Table of Contents

Part I Introduction 1

1 Technical Support 1
2 Trial Limitations 3
3 Web Links and Online Resources 3

Part II Deployment 5

1 Libraries of Report.WEB Package 6
2 Libraries of Reports.NET Package 7
3 Scripts of Reports.JS Package 8
4 Libraries of Reports.WPF Package 9
5 Scripts of Reports.PHP Package 11
6 JAR Files of Reports.JAVA Package 12
7 Libraries of Dashboards.WEB Package 13
8 Libraries of Dashboards.WIN Package 14
9 Scripts of Dashboards.JS Package 15
10 Scripts of Dashboards.PHP Package 16
11 Files of Stimulsoft Designer Application 18
12 Files of Stimulsoft Designer.JS Application 22
13 Files of Stimulsoft Demo Application 23

Part III Reports and Dashboards for ASP.NET WebForms 26

1 HTML5 Viewer 26

How this Works 28
Activation 29
Showing Reports and Dashboards 29
Connecting Data 33
Localization 40
Using Themes 40
Basic Features 43
Printing Reports 43
Exporting Reports and Dashboards 46
Viewing Modes 50
Work with Parameters 53
Work with Bookmarks 55
Dynamic Sorting, Collapsing, and Drill-Down 57
Editing Report 61
Sending Report by Email 62
Calling Designer from Viewer 65
Caching 65
Export and Printing from Code 68
Viewer Events 70

© 2003-2024 Stimulsoft

Timeout ... 71
Viewer Settings ... 73
2 HTML5 Designer ... 90
How this Works ... 92
Activation ... 93
Editing Reports and Dashboards ... 94
Creating New Reports and New Dashboards ... 96
Preview ... 98
Additional Features of Preview ... 100
Saving Reports and Dashboards ... 101
Localization ... 105
Using Themes ... 107
Caching ... 108
Designer Events ... 111
Timeout ... 113
Add custom functions ... 115
Settings ... 115
Part IV Reports and Dashboards for ASP.NET MVC 135
1 HTML5 Viewer ... 135
How this Works ... 138
Activation ... 138
Showing Reports and Dashboards ... 139
Connecting Data ... 142
Localization ... 149
Using Themes ... 150
Basic Features ... 153
Printing Reports ... 154
Exporting Reports and Dashboards ... 156
Viewing Modes ... 160
Work with Parameters ... 163
Work with Bookmarks ... 166
Dynamic Sorting, Collapsing, and Drill-Down ... 168
Editing Report ... 172
Sending Report by Email ... 173
Calling Designer from Viewer ... 176
Caching ... 177
Additional Methods ... 180
Export and Printing from Code ... 183
Timeout ... 185
Viewer Settings ... 186
2 HTML5 Designer ... 205
How this Works ... 207
Activation ... 207
Editing Reports and Dashboards ... 208
Creating New Reports and New Dashboards ... 211
Preview ... 212
Additional Features of Preview ... 215
Saving Reports and Dashboards ... 216
Localization ... 220
Using Themes ... 221
Caching ... 222
Additional Methods ... 225

© 2003-2024 Stimulsoft

Timeout 228
Add custom functions 230
Settings 231

Part V Reports and Dashboards for ASP.NET Core MVC 252

1 HTML5 Viewer 252
How this Works 254
Activation 255
Showing Reports and Dashboards 256
Connecting Data 259
Localization 266
Using Themes 267
Basic Features 270
Printing Reports 271
Exporting Reports and Dashboards 273
Viewing Modes 277
Work with Parameters 280
Work with Bookmarks 283
Dynamic Sorting, Collapsing, and Drill-Down 285
Editing Report 288
Sending Report by Email 289
Calling Designer from Viewer 292
Caching 293
Additional Methods 296
Export and Printing from Code 300
Timeout 301
Viewer Settings 303

2 HTML5 Designer 321
How this Works 323
Activation 324
Editing Reports and Dashboards 325
Creating New Reports and New Dashboards 327
Preview 328
Additional Features of Preview 331
Saving Reports and Dashboards 332
Localization 336
Using Themes 337
Caching 338
Additional Methods 341
Timeout 345
Add custom functions 347
Settings 348

Part VI Reports and Dashboards for ASP.NET Core Razor 369

1 HTML5 Viewer 370
How this Works 372
Activation 372
Showing Reports and Dashboards 373
Connecting Data 376
Localization 383

© 2003-2024 Stimulsoft

Using Themes ... 384
Basic Features ... 387
Printing Reports ... 388
Exporting Reports and Dashboards ... 391
Viewing Modes ... 395
Work with Parameters ... 398
Work with Bookmarks ... 401
Dynamic Sorting, Collapsing, and Drill-Down ... 403
Editing Report ... 406
Sending Report by Email ... 407
Calling Designer from Viewer ... 410
Caching ... 411
Additional Methods ... 413
Export and Printing from Code ... 417
Timeout ... 418
Viewer Settings ... 420

2 HTML5 Designer ... 439

How this Works ... 440
Activation ... 441
Editing Reports and Dashboards ... 441
Creating New Reports and New Dashboards ... 444
Preview ... 445
Additional Features of Preview ... 448
Saving Reports and Dashboards ... 449
Localization ... 453
Using Themes ... 454
Caching ... 455
Additional Methods ... 458
Timeout ... 461
Add custom functions ... 463
Settings ... 464

Part VII Reports and Dashboards for Blazor 485

1 Viewer ... 485

Activation ... 487
Showing Reports ... 488
Connecting Data ... 490
Localization ... 497
Using Themes ... 497
Basic Features ... 500
Printing Reports ... 501
Exporting Reports ... 503
Vieweing Modes ... 507
Work with Parameters ... 511
Work with Bookmarks ... 514
Dynamic Sorting, Collapsing, and Drill-Down ... 517
Editing Report ... 521
Sending Report by Email ... 522
Calling Designer from Viewer ... 525
Export and Printing from Code ... 526
Viewer Events ... 527
Viewer Settings ... 528

2 Designer ... 543

© 2003-2024 Stimulsoft

Activation ... 545
Editing Reports ... 545
Creating New Reports ... 547
Preview ... 549
Additional Features of Preview ... 550
Saving Reports ... 551
Localization ... 555
Using Themes ... 555
Designer Events ... 556
Settings ... 558

Part VIII Reports for Angular 576

1 Get Started ... 576
2 Activation ... 581
3 Angular Viewer ... 581
How this Works ... 583
Showing Reports ... 583
Connecting Data ... 586
Localization ... 592
Using Themes ... 592
Basic Features ... 596
Printing Reports ... 597
Exporting Reports ... 599
Viewing Modes ... 603
Work with Parameters ... 606
Work with Bookmarks ... 609
Dynamic Sorting, Collapsing, and Drill-Down ... 611
Editing Report ... 615
Sending Report by Email ... 616
Calling Designer from Viewer ... 619
Caching ... 619
Additional Methods ... 622
Timeout ... 626
Viewer Settings ... 628
API References ... 646

Part IX Reports and Dashboards for JS 650

1 Get Started for Browser ... 650
2 Get Started for Node.js ... 653
3 HTML5 Viewer ... 655
How this Works ... 657
Showing Reports ... 658
Using Themes ... 661
Printing Reports ... 664
Exporting Reports ... 666
Viewing Modes ... 669
Work with Parameters ... 672
Work with Bookmarks ... 673
Dynamic Sorting, Collapsing, and Drill-Down ... 675
Editing Report ... 679
Sending Report by Email ... 680

© 2003-2024 Stimulsoft

Calling Designer from Viewer ... 682
Viewer Events ... 683
Viewer Settings ... 697
4 HTML5 Designer ... 711
How this Works ... 713
Editing Reports ... 713
Creating New Report ... 718
Preview ... 720
Additional Features of Preview ... 722
Saving Reports ... 722
Using Themes ... 725
Designer Events ... 726
Customizations in Designer ... 736
Designer Settings ... 738
5 Engine ... 757
Activation ... 757
Connecting Data Files ... 758
Connecting SQL Databases ... 760
Localization ... 767
Loading and Saving Report ... 768
Saving Rendered Report ... 769
Getting Access to Pages ... 769
Report Events ... 770
Add custom functions ... 777
Part X Reports and Dashboards for PHP 777
1 Engine ... 778
Deployment ... 778
Optimization of Scripts Loading ... 782
Activation ... 785
Loading and Saving Reports ... 787
Rendering a Report ... 794
PHP Events Handler ... 796
Connecting Data Files ... 799
Connecting SQL Data Adapters ... 804
Work with Report Variables ... 813
Printing Report from Code ... 818
Export Report from Code ... 820
Engine Events ... 822
2 HTML5 Viewer ... 830
Deployment ... 831
Activation ... 835
Showing Reports and Dashboards ... 837
Localization ... 838
Printing Report ... 838
Report Export ... 842
Viewing Modes ... 847
Work with Report Variables ... 850
Work with Bookmarks ... 852
Dynamic Sorting, Collapsing, and Drill-Down ... 853
Editing Report ... 855
Sending Report By Email ... 856

© 2003-2024 Stimulsoft

Calling Designer from Viewer ... 859
Appearance ... 860
Viewer Events ... 861
Viewer Settings ... 876
3 HTML5 Designer ... 891
Deployment ... 892
Activation ... 896
Creating and Editing Reports ... 898
Saving Reports ... 900
Localization ... 904
Preview ... 905
Appearance ... 907
Designer Events ... 907
Designer Settings ... 913
Part XI Reports and Dashboards for WinForms 932
1 Activation ... 933
2 WinForms Viewer ... 934
How to Show Report ... 934
Dot-Matrix Mode of WinForms Viewer ... 935
3 Add custom functions ... 938
Part XII Reports and Dashboards for Python 939
1 Report Engine ... 939
Deployment ... 940
Optimizing Scripts Loading ... 945
License Activation ... 948
Report Rendering ... 949
Printing a Report From Code ... 951
Exporting a Report From Code ... 951
Part XIII Reports.Java 954
1 Activation ... 954
2 Java Viewer ... 955
Showing Reports ... 955
Custom Functions ... 956
3 Java HTML5 Viewer ... 958
Installation ... 958
Creating Project ... 958
Creating a Sample Page ... 962
Create a Sample Page With Report HTML5 Viewer ... 965
Description of Webviewer Tag ... 968
Options ... 969
Template JDBC Connections ... 973
4 HTML5 Designer ... 974
Installation and Description HTML5 Designer ... 974
Template JDBC Coonections ... 978
Part XIV Reports.WPF 980

© 2003-2024 Stimulsoft

1 Activation 980
2 Wpf Viewer 981
How to Show Report 981
Dot-Matrix Mode of Wpf Viewer 982

Part XV Engine 986

1 Data 986
Business Objects in Net, Web 986
Working with OData Using Business Objects 996
2 Report Inheritance 996
Basic Approaches 997
3 Right To Left 997
WinForms Viewer 998
WPF Designer and Viewer 998
Icons 1000
4 Exports 1006
Export Reports From Code 1009
ExportDocument Method 1010
Export Service 1011
Formats with Fixed Page Layout 1013
PDF 1013
ZUGFeRD 1023
Special Features of PDF/A 1025
Microsoft Power Point 2007/2010 1027
XPS 1028
Web Documents 1029
HTML 1029
MHT 1032
Text Formats 1033
TXT 1033
RTF 1036
Word 2007/2010 1038
ODT 1040
Spreadsheets 1042
Excel 1042
Excel 2007/2010 1044
ODS 1046
Data 1048
CSV 1048
DBF 1049
XML 1051
DIF 1053
SYLK 1054
Images 1054
5 Scripts 1056
Programming Language of Report 1056
Report Code 1057

Part XVI Forms.WEB 1059

1 Get Started for ASP.NET Core 3.1 1059

© 2003-2024 Stimulsoft

Index 0

© 2003-2024 Stimulsoft

1 Introduction

We are glad to welcome you to the online version of the documentation of **Stimulsoft Reports** products. The documentation describes the basics of using the **API** of our software. Here we will review how to pass data from code in a report, export reports to various file formats, inherit reports, work with the components and more:

Welcome to **Stimulsoft**:

- Technical Support
- Trial Limitations
- Web Links and Online Resources
- Product Evaluation
- Engine
- Reports and Dashboards for ASP.NET WebForms
- Reports and Dashboards for ASP.NET MVC
- Reports and Dashboards for ASP.NET Core MVC
- Reports and Dashboards for ASP.NET Core Razor
- Reports and Dashboards for Blazor
- Reports and Dashboards for WinForms
- Reports for Angular
- Reports and Dashboards for JS
- Reports and Dashboards for PHP
- Reports and Dashboards for Python
- Reports for Java
- Reports for WPF
- Forms for Web

The first part of the documentation contains the description of work with visual parts of **Stimulsoft** products.

1.1 Technical Support

Registered users and users who are evaluating the software may get technical support.

© 2003-2024 Stimulsoft

For technical questions, use the Email address: support@stimulsoft.com
For subscription, payment questions, use Email address: sales@stimulsoft.com
For other questions, use Email address: info@stimulsoft.com
If you have problems with our products, you may contact us through our feedback form at https://www.stimulsoft.com/en/support

If you are a registered user and you contact us for technical support, use the same Email address you used when you purchased our product. Otherwise, it will be difficult to identify you as a registered user. This can slow down our response. Please let us know when your Email address changes.

To solve your problem quickly, we need the following information:

- Product name and its version;
- A detailed description of the problem and how to reproduce it;
- Your operating system (98, ME, 2000, XP, Vista, Window 7 etc.), its version, and the localization of established service packs;

© 2003-2024 Stimulsoft

- Version of Microsoft .NET Framework or other development environment and installed service packs;
- A name of your development environment and its version;
- Additional information that can help us solve the problem.

1.2 Trial Limitations

Trial Versions

The free trial of Stimulsoft software is a full-featured version. It has a few limitations, which are as follows:

- The evaluation period is limited to 60 days for Stimulsoft components;
- The Trial watermark is printed on each report page or the dashboard panel.

Registered Versions

If your reports are displaying a **TRIAL** watermark, this means that you are using a trial version of the product. Log in to your account and activate the product using the license key.

Check out how to activate

- ASP.NET HTML5 Viewer
- ASP.NET HTML5 Designer
- ASP.NET MVC HTML5 Viewer
- ASP.NET MVC HTML5 Designer
- ASP.NET Core HTML5 Viewer
- ASP.NET Core HTML5 Designer
- Angular Components
- JavaScript Components
- WinForms Components
- Java Components
- WPF Components

1.3 Web Links and Online Resources

This section describes how to get information about the latest news and announcements of software products, and information about known issues and questions that users are interested in.

