.Right - vBounds.Left, vBounds.Bottom - vBounds.Top,
            vUndo,
            oStack.UndoCount);
    }
}
```

동일한 동작

- 월드 단위, 아래로 향하는 y축, 시계 방향 각도(도 단위), 그리드, 스냅, 히트 순서입니다.
- 실행 취소 캡션, 트랜잭션당 한 번의 실행 취소 단계, 애플리케이션 내부 클립보드 형식, 팔레트 드래그 페이로드입니다.
- JSON 형식, 경로, 레이아웃, SVG 내보내기는 Delphi와 같은 숫자와 텍스트를 사용합니다.

참고 항목

- .NET 개요
- .NET 클래스
- 이벤트
- JSON 형식

.NET 데이터 바인딩

Delphi의 **TsgcDiagramDataBinding**은 TDataSet으로 다이어그램을 만들며, 레코드마다 노드 하나와 각 레코드에서 부모 레코드로 이어지는 커넥터 하나를 만듭니다. .NET에서는 같은 클래스가 `System.Data.DataTable`(임의의 ADO.NET 공급자의 `DataAdapter`가 채우는 테이블) 또는 sgcUICore의 **IsgcUITable**을 읽습니다. Registered 에디션 기능입니다.

속성	설명
DataSource	레코드를 담은 `DataTable` 입니다.
Table	**IsgcUITable**입니다. 지정되어 있으면 **DataSource**보다 먼저 읽습니다.
KeyField, ParentField, ParentSeparator, LabelField	키 열, 부모 키 열(여러 부모는 **ParentSeparator**로 구분하며 기본값은 세미콜론), 노드 텍스트 열입니다.
Target, Document	**Target**은 다이어그램 뷰이거나 **IsgcDiagramDocumentOwner**를 구현하는 임의의 객체입니다. **Document**는 직접 쓰는 문서로, 뷰가 없는 프로그램에 사용합니다.
Layout, Direction, TreeStyle	다시 만든 다이어그램의 레이아웃(**dgbAuto**, **dgbTree**, **dgbLayered**, **dgbNone**)과 그 옵션입니다.
NodeShape, NodeWidth, NodeHeight, ConnectorKind	노드의 도형과 크기, 커넥터의 종류입니다.
LiveUpdates, Active	**LiveUpdates**가 true인 동안에는 테이블이 변경된 뒤 다이어그램을 다시 만듭니다. **Active**를 설정하면 다이어그램을 만듭니다.
OnCreateNode, OnCreateConnector	생성된 노드와 커넥터마다 행 인덱스와 함께 발생합니다. **CurrentRow**는 해당 노드의 `DataRow`이며 **OnCreateNode**가 실행되는 동안 유효합니다.

예제

```
using System;
using System.Data;
using esegece.sgcDiagram;

public static class DataBindingSample
{
    public static DataTable CreateEmployees()
    {
        DataTable oTable = new DataTable("Employees");
        oTable.Columns.Add("EmpNo", typeof(int));
        oTable.Columns.Add("ManagerNo", typeof(int));
        oTable.Columns.Add("FullName", typeof(string));
        oTable.Columns.Add("Department", typeof(string));
        oTable.Rows.Add(1, DBNull.Value, "Ann Lee", "Board");
        oTable.Rows.Add(2, 1, "Bob Stone", "Sales");
        oTable.Rows.Add(3, 1, "Carla Diaz", "Engineering");
        oTable.Rows.Add(4, 3, "Dan Ito", "Engineering");
        return oTable;
    }

    // Registered edition: one node per row, one connector per parent link
    public static TsgcDiagramDataBinding Bind(TsgcDiagramDocument aDocument, DataTable aTable)
    {
        TsgcDiagramDataBinding oBinding = new TsgcDiagramDataBinding();
        oBinding.KeyField = "EmpNo";
        oBinding.ParentField = "ManagerNo";
```