- The official website of the company is available at https://stimulsoft.com
  - Description of products you may find at https://stimulsoft.com/en/products
  - Download the latest version of the product by the link https://stimulsoft.com/en/downloads
  - You can read the latest news of the company at https://stimulsoft.com/en/blog/news

- In addition, you can download packages of products Stimulsoft from other resources:
  - Reports.Web, Reports.Blazor, Reports.Angular Reports.Net, Reports.Wpf,

© 2003-2024 Stimulsoft

Reports.Web.NetCore, Dashboards.Blazor, Dashboards.Win, Dashboards.Web, Dashboards.Web.NetCore from NuGet at https://www.nuget.org/profiles/Stimulsoft

- Reports.Java from Maven at http://central.maven.org/maven2/com/stimulsoft
- Reports.JS and Dashboards.JS from npm at https://www.npmjs.com/search?q=stimulsoft
- Reports.PHP and Dashboards.PHP from composer at https://packagist.org/?query=stimulsoft

› You can evaluate our reporting and dashboard tools online - https://demo.stimulsoft.com

› To create, store and then deploy reports in your applications, use the cloud service of Stimulsoft https://cloud.stimulsoft.com

› A huge number of video lessons are available on our YouTube channel https://www.youtube.com/user/StimulsoftVideos

› You can use samples for various platforms:

- on GitHub at https://github.com/stimulsoft
- on our website at https://www.stimulsoft.com/en/samples

› Find us in social networks and messengers:

https://twitter.com/stimulsoft

https://www.linkedin.com/company/stimulsoft

https://www.facebook.com/Stimulsoft

https://www.stimulsoft.com/en/rss

https://t.me/stimulsoft

+48690104472

WhatsApp - Stimulsoft

Skype - Stimulsoft

© 2003-2024 Stimulsoft

Also, visit our Forum to communicate with other users of Stimulsoft Reports - http://forum.stimulsoft.com/index.php
Here you can read and discuss various topics related to tools for creating reports. For more information about the product in other Internet resources, please use the search engines.

2 Deployment

Information

Stimulsoft Ultimate package contains libraries, scripts, and other files necessary for working with reports and dashboards in WinForms, ASP.NET, ASP.NET MVC, .NET Core, WPF, JavaScript, PHP, and Java applications. The product includes a report and data analysis engine, reports and dashboards designers and viewers for all supported platforms.

This section describes the various distribution options for reports, libraries, and files supplied with Stimulsoft software.

Name	Description
Reports.WEB	The package contains libraries for working with reports in ASP.NET, ASP.NET MVC, and .NET Core applications.
Reports.NET	The package contains libraries for working with reports in WinForms applications.
Reports.JS	The package contains scripts for working with reports in JavaScript applications.
Reports.WPF	The package contains libraries for working with reports in WPF applications.
Reports.PHP	The package contains scripts and styles for working with reports in client-server applications using PHP.
Reports.JAVA	The package contains JAR files for working with reports in Java applications.
Dashboards.WEB	The package contains libraries for working with dashboards and reports in ASP.NET, ASP.NET MVC, and .NET Core applications.

© 2003-2024 Stimulsoft

Dashboards.WIN	The package contains libraries for working with dashboards and reports in WinForms and WPF applications.
Dashboards.JS	The package contains scripts for working with dashboards and reports in JavaScript applications.
Stimulsoft Designer	The application contains files, libraries to run the standalone report designer.
Stimulsoft Designer.JS	The application contains files, scripts to run the standalone JavaScript report designer.
Stimulsoft Demo	The application contains files, libraries to run the reports and dashboards demonstration.

Information

A dmg is supplied in which **Stimulsoft Designer for Mac** is presented as a single application. After installation, run the **Stimulsoft Designer.app** file to start the report designer.

2.1 Libraries of Report.WEB Package

Reports.WEB package contains the following libraries:

Libraries	Description
Stimulsoft.Base.dll	The library contains common base interfaces and classes for all products.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Report.Check.dll	The library contains all methods for working with the Report Checker.
Stimulsoft.Report.Design.dll	The library contains resources and control classes of the report designer.

© 2003-2024 Stimulsoft

Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Mvc.dll	The library contains resources and control classes of the viewer and designer for ASP.NET MVC.
Stimulsoft.Report.Web.dll	The library contains resources and control classes of the viewer for ASP.NET.
Stimulsoft.Report.WebDesign.dll	The library contains resources and control classes of the designer for ASP.NET.
Stimulsoft.Report.Mvc.NetCore.dll	The library contains resources and control classes of the viewer and designer for .NET Core MVC.
Stimulsoft.System.dll	The library contains system interfaces and classes required for the report generator with .NET Standard compatibility.
Stimulsoft.System.Web.dll	The library contains system interfaces and classes required for Web controls with .NET Standard compatibility.
Svg	The library contains functionality for using SVG images in reports and dashboards.
LibExcel	The library is applicable for using XLS files as a data source in reports and dashboards.

2.2 Libraries of Reports.NET Package

Reports.NET package contains the following libraries:

Libraries	Description
Stimulsoft.Base.dll	The main library contains common base interfaces and classes for all products.
Stimulsoft.Controls.dll	The library contains additional controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Controls.Win.dll	The library contains main controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Datab	The library is used for creating and editing database

© 2003-2024 Stimulsoft

ase.dll	connections. The library contains the SQL Query Builder.
Stimulsoft.Design.dll	The library contains classes for working with reports in Design Time for Visual Studio.
Stimulsoft.Editor.dll	The library contains a text editor, which is used in the report designer.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The main library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Report.Check.dll	The library contains all methods for working with the Report Checker.
Stimulsoft.Report.Design.dll	The library contains resources and control classes of the WinForms report designer.
Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Web.dll	The library contains resources and control classes of the viewer for ASP.NET.
Stimulsoft.Report.Win.dll	The library contains resources and control classes of the WinForms viewer.
Svg	The library contains functionality for using SVG images in reports and dashboards.

2.3 Scripts of Reports.JS Package

Reports.JS package contains the following scripts:

Scripts	Description
Common Scripts *:	
stimulsoft.blockly.editor.js	The script contains visual editor Blockly for creating event scripts in a report. The Event handler is embedded in report engine.
stimulsoft.designer.js	The script for working with the report designer.

© 2003-2024 Stimulsoft

stimulsoft.reports.js	The main script contains all the functionality of the report generator - rendering, exporting, working with data.
stimulsoft.reports.maps.js	The script contains resources for working with region maps.
stimulsoft.viewer.js	The script for working with the report viewer.
stimulosft.reports.js can be split into *:	
stimulsoft.reports.engine.js	The main script of the report generator.
stimulsoft.reports.chart.js	The script is necessary for working with charts.
stimulsoft.reports.export.js	The script is necessary for exporting reports.
stimulsoft.reports.import.xlsx.js	The script is necessary for importing data from XLSX files.
Other Files:	
stimulsoft.reports.d.ts	The header file describes the syntax and structure of functions and properties.

* - Also, you can use packed scripts instead of common scripts or parts of the stimulsoft.reports.js. Packed scripts allow you to reduce the script file size by several times. The names of packed scripts correspond with the names of scripts, but they contain the word **pack** in the name before the **js** extension. For example, a packed analog of the **stimulsoft.reports.js** script will be the **stimulsoft.reports.pack.js**.

2.4 Libraries of Reports.WPF Package

Reports.WPF package contains the following libraries:

Libraries	Description
Stimulsoft.Base.dll	The main library contains common base interfaces and classes for all products.

© 2003-2024 Stimulsoft

Stimulsoft.Client.Designer.dll	The library contains resources and classes of the WPF report designer V2.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Database.dll	The library is used for creating and editing database connections. The library contains the SQL Query Builder.
Stimulsoft.Editor.dll	The library contains a text editor, which is used in the report designer.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The main library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Report.Check.dll	The library contains all methods for working with the Report Checker.
Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Wpf.dll	The library contains resources and classes of the WPF report viewer.
Stimulsoft.Report.Wpf.BlackTheme.dll	This library contains the Black theme using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2003BlueTheme.dll	This library contains the Office 2003 Blue theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2003OliveGreenTheme.dll	This library contains the Office 2003 Olive Green theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2003SilverTheme.dll	This library contains the Office 2003 Silver theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2007BlackTheme.dll	This library contains the Office 2007 Black theme for using it in the WPF viewer and report designer.

© 2003-2024 Stimulsoft

Stimulsoft.Report.Wpf.Office2007BlueTheme.dll	This library contains the Office 2007 Blue theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2007SilverTheme.dll	This library contains the Office 2007 Silver theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2010BlueTheme.dll	This library contains the Office 2010 Blue theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2010WhiteTheme.dll	This library contains the Office 2010 White theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.WpfDesign.dll	The library contains resources and classes of the WPF report designer V1.

2.5 Scripts of Reports.PHP Package

Reports.PHP package contains the following libraries:

Scripts	Description
stimulsoft.blockly.editor.js	The script contains visual editor Blockly for creating event scripts in a report. The Event handler is embedded in report engine.
stimulsoft.designer.js	The script for working with the report designer.
stimulsoft.designer.pack.js	The packed script for working with the report designer. It is similar to the previous one, but a bit smaller.
stimulsoft.reports.js	The main script contains all the functionality of the report generator - rendering, exporting, working with data.
stimulsoft.reports.pack.js	The packed main script contains all the functionality of the report generator - rendering, exporting, working with data. It is similar to the previous one, but a bit smaller.
stimulsoft.reports.maps.js	The script contains resources for working with region maps.

© 2003-2024 Stimulsoft

stimulsoft.reports.maps.pack.js	The packed script contains resources for working with region maps. It is similar to the previous one but a bit smaller.
stimulsoft.viewer.js	The script for working with the report viewer.
stimulsoft.viewer.pack.js	The packed script for working with the report viewer. It is similar to the previous one but a bit smaller.
stimulsoft.reports.d.ts	The header file describes the syntax and structure of functions and properties.

2.6 JAR Files of Reports.JAVA Package

Reports.JAVA package contains the following files:

JAR Files	Description
stimulsoft-reports-base	The file contains base interfaces and classes for the product.
stimulsoft-reports-demo	The file contains a demo application.
stimulsoft-reports-lib	The file contains a utility library.
stimulsoft-reports-report	The file contains all the functionality of the report generator - rendering, exporting, working with data.
stimulsoft-reports-samples	The file contains samples.
stimulsoft-reports-viewer	The file contains resources and control classes of the java viewer.
stimulsoft-reports-web	The file contains a common part of the web designer and viewer.
stimulsoft-reports-webdesigner	The file contains resources and control classes of the web designer.
stimulsoft-reports-webviewer	The file contains resources and control classes of the web viewer.

© 2003-2024 Stimulsoft

2.7 Libraries of Dashboards.WEB Package

Dashboards.WEB package contains the following libraries:

Libraries	Description
Stimulsoft.Base.dll	The main library contains common base interfaces and classes for all products.
Stimulsoft.Dashboard	The library contains the core functionality of the Stimulsoft Dashboards - main objectives, classes, definitions.
Stimulsoft.Dashboard.Drawing	The library is used for drawing the elements of Stimulsoft Dashboards with the help of GDI technology.
Stimulsoft.Dashboard.Export	The library contains all the necessary methods for exporting dashboards.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The main library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Report.Check.dll	The library contains all methods for working with the Report Checker.
Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Mvc.dll	The library contains resources and control classes of the viewer and designer for ASP.NET MVC.
Stimulsoft.Report.Web.dll	The library contains resources and control classes of the viewer for ASP.NET.
Stimulsoft.Report.WebDesign.dll	The library contains resources and control classes of the designer for ASP.NET.
Svg	The library contains functionality for using SVG images in reports and dashboards.
LibExcel	The library is required for using XLS files as a data source in reports and dashboards.

© 2003-2024 Stimulsoft

2.8 Libraries of Dashboards.WIN Package

Dashboards.WIN package contains the following libraries:

Libraries	Description
Stimulsoft.Base.dll	The main library contains common base interfaces and classes for all products.
Stimulsoft.Controls.dll	The library contains additional controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Controls.Win.dll	The library contains main controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Dashboard.dll	The library contains the core functionality of the Stimulsoft Dashboards - main objectives, classes, definitions.
Stimulsoft.Dashboard.Design.dll	The library contains resources and control classes of the WinForms dashboard designer.
Stimulsoft.Dashboard.Drawing.dll	The library is used for drawing the elements of Stimulsoft Dashboards with the help of GDI technology.
Stimulsoft.Dashboard.Drawing.Wpf.dll	The library is used for drawing the elements of Stimulsoft Dashboards with the help of WPF technology.
Stimulsoft.Dashboard.Export.dll	The library contains all the necessary methods for exporting dashboards.
Stimulsoft.Dashboard.Viewer.dll	The library contains resources and control classes of the WinForms dashboard viewer.
Stimulsoft.Dashboard.Viewer.Wpf.dll	The library contains resources and control classes of the WPF dashboard viewer.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Database.dll	The library is used for creating and editing database connections. The library contains the SQL Query Builder.
Stimulsoft.Design.dll	The library contains classes for working with reports in Design Time for Visual Studio.

© 2003-2024 Stimulsoft

Stimulsoft.Editor.dll	The library contains a text editor which is used in the report designer.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The main library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Report.Check.dll	The library contains all methods for working with the Report Checker.
Stimulsoft.Report.Design.dll	The library contains resources and control classes of the WinForms report designer.
Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Web.dll	The library contains resources and control classes of the ASP.NET viewer.
Stimulsoft.Report.Win.dll	The library contains resources and control classes of the WinForms viewer.
Stimulsoft.Report.Wpf.dll	The library contains resources and control classes of the WPF viewer.
Svg	The library contains functionality for using SVG images in reports and dashboards.

2.9 Scripts of Dashboards.JS Package

Dashboards.JS package contains the following scripts:

Scripts	Description
Common Scripts *:	
stimulsoft.blockly.editor.js	The script contains visual editor Blockly for creating event scripts in a report. The Event handler is embedded in report engine.
stimulsoft.dashboards.js	The script for working with dashboards.
stimulsoft.desig	The script for working with the report designer.

© 2003-2024 Stimulsoft

ner.js	
stimulsoft.reports.js	The main script contains all the functionality of the report generator - rendering, exporting, working with data.
stimulsoft.reports.maps.js	The script contains resources for working with region maps.
stimulsoft.viewer.js	The script for working with the report viewer.
stimulosft.reports.js can be split into *:	
stimulsoft.reports.engine.js	The main script of the report generator.
stimulsoft.reports.chart.js	The script is necessary for working with charts.
stimulsoft.reports.export.js	The script is necessary for exporting reports.
stimulsoft.reports.import.xlsx.js	The script is necessary for importing data from XLSX files.
Other Files:	
stimulsoft.reports.d.ts	The header file describes the syntax and structure of functions and properties.

* - Also, you can use packed scripts instead of common scripts or parts of the stimulsoft.reports.js. Packed scripts allow you to reduce script file size by several times. The names of packed scripts correspond with the names of scripts, but they contain a **pack** word in the name before **js** extension. For example, a packed analog of the **stimulsoft.reports.js** script will be the **stimulsoft.reports.pack.js**.