```
        oBinding.LabelField = "FullName";
        oBinding.Layout = TsgcDiagramBindingLayout.dgbTree;
        oBinding.TreeStyle = TsgcDiagramTreeStyle.dgoCompactLeaves;
        oBinding.NodeWidth = 180;
        oBinding.OnCreateNode = (Sender, aNode, aRow) =>
        {
            // CurrentRow is the DataRow of the node while the event runs
            DataRow oRow = ((TsgcDiagramDataBinding)Sender).CurrentRow;
            if ((string)oRow["Department"] == "Engineering")
                aNode.Style.Fill = 0xFFDBEAFE;
        };
        oBinding.Document = aDocument; // or Target = a diagram view
        oBinding.DataSource = aTable;
        oBinding.Active = true;
        return oBinding;
    }

    // without a SynchronizationContext (a console or a service) a change of
    // the table waits until CheckQueued runs in the thread of the document
    public static void AddEmployee(TsgcDiagramDataBinding aBinding, DataTable aTable)
    {
        aTable.Rows.Add(5, 3, "Eva Hart", "Engineering");
        aBinding.CheckQueued();
    }
}
```

TDataSet과의 차이

- `DataTable`에는 커서가 없으므로 **OnCreateNode**는 데이터셋의 필드 대신 **CurrentRow**로 행을 읽습니다.
- 바인딩은 테이블 이벤트를 수신합니다. 행 추가, 변경, 삭제, 변경 거부, 테이블 비우기가 일어나면 다이어그램을 한 번 다시 만듭니다. `AcceptChanges`와 행 사이의 이동은 아무것도 다시 만들지 않습니다. 편집 중인 행(`BeginEdit`)은 해당 `EndEdit`가 실행될 때 사용됩니다. 삭제된 행과 분리된 행은 건너뜁니다.
- 다시 만들기는 Delphi의 **ForceQueue**처럼 큐에 게시됩니다. WinForms, WPF, Avalonia 애플리케이션에서는 바인딩을 만들 때의 현재 **SynchronizationContext**에 게시되므로 따로 호출할 필요가 없습니다. 콘솔 프로그램이나 서비스에는 컨텍스트가 없으므로, 문서의 스레드에서 Delphi의 **CheckSynchronize**에 해당하는 **CheckQueued**를 호출하거나 **Refresh**로 즉시 다시 만드십시오.
- 바인딩은 컴포넌트가 아니라 `IDisposable`이며, 해제된 테이블이나 대상은 스스로 바인딩을 해제합니다.

컨트롤 바인딩

컨트롤의 속성은 다이어그램이 단일 값이 아니라 모델 전체이므로 종속성 속성이나 스타일 속성이 아닌 일반 CLR 속성입니다. MVVM 애플리케이션에서는 **TsgcDiagramDocument** 또는 그 JSON을 뷰 모델에 보관하고 뷰의 **OnChange**와 **OnSelectionChange**에 반응하십시오.

참고 항목

- .NET 개요
- Delphi와 .NET의 차이점
- TsgcDiagramDataBinding
- 데이터 바인딩

.NET 헤드리스 출력

문서에는 창이 필요하지 않습니다. 서비스, 테스트, 보고서에서 다이어그램을 만들거나 불러와 Linux를 포함한 모든 플랫폼에서 JSON, SVG, PNG로 저장할 수 있습니다.

- **TsgcDiagramDocument**는 `new TsgcDiagramDocument()`로 만듭니다. 커넥터는 가장 바깥쪽 트랜잭션이나 업데이트가 끝날 때 라우팅됩니다.
- **TsgcDiagramRenderer**는 여백(기본값 20)과 배율을 적용해 다이어그램 전체를 저장합니다. **SaveToSVGFile**과 **SaveToSVGStream**은 esegece.sgcDiagram에서, **SaveToPNGFile**과 **SaveToPNGStream**은 esegece.sgcDiagram.Rendering에서 사용합니다.
- 문서의 **SaveToJSON**, **LoadFromJSON**, **SaveToFile**, **LoadFromFile**, **SaveToStream**, **LoadFromStream**은 Delphi 파일 형식을 읽고 씁니다.

예제