2.10 Scripts of Dashboards.PHP Package

Reports.PHP package contains the following libraries:

Scripts	Description
Common Scripts *:	

© 2003-2024 Stimulsoft

stimulsoft.blockly.editor.js	The script contains visual editor Blockly for creating event scripts in a report. The Event handler is embedded in report engine.
stimulsoft.dashboards.js	The script for working with dashboards.
stimulsoft.designer.js	The script for working with the report designer.
stimulsoft.reports.js	The main script contains all the functionality of the report generator - rendering, exporting, working with data.
stimulsoft.reports.maps.js	The script contains resources for working with region maps.
stimulsoft.viewer.js	The script for working with the report viewer.
stimulosft.reports.js can be split into *:	
stimulsoft.reports.engine.js	The main script of the report generator.
stimulsoft.reports.chart.js	The script is necessary for working with charts.
stimulsoft.reports.export.js	The script is necessary for exporting reports.
stimulsoft.reports.import.xlsx.js	The script is necessary for importing data from XLSX files.
Other Files:	
stimulsoft.reports.d.ts	The header file describes the syntax and structure of functions and properties.

* - Also, you can use packed scripts instead of common scripts or parts of the stimulsoft.reports.js. Packed scripts allow you to reduce script file size by several times. The names of packed scripts correspond with the names of scripts, but they contain a **pack** word in the name before **js** extension. For example, a packed analog of the **stimulsoft.reports.js** script will be the **stimulsoft.reports.pack.js**.

© 2003-2024 Stimulsoft

2.11 Files of Stimulsoft Designer Application

Stimulsoft Designer application package contains the following files:

Name	Description
Folders:	
Localization	The folder contains a list of localization files.
Themes	The folder contains a list of default styles in reports.
.exe files	
Designer.exe	The launch file of the WinForms report designer.
Designer.Wpf.exe	The launch file of the WPF report designer V1.
DesignerV2.Wpf.exe	The launch file of the WPF report designer V2.
Stimulsoft libraries:	
Stimulsoft.Accounts.Wpf.dll	The library is used for the authorization of users in the report designer.
Stimulsoft.Base.dll	The main library contains common base interfaces and classes for all products.
Stimulsoft.Client.dll	The library contains the user configuration of the report designer and viewer.
Stimulsoft.Client.Designer.dll	The library contains resources and classes of the WPF report designer V2.
Stimulsoft.Controls.dll	The library contains additional controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Controls.Win.dll	The library contains main controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Dashboard	The library contains the core functionality of the Stimulsoft Dashboards - main objectives, classes, definitions.
Stimulsoft.Dashboard.Design	The library contains resources and control classes of the WinForms dashboard designer.

© 2003-2024 Stimulsoft

Stimulsoft.Dashboard.Drawing.dll	The library is used for drawing the elements of Stimulsoft Dashboards with the help of GDI technology.
Stimulsoft.Dashboard.Drawing.Wpf.dll	The library is used for drawing the elements of Stimulsoft Dashboards with the help of WPF technology.
Stimulsoft.Dashboard.Export.dll	The library contains all the necessary methods for exporting dashboards.
Stimulsoft.Dashboard.Viewer.dll	The library contains resources and control classes of the WinForms dashboard viewer.
Stimulsoft.Dashboard.Viewer.Wpf.dll	The library contains resources and control classes of the WPF dashboard viewer.
Stimulsoft.Dashboard.Viewer.Wpf.Design.dll	The library is required to integrate the WPF dashboard viewer with the Visual Studio toolbox.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Database.dll	The library is used for creating and editing database connections. The library contains the SQL Query Builder.
Stimulsoft.Database.Wpf.dll	The library is used for creating and editing database connections for the WPF report engine. The library contains the SQL Query Builder.
Stimulsoft.Design.dll	The library contains classes for working with reports in Design Time for Visual Studio.
Stimulsoft.Editor.dll	The library contains a text editor which is used in the report designer.
Stimulsoft.Editor.Wpf.dll	The library contains a text editor which is used in the WPF report designer.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The main library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Repor	The library contains all methods for working with the Report

© 2003-2024 Stimulsoft

t.Check.dll	Checker.
Stimulsoft.Report.Design.dll	The library contains resources and control classes of the WinForms report designer.
Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Import.dll	The library contains resources for importing reports.
Stimulsoft.Report.Mvc.dll	The library contains resources and control classes of the viewer and designer for ASP.NET MVC.
Stimulsoft.Report.Publish.dll	The library contains resources for publishing reports.
Stimulsoft.Report.Web.dll	The library contains resources and control classes of the viewer for ASP.NET.
Stimulsoft.Report.WebDesign.dll	The library contains resources and control classes of the designer for ASP.NET.
Stimulsoft.Report.Win.dll	The library contains resources and control classes of the viewer for WinForms.
Stimulsoft.Report.Wpf.dll	The library contains resources and control classes of the viewer for WPF.
Stimulsoft.Report.Wpf.BlackTheme.dll	This library contains the Black theme using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Design.dll	The library needed to integrate the WPF viewer with the Visual Studio toolbox.
Stimulsoft.Report.Wpf.Office2003BlueTheme.dll	This library contains the Office 2003 Blue theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2003OliveGreenTheme.dll	This library contains the Office 2003 Olive Green theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2003SilverTheme.dll	This library contains the Office 2003 Silver theme for using it in the WPF viewer and report designer.

© 2003-2024 Stimulsoft

Stimulsoft.Report.Wpf.Office2007BlackTheme.dll	This library contains the Office 2007 Black theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2007BlueTheme.dll	This library contains the Office 2007 Blue theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2007SilverTheme.dll	This library contains the Office 2007 Silver theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2010BlueTheme.dll	This library contains the Office 2010 Blue theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.Wpf.Office2010WhiteTheme.dll	This library contains the Office 2010 White theme for using it in the WPF viewer and report designer.
Stimulsoft.Report.WpfDesign.dll	The library contains resources and classes of the WPF report designer V1.
Stimulsoft.Report.WpfDesign.Design.dll	The library needed to integrate the WPF designer with the Visual Studio toolbox.
Stimulsoft.Server.Connect.dll	The library contains classes and methods for connecting to the server.
Stimulsoft.Server.Objects.dll	The library contains a description of objects that can be transferred to the server.
Stimulsoft.Wizard.Wpf.dll	The library contains the resources needed to create reports using the report wizards.
External libraries:	
adodb.dll	The library contains the necessary classes for accessing various types of databases.
LibExcel.dll	The library is used for XLS files as a data source in reports and dashboards.
Microsoft.Web.Infrastructure.dll	The library is used to register HTTP modules at run time dynamically.

© 2003-2024 Stimulsoft

Microsoft.Web.XmlTransform.dll	The library is used to convert XML files.
Newtonsoft.Json.dll	The library contains classes for converting objects to JSON objects.
NuGet libraries	All NuGet libraries are required to download Stimulsoft packages from NuGet.
Svg.dll	The library contains functionality for using SVG images in reports and dashboards.
System.Web	All the necessary libraries for publishing Web packages from the report designer.

2.12 Files of Stimulsoft Designer.JS Application

Stimulsoft Designer.JS application package contains the following files:

Name	Description
locales folder	The folder contains a list of localization files.
scripts folder:	The folder contains a list of Stimulsoft scripts.
stimulsoft.dashboards.js	The script for working with dashboards.
stimulsoft.designer.js	The script for working with the report designer.
stimulsoft.designer.runtime.js	The additional script for the standalone report designer.
stimulsoft.reports.js	The main script contains all the functionality of the report generator - rendering, exporting, working with data.
stimulsoft.reports.maps.js	The script contains resources for working with region maps.
stimulsoft.viewer.js	The script for working with the report viewer.
styles folder	The folder contains a list of designer and viewer themes.
Designer.exe file	The launch file of the report designer.

© 2003-2024 Stimulsoft

Information

Also, the **Stimulsoft Designer.JS** directory contains third-party folders, scripts, and files necessary for working with the auxiliary framework.

2.13 Files of Stimulsoft Demo Application

Stimulsoft Demo application package contains the following files:

Name	Description
Folders:	
Localization	The folder contains a list of localization files.
Reports	The folder contains a list of reports and dashboards templates.
.exe files	
Designer.exe	The launch file of the WinForms report designer.
Demo.exe	The launch file of the demo application.
Stimulsoft Libraries:	
Stimulsoft.Accounts.Wpf.dll	The library is used for the authorization of users in the report designer.
Stimulsoft.Base.dll	The main library contains common base interfaces and classes for all products.
Stimulsoft.Client.dll	The library contains the user configuration of the report designer and viewer.
Stimulsoft.Controls.dll	The library contains additional controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Controls.Win.dll	The library contains main controls that are used in the designer and viewer for desktop applications.
Stimulsoft.Dashboard	The library contains the core functionality of the Stimulsoft Dashboards - main objectives, classes, definitions.

© 2003-2024 Stimulsoft

Stimulsoft.Dashboard.Design	The library contains resources and control classes of the WinForms dashboard designer.
Stimulsoft.Dashboard.Drawing.dll	The library is used for drawing the elements of Stimulsoft Dashboards with the help of GDI technology.
Stimulsoft.Dashboard.Drawing.Wpf.dll	The library is used for drawing the elements of Stimulsoft Dashboards with the help of WPF technology.
Stimulsoft.Dashboard.Export.dll	The library contains all the necessary methods for exporting dashboards.
Stimulsoft.Dashboard.Viewer.dll	The library contains resources and control classes of the WinForms dashboard viewer.
Stimulsoft.Dashboard.Viewer.Wpf.dll	The library contains resources and control classes of the WPF dashboard viewer.
Stimulsoft.Data.dll	The library contains classes and methods for data analysis, transformation, and filtering.
Stimulsoft.Database.dll	The library is used for creating and editing database connections. The library contains the SQL Query Builder.
Stimulsoft.Database.Wpf.dll	The library is used for creating and editing database connections for the WPF report engine. The library contains the SQL Query Builder.
Stimulsoft.Design.dll	The library contains classes for working with reports in Design Time for Visual Studio.
Stimulsoft.Editor.dll	The library contains a text editor which is used in the report designer.
Stimulsoft.Editor.Wpf.dll	The library contains a text editor which is used in the WPF report designer.
Stimulsoft.Map.dll	The library contains resources for working with region maps.
Stimulsoft.Report.dll	The main library contains all the functionality of the report generator - rendering, exporting, working with data.
Stimulsoft.Report.Check.dll	The library contains all methods for working with the Report Checker.

© 2003-2024 Stimulsoft

Stimulsoft.Report.Design.dll	The library contains resources and control classes of the WinForms report designer.
Stimulsoft.Report.Design.WebViewer.dll	The library contains resources and control classes of the web viewer.
Stimulsoft.Report.Helper.dll	The library contains resources extended products localization – description of properties and actions, error messages.
Stimulsoft.Report.Import.dll	The library contains resources for importing reports.
Stimulsoft.Report.Publish.dll	The library contains resources for publishing reports.
Stimulsoft.Report.Win.dll	The library contains resources and control classes of the viewer for WinForms.
Stimulsoft.Report.Wpf.dll	The library contains resources and control classes of the viewer for WPF.
Stimulsoft.Server.Connect.dll	The library contains classes and methods for connecting to the server.
Stimulsoft.Server.Objects.dll	The library contains a description of objects that can be transferred to the server.
Stimulsoft.Wizard.Wpf.dll	The library contains the resources needed to create reports using the report wizards.
External libraries:	
adodb.dll	The library contains the necessary classes for accessing various types of databases.
LibExcel.dll	The library is required for using XLS files as a data source in reports and dashboards.
Newtonsoft.Json.dll	The library contains classes for converting objects to JSON objects.
NuGet libraries	All NuGet libraries are required to download Stimulsoft packages from NuGet.
Svg.dll	The library contains functionality for using SVG images in reports and dashboards.

© 2003-2024 Stimulsoft

3 Reports and Dashboards for ASP.NET WebForms

ASP.NET is the technology for creating web applications and web services. Stimulsoft provides tools for creating and displaying reports and dashboards on any device using this technology.

Tools for creating and editing reports:	Tools for viewing and converting reports:	Tools for creating and editing dashboards:	Tools for viewing and converting dashboards:
HTML5 designer	HTML5 viewer	HTML5 designer	HTML5 viewer

3.1 HTML5 Viewer

YouTube

Watch videos for the ASP.NET HTML5 Viewer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

Samples

See on GitHub examples for working with the ASP.NET HTML5 Viewer component. All examples are separate projects grouped into one solution for Visual Studio.

The **HTML5 Viewer** (**StiWebViewer**) component is designed for viewing reports in the web browser. You do not need to install the .NET Framework, ActiveX components, or any special plug-ins on the client-side. All you need is any modern Web browser.

© 2003-2024 Stimulsoft

With the help of the **HTML5 Viewer**, you can view, print, and export reports on any computer with any operating system installed. Since the viewer only uses HTML and JavaScript technologies, it can be run on devices with no Flash or Silverlight support - tablets, smartphones. Also, the viewer supports Mobile and Touch interfaces, which automatically enable when using mobile devices and touch screen monitors.

The **HTML5 Viewer** component uses the AJAX technology to perform all actions (uploading a report, paging, scaling, interactivity in reports, etc.), allowing you to get rid of reloading the entire page and save Web traffic, and speed up work.

The **HTML5 Viewer** supports many themes, animated interface, bookmarks, interactive reports, editing of report elements on the page, full-screen mode, search panel, and other necessary features for viewing reports.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Viewer** in a Web project, you need to install the NuGet package of Stimulsoft.Reports.Web:

- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Reports.Web in the search bar on the Browse tab;
- Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

If this is not possible, you should add the following assemblies to the project:

- Stimulsoft.Base.dll
- Stimulsoft.Report.dll
- Stimulsoft.Report.Check.dll
- Stimulsoft.Report.Helper.dll
- Stimulsoft.Report.Web.dll

To add the ability to view and export dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web (this package is associated with the package Stimulsoft.Reports.Web. If it is missed, it will be installed automatically):

© 2003-2024 Stimulsoft

- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Dashboards.Web in the search bar on the Browse tab;
- Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

If for any reason this is not possible, you should additionally add the following assemblies to the project:

- Stimulsoft.Dashboard.dll
- Stimulsoft.Dashboard.Drawing.dll
- Stimulsoft.Dashboard.Export.dll

- How this Works?
- Activation
- Showing Reports and Dashboards
- Connecting Data
- Localization
- Printing Reports
- Exporting Reports and Dashboards
- Viewing Modes
- Work with Parameters
- Work with Bookmarks
- Interactive Reports
- Timeout
- Editing Rendered Report
- Sending Report by Email
- Calling Designer and Viewer
- Caching
- Export and Printing from Code
- Basic Features
- Viewer Events
- Viewer Settings

3.1.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

To run the viewer, you need to place the **StiWebViewer** component on the ASPX page, set the necessary properties, and, if necessary, set the necessary event handlers. When the report viewer runs, the following actions occur:

› The .NET component generates HTML and JavaScript code that is necessary for displaying and running the viewer;

› When the component is output, the JavaScript method is launched. It requests the first page of the report on the server-side or the entire report (depending on the selected mode) and the required report parameters;

› Each action in the viewer (for example, paging, printing or exporting a report, etc.) calls a certain action on the server-side, in which you can perform the necessary manipulations with the report by subscribing to the corresponding viewer events;

› The viewer saves the report in cache or server session to speed up the work, which makes it impossible to re-build the report.