```
using System.IO;
using esegece.sgcDiagram;

public static class HeadlessSample
{
    // a flowchart built in code, no window and no UI framework
    public static TsgcDiagramDocument CreateFlowchart()
    {
        TsgcDiagramDocument oDocument = new TsgcDiagramDocument();
        oDocument.BeginTransaction("Create");
        try
        {
            TsgcDiagramNode oStart = oDocument.AddNode(TsgcDiagramShape.dgsTerminator, 40, 40, 140, 50, "New order");
            TsgcDiagramNode oCheck = oDocument.AddNode(TsgcDiagramShape.dgsDecision, 30, 140, 160, 90, "Paid?");
            TsgcDiagramNode oShip = oDocument.AddNode(TsgcDiagramShape.dgsProcess, 40, 290, 140, 60, "Ship");
            TsgcDiagramNode oRemind = oDocument.AddNode(TsgcDiagramShape.dgsProcess, 260, 155, 140, 60, "Send reminder");
            oDocument.AddConnector(oStart, oCheck);
            oDocument.AddConnector(oCheck, oShip).Labels.Add().Text = "Yes";
            oDocument.AddConnector(oCheck, oRemind).Labels.Add().Text = "No";
        }
        finally
        {
            oDocument.EndTransaction(); // the connectors are routed here
        }
        return oDocument;
    }

    public static void Export(TsgcDiagramDocument aDocument, string aFolder)
    {
        TsgcDiagramRenderer oRenderer = new TsgcDiagramRenderer();
        // PNG needs esegece.sgcDiagram.Rendering, SVG and JSON only esegece.sgcDiagram
        oRenderer.SaveToPNGFile(aDocument, Path.Combine(aFolder, "order.png"), 2);
        oRenderer.SaveToSVGFile(aDocument, Path.Combine(aFolder, "order.svg"));
        aDocument.SaveToFile(Path.Combine(aFolder, "order.json"));
    }

    public static TsgcDiagramDocument Load(string aFileName)
    {
        TsgcDiagramDocument oDocument = new TsgcDiagramDocument();
        oDocument.LoadFromFile(aFileName);
        return oDocument;
    }
}
```

자동 레이아웃

Registered 에디션에서는 코드로 만든 다이어그램을 저장하기 전에 배치할 수 있습니다. **Execute**는 실행 취소 한 단계이며 커넥터를 다시 라우팅합니다.

```
using esegece.sgcDiagram;

public static class LayoutSample
{
    // Registered edition: arrange the nodes in layers, one undo step
    public static int ArrangeLayered(TsgcDiagramDocument aDocument)
    {
        TsgcDiagramLayeredLayout oLayout = new TsgcDiagramLayeredLayout(aDocument);
        oLayout.Direction = TsgcDiagramDirection.dglTopToBottom;
        oLayout.LayerSpacing = 60;
        oLayout.NodeSpacing = 30;
        oLayout.Execute();
        return oLayout.Crossings;
    }