3.1.2 Activation

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiWebViewer** component.

Default.aspx.cs

```
...
public partial class _Default : Page
{
   static _Default()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
      var path = HttpContext.Current.Server.MapPath("license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

3.1.3 Showing Reports and Dashboards

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

To show a report, you should add the **StiWebViewer** component to the ASPX page and assign a loaded report to it.

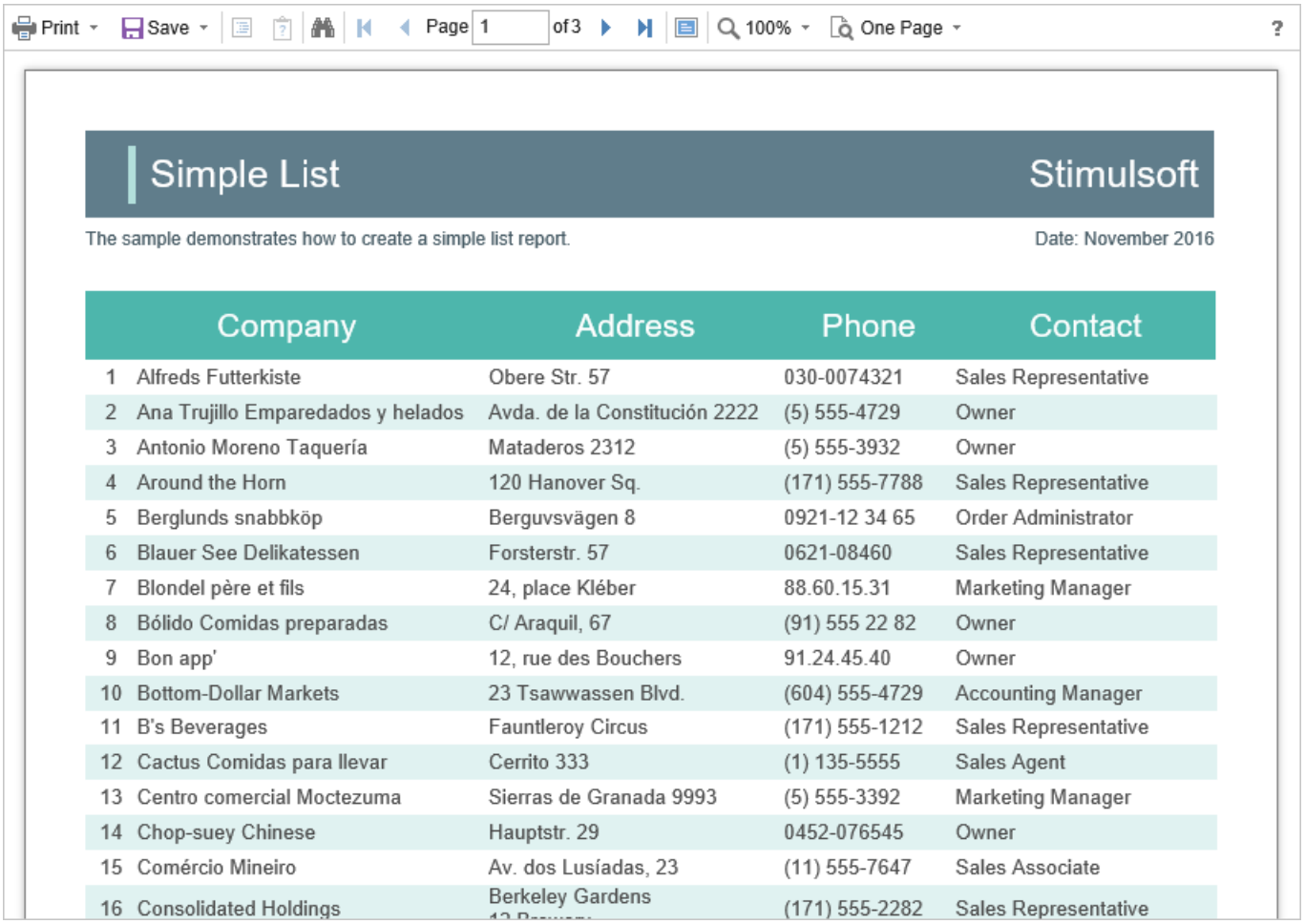
Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));
   // report.Load(Server.MapPath("Reports/Dashboard.mrt"));

   StiWebViewer1.Report = report;
}
...
```

© 2003-2024 Stimulsoft



Also, the **HTML5 Viewer** has a special **OnGetReport** event that you can use to assign a report to the viewer. In this case, you need to load the report in the event handler.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   OnGetReport="StiWebViewer1_GetReport">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_GetReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = new StiReport();
```

© 2003-2024 Stimulsoft

```
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));

   e.Report = report;
}
...
```

Information

To assign a report, you should use the specified OnGetReport event. In this case, if the report object is lost for any reason in the cache of the server or session, the client part of the viewer initiates this event, and the report preview will be continued.

If the report is not rendered before showing, the **HTML5 Viewer** component will automatically generate it. So, you can use various types of reports to display the report - report templates, rendered reports, and reports as classes.

Default.aspx.cs

```
...
protected void StiWebViewer1_GetReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = new StiReport();
   report.LoadDocument(Server.MapPath("Reports/SimpleList.mdc"));

   e.Report = report;
}
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_GetReport(object sender,
StiReportDataEventArgs e)
{
   e.Report = new StiReportCompiledClass();
}
...
```

Since the dashboard is not a static document and requires data to work, the format of the rendered MDC document is not available for it. Instead, it is possible to use a snapshot of the report in the MRT format, which contains all the data necessary for

© 2003-2024 Stimulsoft

the dashboard to work and can be correctly displayed in the viewer.

Default.aspx.cs

```
...
protected void StiWebViewer1_GetReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/Snapshot.mrt"));

   e.Report = report;
}
...
```

Loading custom fonts

You can load custom fonts using the **StiFontCollection** class, having specified the file that contains a font. To do this, you should call the static method in the constructor to load a font.

Default.aspx.cs

```
...
public partial class _Default : Page
{
   static _Default()
   {
      Stimulsoft.Base.StiFontCollection.AddFontFile(Server.MapPath("fonts/
      my-font/font-name.ttf"));
   }
}
...
```

3.1.4 Connecting Data

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Data for rendering a report can be connected in various ways. The easiest way is to store connection settings in the report template. Also, data can be connected from the code. You can do this before assigning the report to the viewer.

© 2003-2024 Stimulsoft

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   DataSet ds = new DataSet();
   ds.ReadXml(Server.MapPath("Reports/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/TwoSimpleLists.mrt"));
   report.Dictionary.Databases.Clear();
   report.RegData("Demo", ds);

   StiWebViewer1.Report = report;
}
...
```

To connect report data, you can use the special **OnGetReportData** event, which will be called before the report is rendered.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   OnGetReportData="StiWebViewer1_GetReportData">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_GetReportData(object sender,
StiReportDataEventArgs e)
{
   DataSet dataSet = new DataSet();
   dataSet.ReadXml(Server.MapPath("Reports/Demo.xml"));
   e.Report.RegData(dataSet);
}
...
```

SQL data sources

© 2003-2024 Stimulsoft

The connection parameters to the SQL data source and any other ones can be stored in the report template. Suppose you want to set the connection parameters from the code before rendering the report (for example, for security reasons or depending on the authorized user). In that case, you can use the example below.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   OnGetReportData="StiWebViewer1_GetReportData">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_GetReportData(object sender,
StiReportDataEventArgs e)
{
   OracleConnection connection = new OracleConnection("Data
   Source=Oracle8i;Integrated Security=yes");
   connection.Open();
   OracleDataAdapter adapter = new OracleDataAdapter();
   adapter.SelectCommand = new OracleCommand("SELECT * FROM Products",
   connection);

   DataSet dataSet = new DataSet("productsDataSet");
   adapter.Fill(dataSet, "Products");

   e.Report.RegData("Products", dataSet);
}
...
```

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection and data sources in the report.

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
```

© 2003-2024 Stimulsoft

```
    StiReport report = new StiReport();
    report.Load(Server.MapPath("Report.mrt"));
    ((StiSqlDatabase)
    report.Dictionary.Databases["Connection"]).ConnectionString = @"Data
    Source=server;Integrated Security=True;Initial Catalog=DataBase";
    ((StiSqlSource)
    report.Dictionary.DataSources["DataSourceName"]).SqlCommand = "select *
    from Table where Column = 100";
    ((StiSqlSource)
    report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

    StiWebViewer1.Report = report;
}
...
```

Information

For SQL data sources of other types, the connection is created similarly, and an adapter corresponding to the data source type is connected. For example, for the MS SQL data source, you should connect SqlDataAdapter. For OLE DB - OleDbDataAdapter is required. Also, you should specify a connection string that matches the connection type.

You can also use data for designing reports and dashboards obtained from OData storage. In this case, you can do the authorization using a username, user password, or token. Authorization parameters are specified in the connection string to the OData storage using the ";" separator.

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{

    var report = new StiReport();

    //Authorization using a user account
    var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
    services.odata.org/V4/Northwind/
    Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Password;C
    lient_Id=Your Client ID", false, null);

    //Authorization using a user token
    var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
    services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your token",
    false, null);
```

© 2003-2024 Stimulsoft

```
    report.Dictionary.Databases.Add(oDataDatabase);
    oDataDatabase.Synchronize(report);

    //Query with data filter
    ((StiSqlSource)report.Dictionary.DataSources["Products"]).SqlCommand =
    "Products?$filter=ProductID eq 2";

    StiWebViewer1.Report = report;
}
...
```

The table below shows the connection string templates for different types of data sources.

Data Source	Samples of Connection Strings
MS SQL	Integrated Security=False; Data Source=myServerAddress;Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
MySQL	Server=myServerAddress; Database=myDataBase;UserId=myUsername; Pwd=myPassword;
ODBC	Driver={SQL Server}; Server=myServerAddress;Database=myDataBase; Uid=myUsername; Pwd=myPassword;
OLE DB	Provider=SQLOLEDB.1; Integrated Security=SSPI;Persist Security Info=False; Initial Catalog=myDataBase;Data Source=myServerAddress
Oracle	Data Source=TORCL;User Id=myUsername;Password=myPassword;
MS Access	Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Password=pass;Data Source=C:\ \myAccessFile.accdb;
PostgreSQL	Server=myServerAddress; Port=5432; Database=myDataBase;User Id=myUsername; Password=myPassword;

© 2003-2024 Stimulsoft

Firebird	User=SYSDBA; Password=masterkey; Database=SampleDatabase.fdb;DataSource=myServerAddress; Port=3050; Dialect=3; Charset=NONE;Role=; Connection lifetime=15; Pooling=true; MinPoolSize=0;MaxPoolSize=50; Packet Size=8192; ServerType=0;
SQL CE	Data Source=c:\MyData.sdf; Persist Security Info=False;
SQLite	Data Source=c:\mydb.db; Version=3;
DB2	Server=myAddress:myPortNumber;Database=myDataBase;UID=myUsername;PWD=myPassword; Max Pool Size=100;Min Pool Size=10;
Infomix	Database=myDataBase;Host=192.168.10.10;Server=db_engine_tcp;Service=1492;Protocol=onsoctcp;UID=myUsername;Password=myPassword;
Sybase	Data Source=myASEserver;Port=5000;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
Teradata	Data Source=myServerAddress;User ID=myUsername;Password=myPassword;
VistaDB	Data Source=D:\folder \myVistaDatabaseFile.vdb4;Open Mode=ExclusiveReadWrite;
Universal(dotConnect)	Provider=Oracle;direct=true;data source=192.168.0.1;port=1521;sid=sid;user=user; password=pass
MongoDB	mongodb://<user>:<password>@localhost/test
OData	http://services.odata.org/v3/odata/OData.svc/
Other...	The table shows the most commonly used templates for the connection string. You can view various connection string options at the special website.

Data from XML, JSON, Excel files

© 2003-2024 Stimulsoft

Connecting to XML and JSON data sources can be stored in the report template. If you want to specify data files from the code, you can use the example below.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  OnGetReportData="StiWebViewer1_GetReportData">
</cc1:StiWebViewer>
...
```

Default.aspx.cs (XML file)

```
...
protected void StiWebViewer1_GetReportData(object sender,
StiReportDataEventArgs e)
{
  DataSet data = new DataSet();
  data.ReadXml(Server.MapPath("Data/Demo.xml"));

  e.Report.RegData(data);
}
...
```

Default.aspx.cs (JSON file)

```
...
protected void StiWebViewer1_GetReportData(object sender,
StiReportDataEventArgs e)
{
  DataSet data
  = StiJsonToDataSetConverterV2.GetDataSetFromFile(Server.MapPath("Data/
  Demo.json"));

  e.Report.RegData(data);
}
...
```

Information

The viewer has the possibility of obtaining data from an Excel file. To do this, you can use the following method.

```
DataSet dataSet = StiExcelConnector.Get().GetDataSet(new StiExcelOptions(ar
```

© 2003-2024 Stimulsoft

3.1.5 Localization

The **HTML5 Viewer** component supports the complete localization of its interface. To localize the report viewer interface, use the special **Localization** property. As a value of this property, you should specify the path to the localization XML file (relative or absolute).

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  Localization="Localization/en.xml">
</cc1:StiWebViewer>
...
```

When you load the report viewer, the localization file will be loaded automatically.

3.1.6 Using Themes

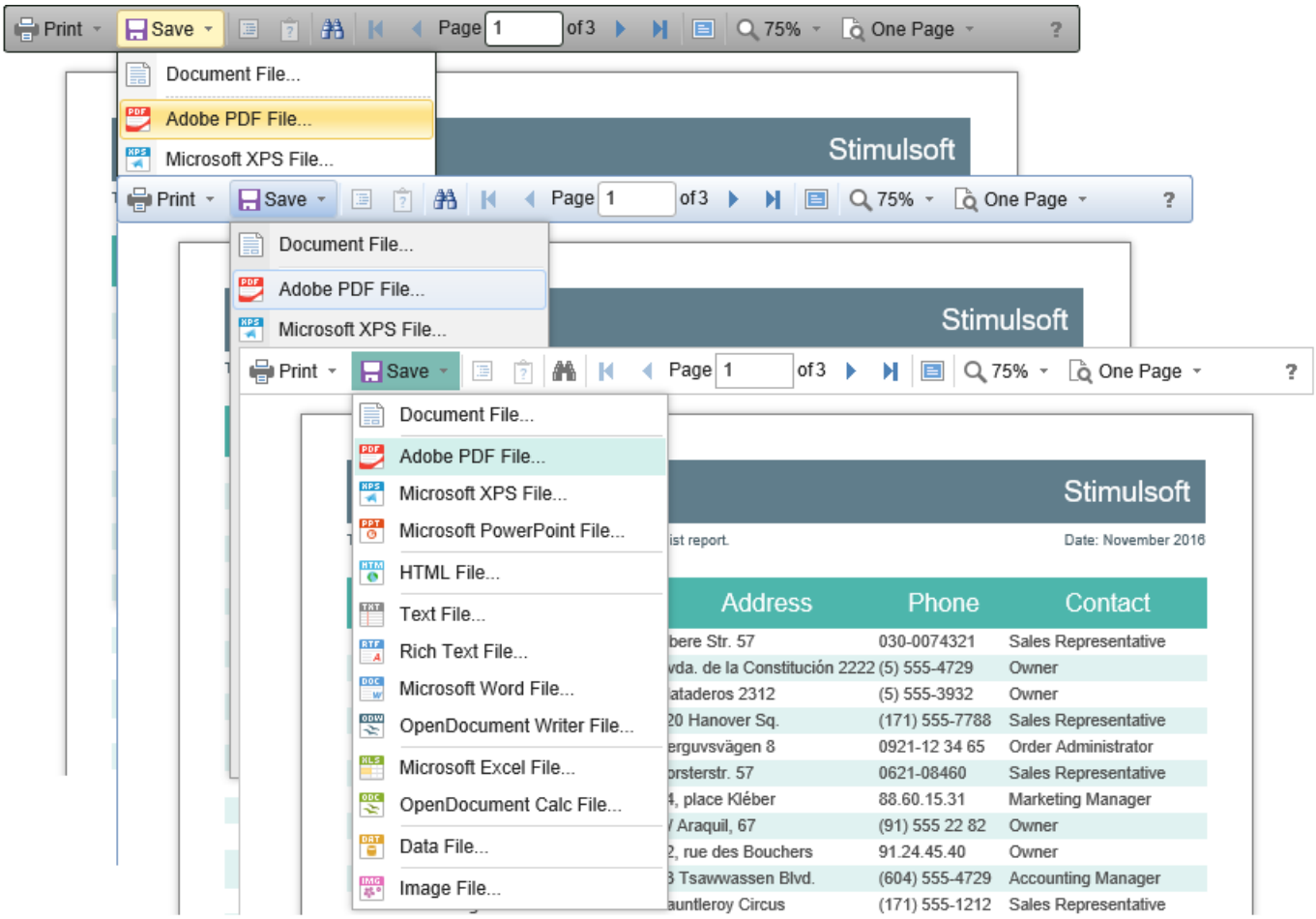
In the **HTML5 Viewer** component, you can change the appearance of visual controls. To change the theme, you should use the **Theme** property.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  Theme="Office2022WhiteTeal">
</cc1:StiWebViewer>
...
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.

© 2003-2024 Stimulsoft

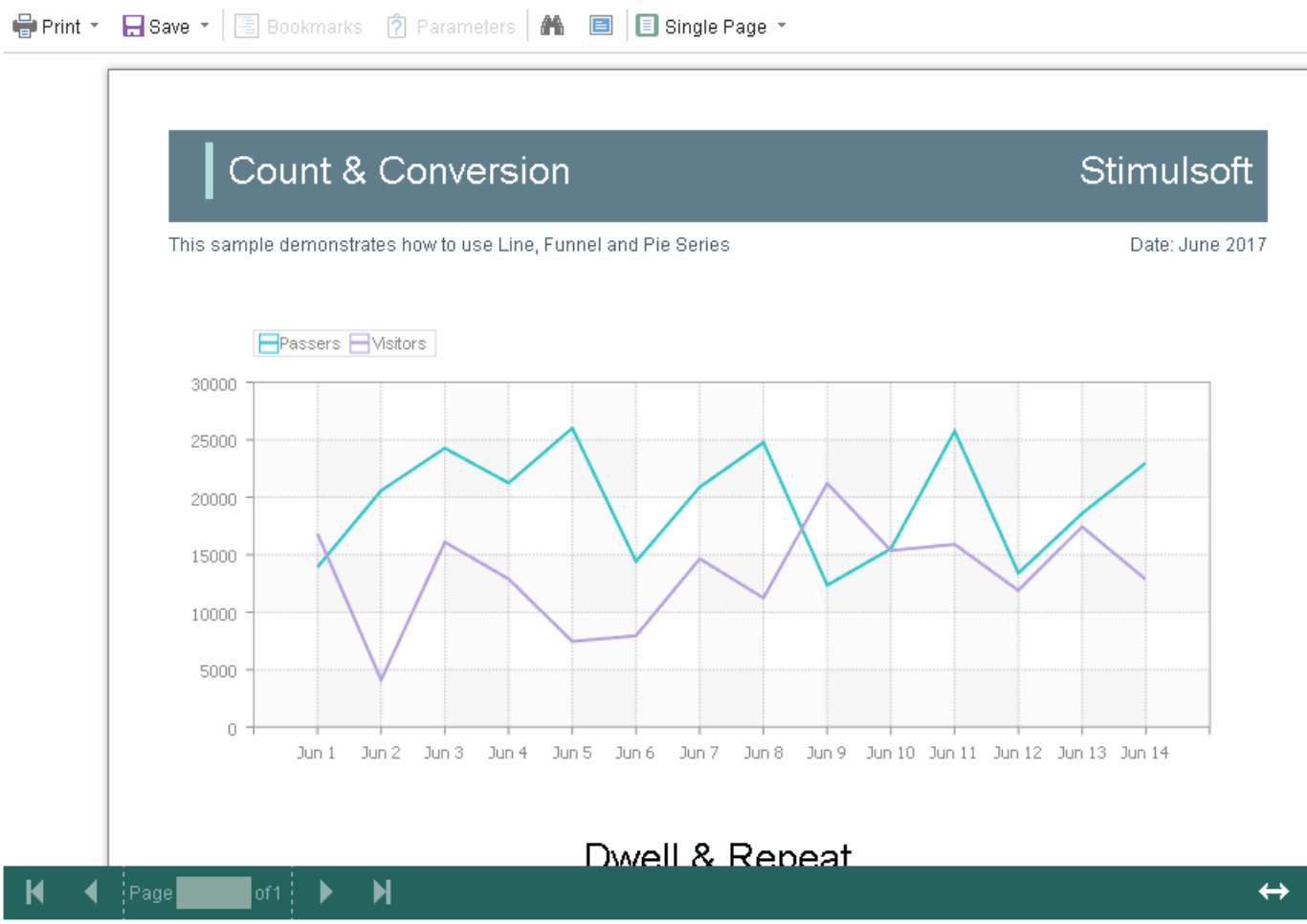


By default, the viewer has only the top toolbar on which all the report controls are located. If necessary, the toolbar can be split into top and bottom parts. The top panel will contain the menu for printing and exporting the report and the buttons for working with parameters and bookmarks. The bottom panel will contain controls to switch between the report pages and setting the zoom of pages. To enable this mode, enable the **ToolbarDisplayMode** property. It has values **Simple** and **Separated**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ToolbarDisplayMode="Separated"
   ScrollbarsMode="true">
</cc1:StiWebViewer>
...
```

© 2003-2024 Stimulsoft



In addition, it is possible to set the appearance parameters for the main elements of the viewer. For example, you can change the font and color of the control panel inscriptions of the viewer, set the background of the viewer, set the color of page borders, etc. Below is a list of available properties that change the appearance of the viewer and their default values.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   BackgroundColor="White"
   ShowPageShadow="true"
   PageBorderColor="Gray"
   ToolbarBackgroundColor="Empty"
   ToolbarBorderColor="Empty"
   ToolbarFontColor="Empty"
   ToolbarFontFamily="Arial">
</cc1:StiWebViewer>
...
```

© 2003-2024 Stimulsoft

3.1.7 Basic Features

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The main features of the viewer include the following operations - showing reports, switching between report pages, changing the scale, and displaying the preview mode. All specified operations are performed in the AJAX mode without restarting the browser page. No special options or events are required to execute them.

The report viewer has a special **OnViewerEvent** event that will be called when the viewer is running. In this event, you can find out the type of action that the viewer is currently executing and get all viewer parameters passed to the server-side.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   OnViewerEvent="StiWebViewer1_ViewerEvent">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_ViewerEvent(object sender, StiViewerEventArgs
e)
{
   StiAction action = e.Action;
   StiRequestParams parameters = e.RequestParams;
}
...
```

3.1.8 Printing Reports

Information

Please note that the print option is available only for reports, and not for dashboards.

© 2003-2024 Stimulsoft

The **HTML5 Viewer** component provides several options for printing a report. Each has its advantages and disadvantages.

Print to PDF
Printing will be done by exporting the report to **PDF**. The advantages are greater accuracy of positioning and printing of the report elements compared to other printing options. Among the drawbacks is the mandatory presence of a plug-in installed in a web browser for viewing PDF files (modern browsers have embedded PDF viewer and printer).

Print with Preview
The report will be printed in a separate pop-up browser window in **HTML**. The report can be previewed and then sent to the printer or copied to another location as text or HTML code. Advantages - cross-browser compatibility when printing, no need to install special plug-ins. The disadvantage is the relatively low accuracy of the position of the report elements due to the peculiarities of the implementation of HTML formatting.

Print without Preview
The report will be printed directly to the printer without preview. After selecting this menu item, the system print dialog is displayed. Since printing in this mode is carried out in the **HTML** format, the print quality is similar to printing a report with a preview.

Information

When printing to the **HTML format**, you should check the compliance of report page settings to printer parameters (paper size, orientation, margins, indents), and check your browser print settings, such as margins, headers, footers, watermarks printing, color printing.

Report printing events

To perform any actions, a special **OnPrintReport** event is assigned before the report is printed. In this event, you can get the type of report printing, the report itself, and

© 2003-2024 Stimulsoft

the export report settings in case of printing to **PDF**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
    OnPrintReport="StiWebViewer1_PrintReport">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_PrintReport(object sender,
StiPrintReportEventArgs e)
{
   StiPrintAction action = e.PrintAction;
   StiReport report = e.Report;
   StiExportSettings settings = e.Settings;
}
...
```

Print setup

If you choose to print a report in the viewer panel, a menu with printing options is displayed. The **HTML5 Viewer** component can force the required printing mode. To do this, set the **PrintDestination** property to one of the following values.

- **Default** – the menu will be displayed (the default property value);
- **Pdf** – print to the PDF format;
- **Direct** – printing to the HTML format directly to the printer, the system print dialog will be displayed;
- **WithPreview** – print to HTML format with preview in a pop-up window.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   PrintDestination="Default">
</cc1:StiWebViewer>
...
```

The **HTML5 Viewer** component can completely disable report printing. To do this, set the value of the **ShowPrintButton** property to **false**.

© 2003-2024 Stimulsoft

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
    ShowPrintButton="false">
</cc1:StiWebViewer>
...
```

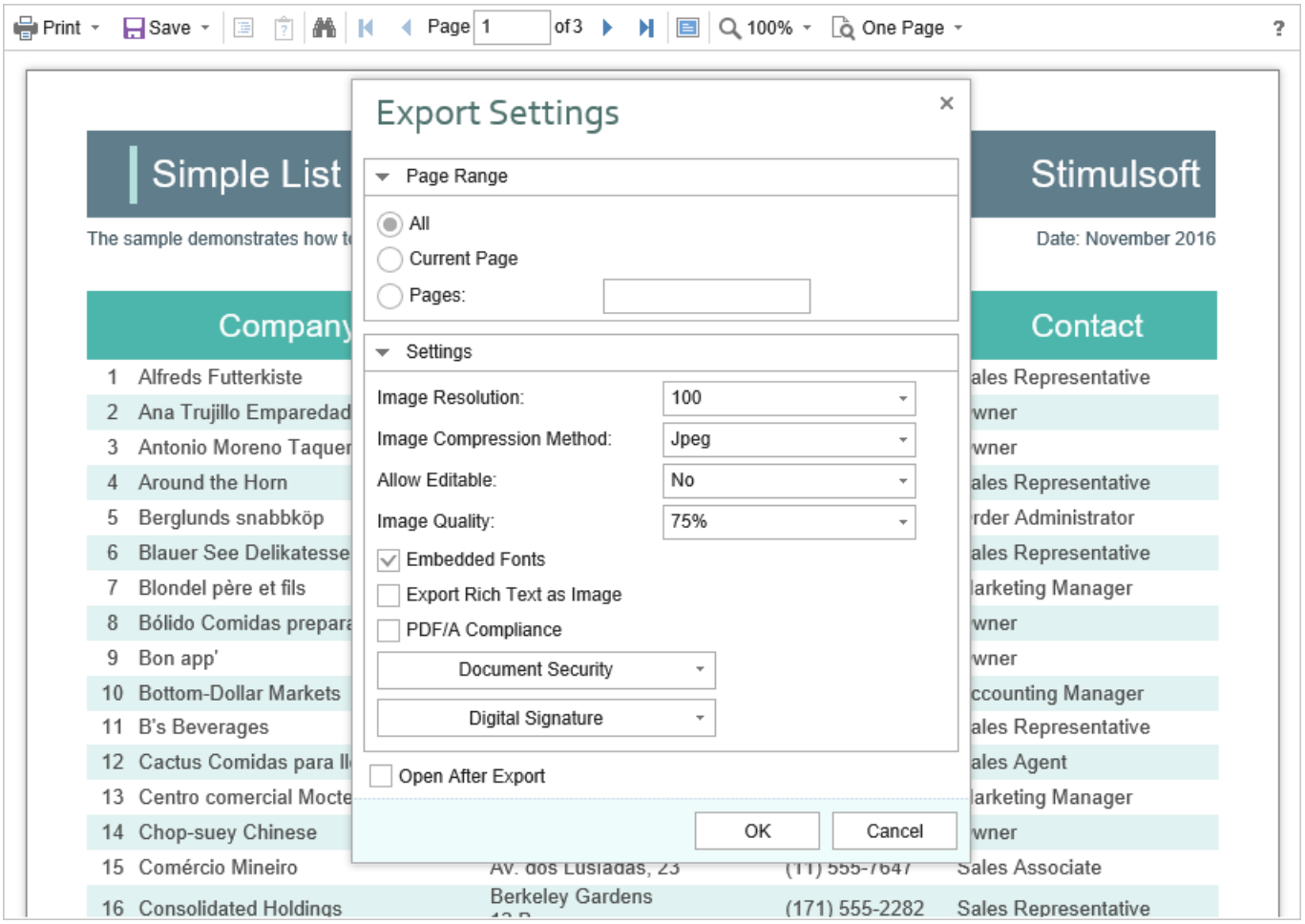
3.1.9 Exporting Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to export the displayed report to three dozen various formats, such as **PDF**, **HTML**, **Word**, **Excel**, **XPS**, **RTF**, **images**, **text**, and others. Export functions do not require additional settings for the viewer. You may export the dashboard to PDF, Excel, image files.