    // Registered edition: a tree from the roots, leaves stacked under their parent
    public static void ArrangeTree(TsgcDiagramDocument aDocument)
    {
        TsgcDiagramTreeLayout oLayout = new TsgcDiagramTreeLayout(aDocument);
        oLayout.Direction = TsgcDiagramDirection.dglLeftToRight;
        oLayout.TreeStyle = TsgcDiagramTreeStyle.dgoCompactLeaves;
        oLayout.Execute();
    }
}
```

참고

- 이미지 크기는 콘텐츠 경계에 여백을 더한 값에 배율을 곱한 것입니다. 배율이 2이면 너비와 높이가 각각 두 배인 이미지가 저장됩니다.
- 문서의 **Theme**은 내보내기에 사용할 밝은 색 또는 어두운 색을 선택합니다.
- 렌더러의 **Painter.SetFontFile**은 기본 글꼴이 없는 서버를 위해 일반 및 굵은 TrueType 글꼴 파일을 받습니다.
- 컨트롤에도 **SaveToPNG**와 **SaveToSVG**가 있습니다.

참고 항목

- .NET 개요
- .NET 클래스
- 빠른 시작 헤드리스
- TsgcDiagramRenderer
- 계층형 레이아웃
- 트리 레이아웃

.NET 데모

데모는 `.net\demos\Diagram`에 있으며, 데모마다 프로젝트, `Program.cs`, README.md가 들어 있는 폴더가 하나씩 있습니다. 사용자 인터페이스는 XAML 없이 코드로 만듭니다. 모든 데모에는 화면에 표시하지 않고 자체 검사를 실행하여 PASS와 FAIL 줄을 출력하고 실패한 검사 수로 종료하는 `--test` 스위치와, 다이어그램을 PNG 파일로 저장하는 `--snapshot` 스위치가 있습니다.

데모	호스트	보여 주는 내용
FlowchartEditor	WinForms	팔레트와 뷰를 갖춘 Delphi FlowchartEditor 데모의 .NET 쌍둥이입니다. 팔레트에서 도형을 끌어 놓고, 커넥터 도구나 포트로 연결하며, 실행 취소와 다시 실행, 확대/축소, 테마, 계층형 레이아웃, JSON, PNG, SVG 파일을 다룰 수 있습니다.
FlowchartWpf	WPF	**TsgcWpfDiagramPalette**와 **TsgcWpfDiagramView**를 사용한 같은 편집기입니다.
FlowchartAvalonia	Avalonia	**TsgcAvaloniaDiagramPalette**와 **TsgcAvaloniaDiagramView**를 사용한 같은 편집기로, Windows, Linux, macOS에서 동작하며 핀치 확대와 한 손가락 이동을 지원합니다.
OrgChart	WinForms	`DataGridView`에 직원 60명의 `DataTable`을 표시하고, **TsgcDiagramDataBinding**을 통해 읽기 전용 뷰에 트리 레이아웃의 조직도를 표시합니다. 그리드에서 편집하면 조직도가 갱신됩니다. Registered 에디션.
NodeEditor	WinForms	노드 그래프 편집기입니다. 형식이 지정된 입력 포트와 출력 포트를 가진 블록이며, 형식이 일치할 때만 연결됩니다.
ConsoleRender	콘솔	창이 없습니다. 지원 티켓 순서도를 코드로 만들고 계층형 레이아웃으로 배치한 뒤, Windows와 Linux에서 flowchart.png, flowchart.svg, flowchart.json을 저장합니다. Delphi Headless 데모의 .NET 쌍둥이입니다.

세 Flowchart 데모의 UI와 무관한 부분(샘플 순서도, 도구 모음 동작, 자체 검사)은 `FlowchartEditor_Shared.cs`에 있으며, WinForms, WPF, Avalonia 프로젝트가 이를 컴파일하므로 세 호스트에서 같은 검사가 실행됩니다.
Community 에디션에서 Flowchart 데모는 샘플의 대략적인 위치를 유지하고 레이아웃 버튼을 숨기며, ConsoleRender는 행을 코드로 배치하고, OrgChart는 테이블을 표시하고 차트에 Registered 에디션이 필요하다고 알려 줍니다.

데모 빌드

제품 트리 안에서 데모는 라이브러리 프로젝트에 대해 컴파일됩니다. 데모 zip은 기본적으로 Community 에디션의 NuGet 패키지를 사용합니다(OrgChart는 Registered 패키지를 사용합니다). **SgcDiagramEdition**을 설정하면 Registered 패키지로 데모를 빌드할 수 있습니다.