© 2003-2024 Stimulsoft



Export Events

To perform any actions, a special **OnExportReport** event is assigned before the report is exported. In this event, you can get the report export type, get the report itself, get the report export settings, and change them if necessary.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  OnExportReport="StiWebViewer1_ExportReport">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_ExportReport(object sender,
```

© 2003-2024 Stimulsoft

```
StiExportReportEventArgs e)
{
   StiExportFormat format = e.Format;
   StiReport report = e.Report;
   StiExportSettings settings = e.Settings;
}
...
```

To perform any actions with an already exported report, the **OnExportReportResponse** event is used. This event will be called immediately before the file is saved. In this event, you can get the export format, the type of the Web content, and the name of the file to save. You can also get and, if necessary, change the byte stream of the final export file.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   OnExportReportResponse="StiWebViewer1_ExportReportResponse">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_ExportReportResponse(object sender,
StiExportReportResponseEventArgs e)
{
   StiExportFormat format = e.Format;
   string contentType = e.ContentType;
   string fileName = e.FileName;
   Stream stream = e.Stream;
}
...
```

Export Settings

Each report export format of the **HTML5 Viewer** component has a lot of settings, and each setting has its default values. Sometimes you need to set other default values. For this purpose, a special **DefaultExportSettings** property of the viewer is used. It is a container for all the default export settings.

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
```

© 2003-2024 Stimulsoft

```
{
   StiWebViewer1.DefaultExportSettings.ExportToPdf.ImageQuality = 0.75f;
   StiWebViewer1.DefaultExportSettings.ExportToPdf.ImageFormat =
   StiImageFormat.Color;
   StiWebViewer1.DefaultExportSettings.ExportToHtml.ExportMode =
   StiHtmlExportMode.Div;
}
...
```

When calling the required export through the Viewer menu, the Export Settings dialog box will be displayed. The values of the dialog box items will match the default export settings. If you change the settings in the dialog box and confirm the export of the report, the settings will be saved in the browser cookies, and upon the subsequent call, the export will be restored. The viewer allows you to disable saving the export settings if you always want to restore the default settings. To do this, just set the value of the **StoreExportSettings** property to **false**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   StoreExportSettings="false">
</cc1:StiWebViewer>
...
```

Also, if required, you can completely hide export dialogs. Exporting will always be done with default settings. For this, it is enough to set the value of the **ShowExportDialog** property to **false**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowExportDialog="false">
</cc1:StiWebViewer>
...
```

The **HTML5 Viewer** component contains 30+ export formats, and sometimes you need to disable unwanted formats. This allows you to simplify UI and the use of the viewer. To disable unused export formats, it is enough to set the values for the corresponding properties of the viewer listed in the list below to **false**.

© 2003-2024 Stimulsoft

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowExportToDocument="true"
   ShowExportToPdf="true"
   ShowExportToXps="true"
   ShowExportToPowerPoint="true"
   ShowExportToHtml="true"
   ShowExportToHtml5="true"
   ShowExportToMht="true"
   ShowExportToText="true"
   ShowExportToRtf="true"
   ShowExportToWord2007="true"
   ShowExportToOpenDocumentWriter="true"
   ShowExportToExcel="true"
   ShowExportToExcelXml="true"
   ShowExportToExcel2007="true"
   ShowExportToOpenDocumentCalc="true"
   ShowExportToCsv="true"
   ShowExportToDbf="true"
   ShowExportToXml="true"
   ShowExportToDif="true"
   ShowExportToSylk="true"
   ShowExportToImageBmp="true"
   ShowExportToImageGif="true"
   ShowExportToImageJpeg="true"
   ShowExportToImagePcx="true"
   ShowExportToImagePng="true"
   ShowExportToImageTiff="true"
   ShowExportToImageMetafile="true"
   ShowExportToImageSvg="true"
   ShowExportToImageSvgz="true">
</cc1:StiWebViewer>
...
```

The **HTML5 Viewer** component can completely disable the export menu. To do this, set the value of the **ShowSaveButton** property to **false**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowSaveButton="false">
</cc1:StiWebViewer>
...
```

3.1.10 Viewing Modes

The **HTML5 Viewer** component has two modes for displaying reports - with and without scrollbars. By default, the view mode without scrollbars is set. To enable the

© 2003-2024 Stimulsoft

scrollbar view mode, set the value of the **ScrollbarsMode** property to **true**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ScrollbarsMode="false">
</cc1:StiWebViewer>
...
```

In the first mode (without scrollbars), the viewer displays a page or report as a whole, automatically stretching the preview space. If the width and height are specified, the viewer will truncate the page that is out of bounds. In the second mode, unlike the first one, when the page is out of bounds of the viewer's size, no truncation will be performed. Scrollbars will appear, using which you can view the entire page or report.

Information

In the report mode with scrollbars, you should set the height of the viewer. Otherwise, the default height will be set to **650 pixels**.

The **HTML5 Viewer** component provides the full-screen report or dashboard mode. By default, the standard view mode is enabled, the viewer has the specified dimensions in the settings. To enable the full-screen mode, set the **FullScreenMode** property to **true**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   FullScreenMode="false">
</cc1:StiWebViewer>
...
```

Also, to enable or disable the full-screen mode, you can use the corresponding button on the control panel of the viewer.

The **HTML5 Viewer** component has three modes to display reports - page-by-page,

© 2003-2024 Stimulsoft

entire report, and tabular display of report pages. To control the modes, the **ViewMode** property is used. It can have one of the specified values - **SinglePage**, **Continuous**, **MultiplePages**.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
    ViewMode="OnePage">
</cc1:StiWebViewer>
...
```

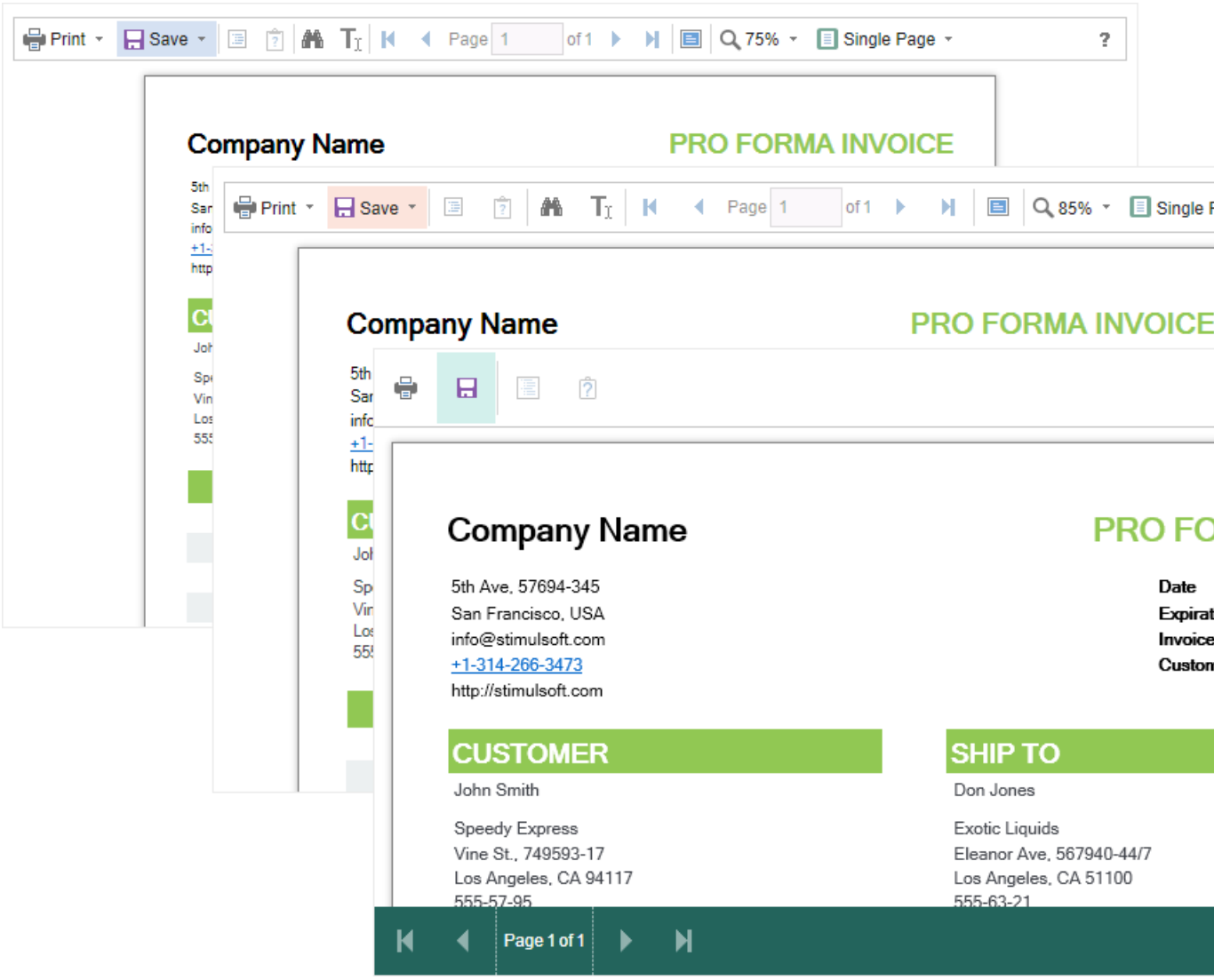
HTML5 Viewer component supports interaction on a regular PC display and works with a touchscreen of screens and mobile devices. **InterfaceType** property allows controlling the interface modes. The property can have one of the following values:

- **Auto** – the viewer's interface is determined automatically depending on the device the report is displayed on. That is the default value.
- **Mouse** – the standard interface with a mouse control will be used for all the screen types.
- **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability.
- **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with smartphones.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  InterfaceType="Auto">
</cc1:StiWebViewer>
...
```

© 2003-2024 Stimulsoft



3.1.11 Work with Parameters

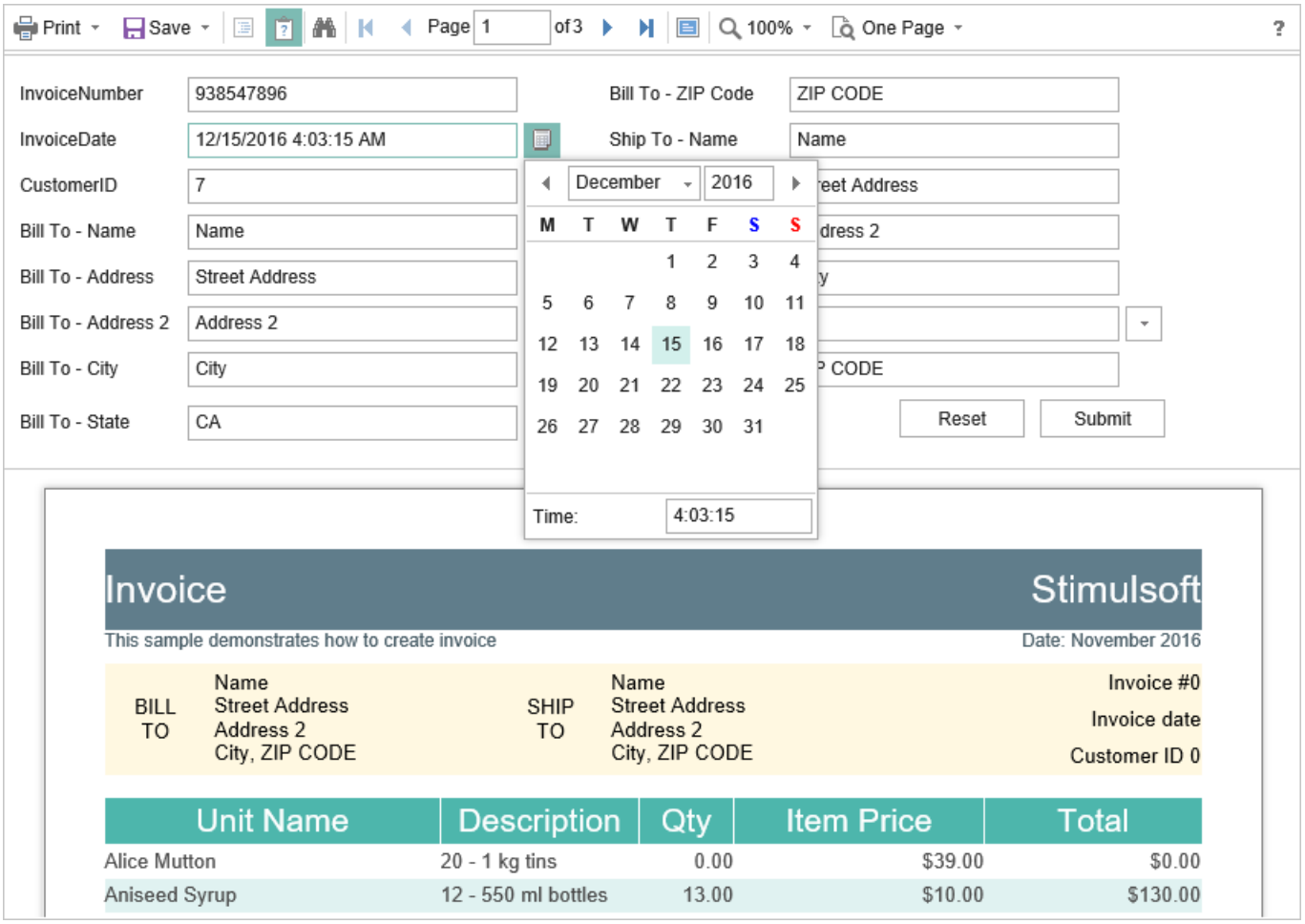
Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To work with report parameters in the **HTML5 Viewer**, there is a special settings panel. To add a parameter to the panel, you need to define a variable in a report requested by the user. When viewing a report in the viewer, such a variable will be

© 2003-2024 Stimulsoft

automatically added to the settings panel. It supports all types of report variables (normal variables, date and time, borders, lists, etc.).



To perform any action before applying parameters, a special **OnInteraction** event will be called when there are some interactive activities in the viewer. When you use the options panel, the type of action will have the **Variables** value.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  OnInteraction="StiWebViewer1_Interaction">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

© 2003-2024 Stimulsoft

```
...
protected void StiWebViewer1_Interaction(object sender,
StiReportDataEventArgs e)
{
   if (e.Action == StiAction.Variables)
   {
      // The values of the variables passed from the client
      Hashtable variables = e.RequestParams.Interaction.Variables;
   }
}
...
```

When working with parameters is not required, you can disable this feature. For this, you can use the **ShowParametersButton** property. Set this property to **false** in this case.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowParametersButton="false">
</cc1:StiWebViewer>
...
```