```
dotnet build FlowchartEditor\FlowchartEditor.csproj -p:SgcDiagramEdition=Registered
dotnet run --project ConsoleRender\ConsoleRender.csproj -- --test
```

참고 항목

- .NET 개요
- NuGet 설치
- 빠른 시작 WinForms
- .NET 헤드리스 출력

문제 해결

문제	원인과 해결 방법
코드로 만든 커넥터에 경로가 없거나, 경로가 이동한 노드를 따라가지 않습니다.	경로는 가장 바깥쪽 BeginUpdate와 EndUpdate 쌍이 끝날 때, 그리고 Routing.AutoRoute가 True인 동안에만 계산됩니다. 모든 BeginUpdate를 닫거나, AutoRoute를 끈 상태에서 한 대량 변경 후에 RouteAll을 호출하십시오.
직교 커넥터가 노드를 가로지릅니다.	Community 에디션에서 dgcOrthogonal 커넥터는 장애물 회피가 없는 단순한 엘보입니다. Registered 에디션은 노드를 피해 라우팅합니다.
Layers.Add가 EsgcDiagramError를 발생시킵니다.	Community 에디션에서는 레이어를 하나만 사용할 수 있습니다. 레이어를 참조하십시오.
이미지 노드에 X 표시가 있는 상자가 나타납니다.	이미지 키가 Images에 없거나, 데이터가 PNG가 아니고 어떤 OnDecodeImage 핸들러도 디코딩하지 않았습니다.
이미지를 삭제하면 EsgcDiagramError가 발생합니다.	노드가 아직 ImageKey로 사용하고 있습니다. 먼저 해당 노드를 변경하거나 삭제하십시오.
선택한 노드의 변에서 끌면 커넥터를 그리지 않고 크기가 바뀝니다.	선택한 노드의 크기 조정 핸들이 변의 포트를 덮고 있습니다. 먼저 빈 공간을 클릭하거나 Tool을 dgtConnector로 설정하십시오.
텍스트 편집기가 열리지 않습니다.	뷰가 ReadOnly이거나, 노드 또는 그 레이어가 Locked이거나, OnTextEditing 핸들러가 aAllow를 False로 설정했습니다.
다른 애플리케이션에서 복사한 후 붙여넣어도 아무 일도 일어나지 않습니다.	sgcDiagram의 클립보드는 애플리케이션 내부용입니다. 다른 프로그램과 데이터를 교환하지 않습니다.
코드로 변경한 후 실행 취소해도 아무 일도 일어나지 않습니다.	BeginTransaction과 EndTransaction 밖에서 한 변경은 실행 취소 및 다시 실행 스택을 지웁니다. 코드로 한 변경을 실행 취소할 수 있게 하려면 트랜잭션으로 감싸십시오.
iOS 또는 Android에서는 팔레트에서 도형을 끌 수 없습니다.	모바일 플랫폼에는 드래그 서비스가 없습니다. 팔레트 항목을 탭한 다음 뷰를 탭하십시오.
FireMonkey 뷰에서 Tab을 눌러도 항목이 순환하지 않습니다.	FireMonkey 폼은 포커스된 컨트롤보다 먼저 Tab을 처리하고 포커스를 이동합니다.
항목의 Id를 설정하면 EsgcDiagramError가 발생합니다.	Id는 문서 안에서 고유합니다. 다른 노드나 커넥터가 사용하지 않는 Id를 선택하십시오.
레이아웃 및 데이터 바인딩 클래스가 없습니다.	이들은 Registered 에디션의 일부입니다. Community 에디션에서는 sgcDiagram_Layout과 sgcDiagram_Data가 빈 유닛입니다.

관련 항목

- 에디션
- 플랫폼

Index

Copyright © 2012-2026 eSeGeCe Software
info@esegece.com
www.esegece.com