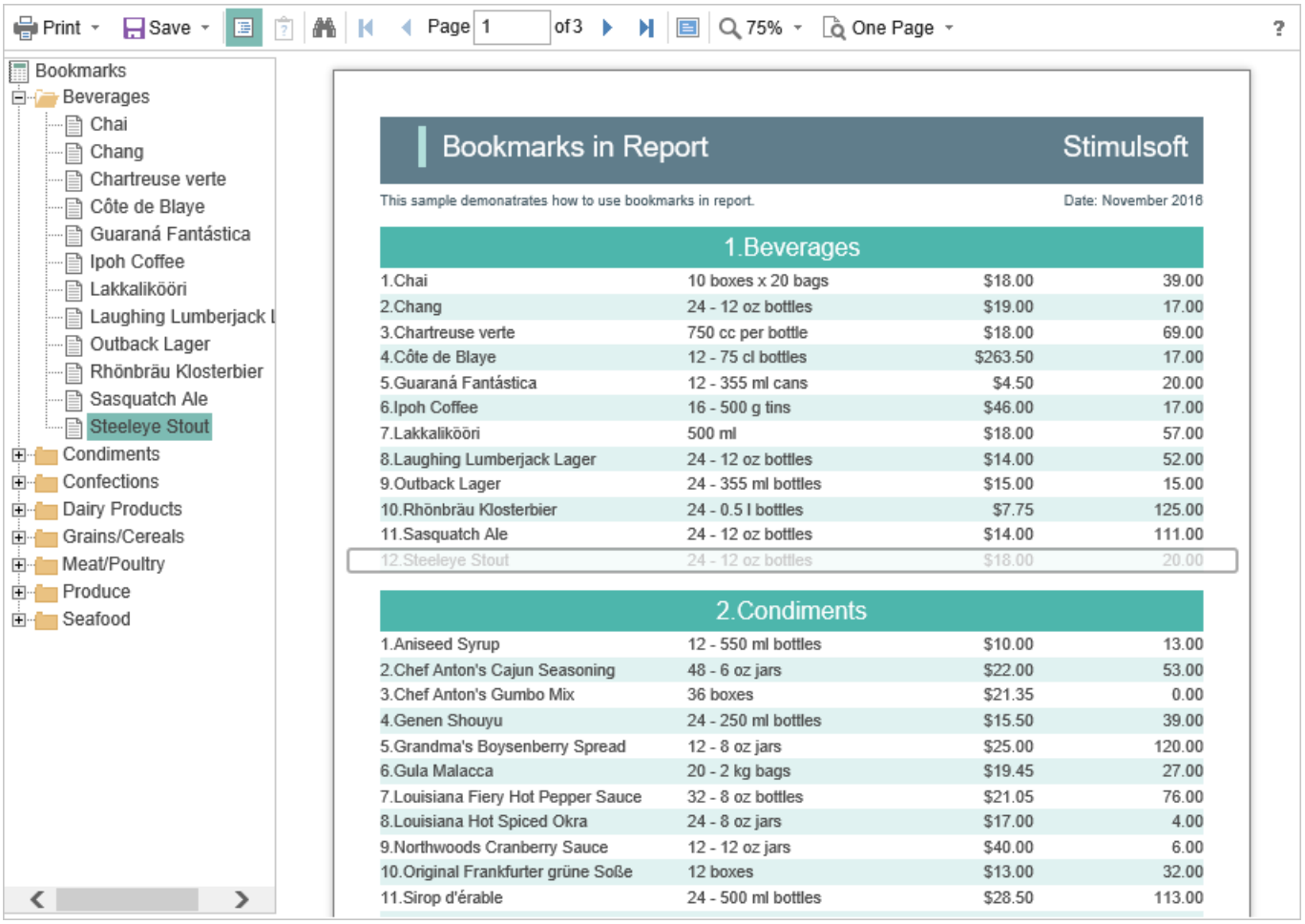
Information

With this viewer configuration, the parameters panel will not be displayed, even if the parameters are present in the report.

3.1.12 Work with Bookmarks

The **HTML5 Viewer** component supports report bookmarks. A panel with bookmarks will be displayed when displaying such a report on the left side of the page. When you select a bookmark of the report, the viewer will carry out an automatic transition to the specified page, and the report item with a bookmark is highlighted.

© 2003-2024 Stimulsoft



By default, the bookmarks bar width is 180 pixels. The **HTML5 Viewer** component allows you to change this value. For this, the **BookmarksTreeWidth** property, which value is specified in pixels, is used.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
    BookmarksTreeWidth="200">
</cc1:StiWebViewer>
...
```

If you do not need to work with report bookmarks, you can disable this option. For this, set the **ShowBookmarksButton** property to **false**.

Default.aspx

© 2003-2024 Stimulsoft

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
    ShowBookmarksButton="false">
</cc1:StiWebViewer>
...
```

Information

In this case, report bookmarks will not be displayed, even if they are present in the displayed report. This property does not affect printing and exporting reports.

When printing a report with bookmarks, the bookmark tree will be hidden. If you want to print bookmarks with the report, it is necessary to set the **BookmarksPrint** property to **true**.

Default.aspx

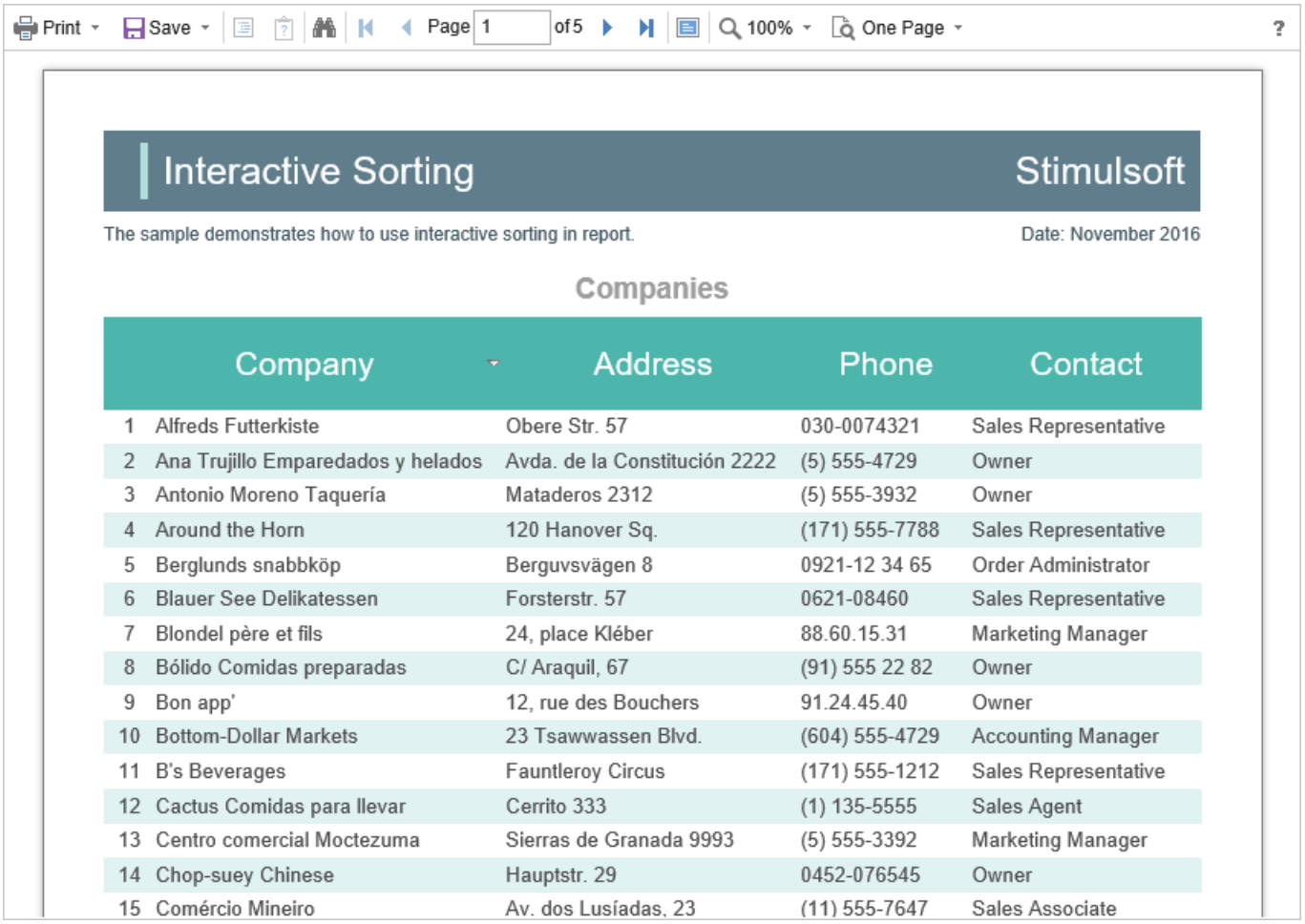
```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
    BookmarksPrint="true">
</cc1:StiWebViewer>
...
```

3.1.13 Dynamic Sorting, Collapsing, and Drill-Down

The **HTML5 Viewer** component supports dynamic sorting, collapsing, and drill-down of reports. Dynamic sorting provides the ability to change the direction of sorting in a rendered report. To do this, click on the component that has dynamic sorting enabled. Dynamic sorting is carried out in the following directions - **Ascending** and **Descending**. Each time when you click the component, the sorting direction is reversed.

Multi-level sorting is allowed in the report. To do this, hold down the **Ctrl** key and sequentially click on the sorted components in the report. To reset sorting, you can click on any sorted component without holding down the **Ctrl** key.

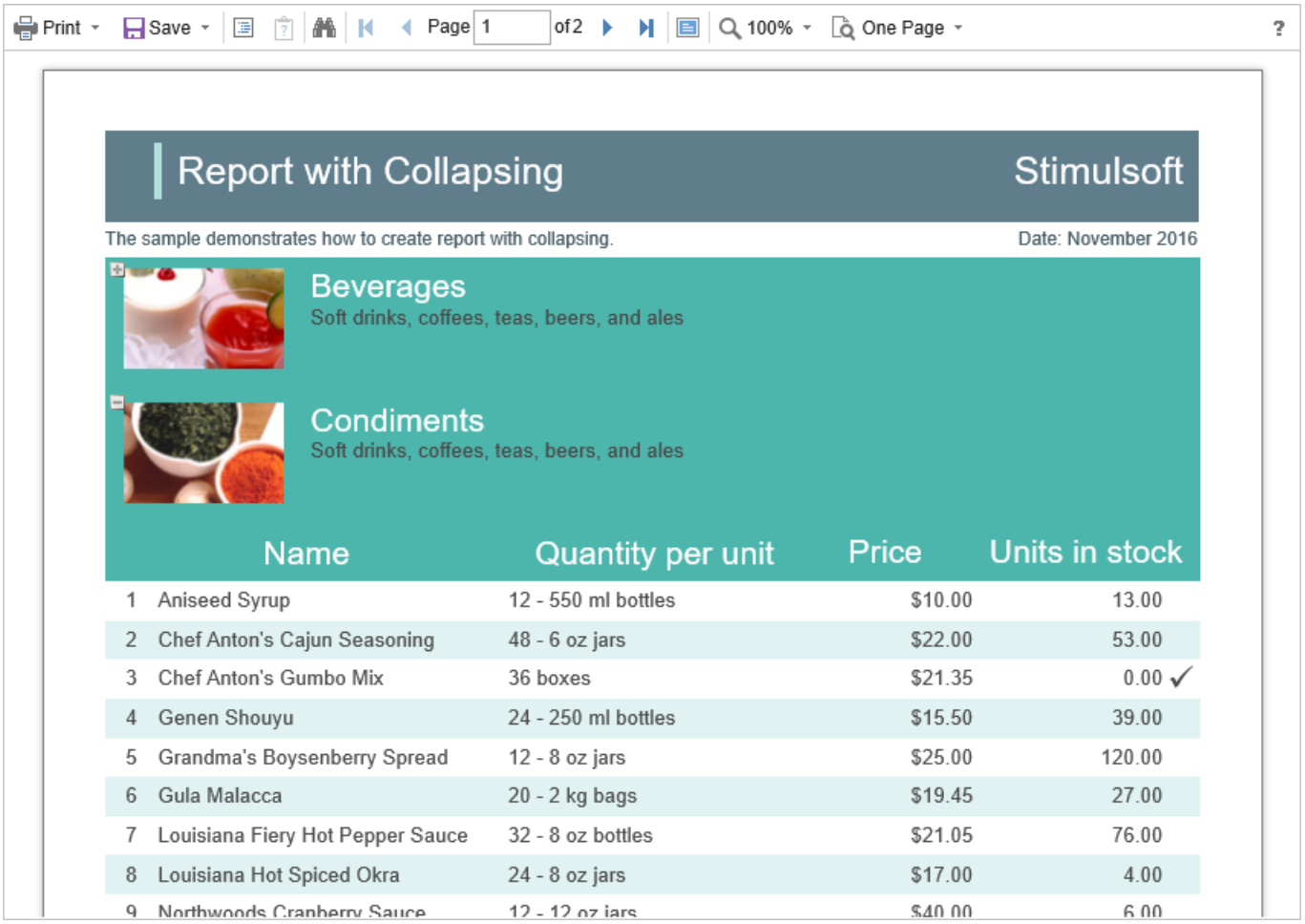
© 2003-2024 Stimulsoft



	Company	Address	Phone	Contact
1	Alfreds Futterkiste	Obere Str. 57	030-0074321	Sales Representative
2	Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222	(5) 555-4729	Owner
3	Antonio Moreno Taquería	Mataderos 2312	(5) 555-3932	Owner
4	Around the Horn	120 Hanover Sq.	(171) 555-7788	Sales Representative
5	Berglunds snabbköp	Berguvsvägen 8	0921-12 34 65	Order Administrator
6	Blauer See Delikatessen	Forsterstr. 57	0621-08460	Sales Representative
7	Blondel père et fils	24, place Kléber	88.60.15.31	Marketing Manager
8	Bólido Comidas preparadas	C/ Araquil, 67	(91) 555 22 82	Owner
9	Bon app'	12, rue des Bouchers	91.24.45.40	Owner
10	Bottom-Dollar Markets	23 Tsawwassen Blvd.	(604) 555-4729	Accounting Manager
11	B's Beverages	Fauntleroy Circus	(171) 555-1212	Sales Representative
12	Cactus Comidas para llevar	Cerrito 333	(1) 135-5555	Sales Agent
13	Centro comercial Moctezuma	Sierras de Granada 9993	(5) 555-3392	Marketing Manager
14	Chop-suey Chinese	Hauptstr. 29	0452-076545	Owner
15	Comércio Mineiro	Av. dos Lusíadas, 23	(11) 555-7647	Sales Associate

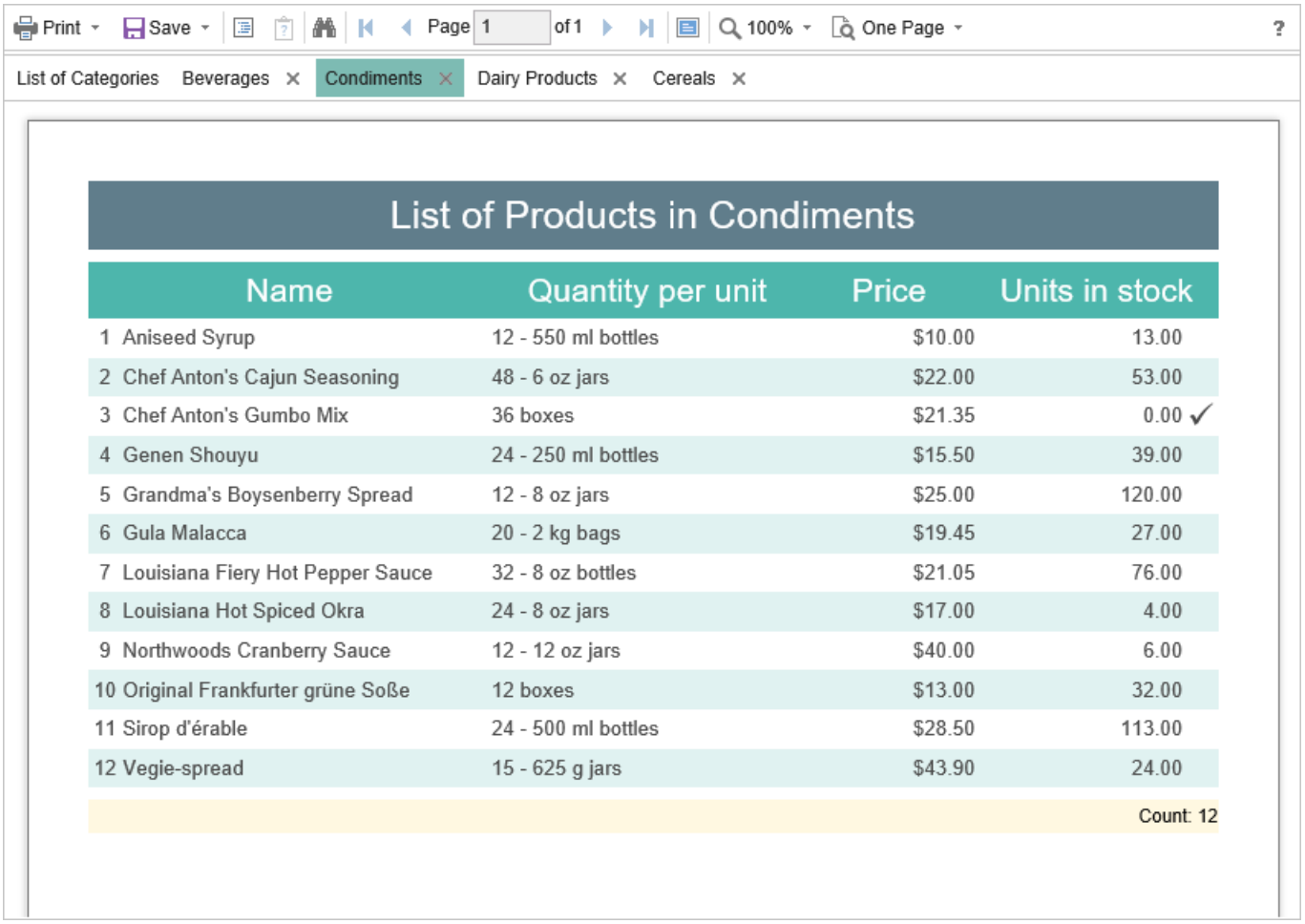
A report with dynamic collapsing is an interactive report in which blocks can collapse/expand their content when you click on the block title. Report elements, which can be collapsed/expanded, are indicated by special icons - **[-]** or **[+]**.

© 2003-2024 Stimulsoft



When using drill-down, under the main panel of the viewer, the drill-down panel with tabs for drill-down reports will be displayed. The currently displayed report will be highlighted.

© 2003-2024 Stimulsoft



To work with dynamic sorting, collapsing, and drill-down reports, no additional viewer settings are required. To perform any actions before the sorting, collapsing, or drill-down of the report, a special **OnInteraction** event is used. It will be called when interacting with the viewer. For each type of interactivity, the viewer has a certain type of action.

- **Sorting** – when using column sorting;
- **DrillDown** – when using drill-down;
- **Collapsing** – when using collapsing.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   OnInteraction="StiWebViewer1_Interaction">
</cc1:StiWebViewer>
...
```

© 2003-2024 Stimulsoft

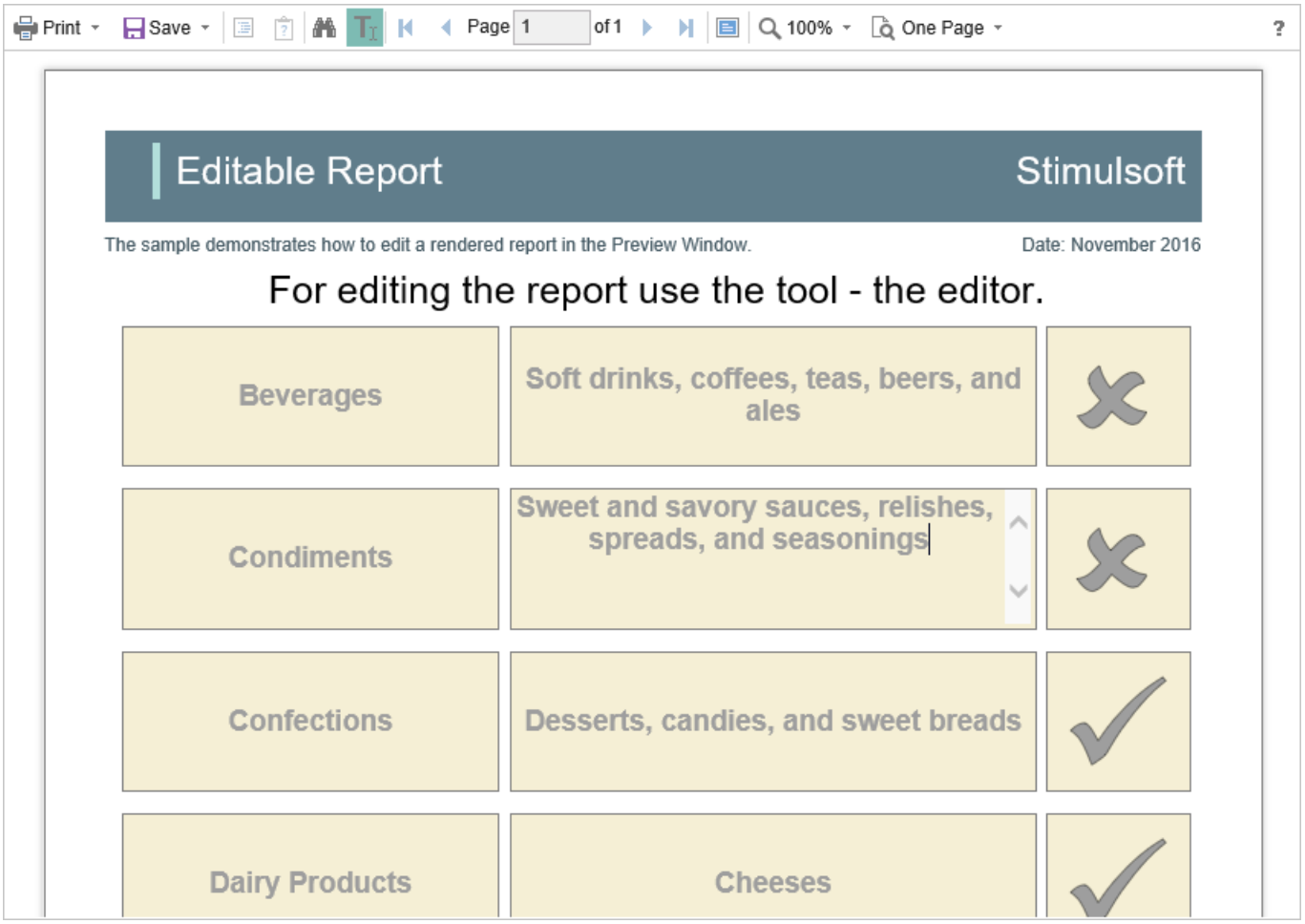
Default.aspx.cs

```
...
protected void StiWebViewer1_Interaction(object sender,
StiReportDataEventArgs e)
{
   switch (e.Action)
   {
      case StiAction.Sorting:
        break;

      case StiAction.DrillDown:
        break;

      case StiAction.Collapsing:
        break;
   }
}
...
```

3.1.14 Editing Report

The **HTML5 Viewer** component has the ability to edit report items, such as text boxes and checkboxes. You should mark the required components as editable in the report template for the editing to be possible. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be applied to the report.

© 2003-2024 Stimulsoft



For the report edit mode, no special settings of the viewer are required.

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

3.1.15 Sending Report by Email

Information

Please note that the **Send Report by Email** option is available only for reports, and not for dashboards.

© 2003-2024 Stimulsoft

The **HTML5 Viewer** component provides the ability to send reports by email. To activate this feature, you should set the **ShowSendEmailButton** property of the viewer to **true** and define the **OnEmailReport** event handler.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowSendEmailButton="true"
   OnEmailReport="StiWebViewer1_EmailReport">
</cc1:StiWebViewer>
...
```

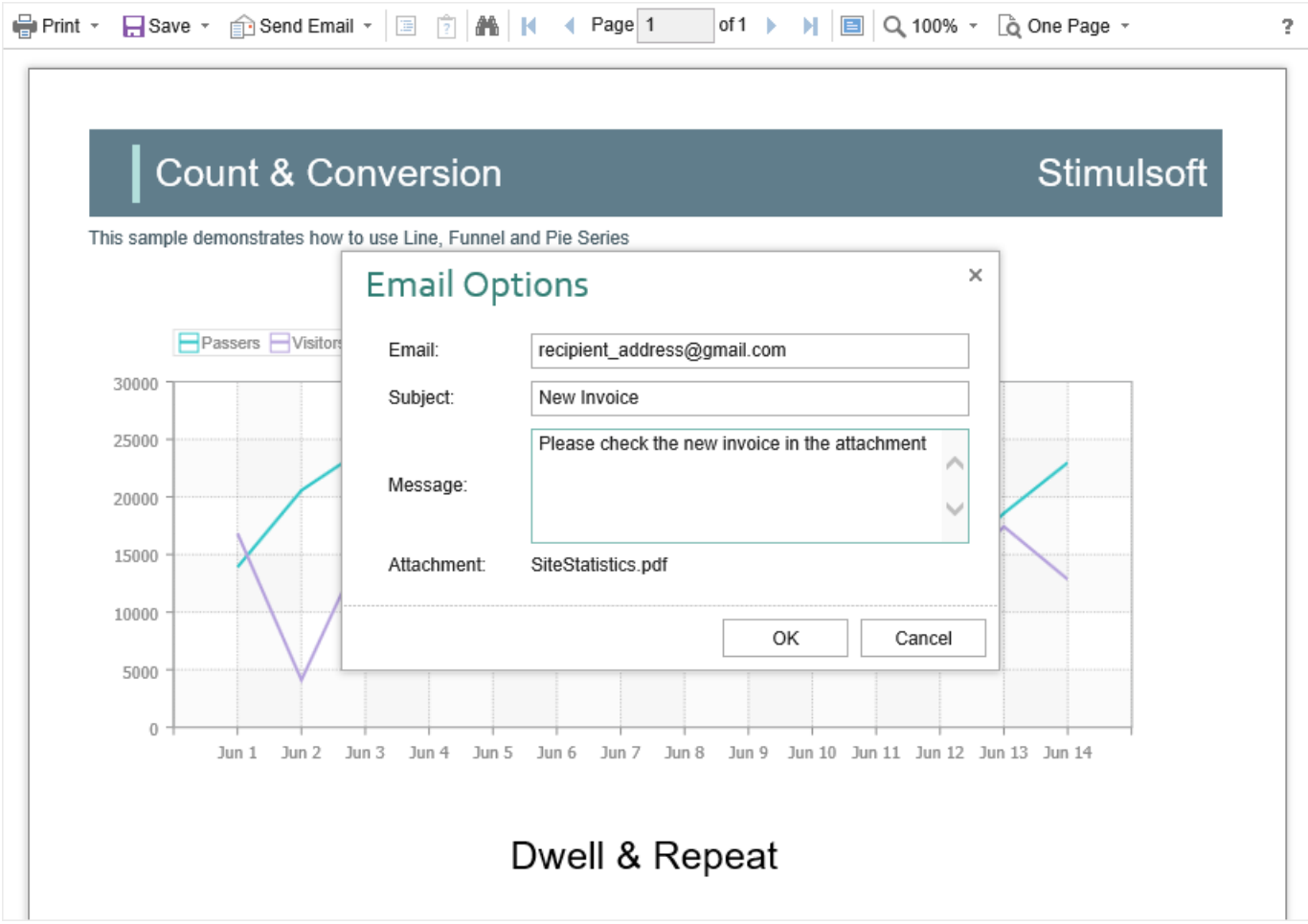
Default.aspx.cs

```
...
protected void StiWebViewer1_EmailReport(object sender,
StiEmailReportEventArgs e)
{
   StiExportFormat format = e.Format;
   StiReport report = e.Report;
   StiExportSettings settings = e.Settings;
   StiEmailOptions options = e.Options;

   // Passed from the viewer, can be checked and changed
   // options.AddressTo = "";
   // options.Subject = "";
   // options.Body = "";

   // Should be filled here
   options.AddressFrom = "admin_address@test.com";
   options.Host = "smtp.test.com";
   options.Port = 465;
   options.UserName = "admin_address@test.com";
   options.Password = "admin_password";

   // options.CC.Add("email@test.com");
   // options.BCC.Add("email@test.com");
   // options.EnableSsl = true;
}
...
```

In the **OnEmailReport** event, you can find the export type, get the report, and get the report export settings and change them, if necessary. Also, in this event, you should set email parameters, such as sender address, server name, port number, user name, and password - all these parameters will be used to send an email.

© 2003-2024 Stimulsoft

When you send a report by email, the menu to select the attachment format is displayed. This matches the menu to select an export format. After choosing the format, the dialog to put send email parameters such as email recipient, subject, and message.



After confirmation of sending the email, the above described **OnEmailReport** event will be called. You can check and correct the data entered in this form. The exported report file will be attached to the email automatically.

The **HTML5 Viewer** component allows you to set default values for the send email form. The **DefaultEmailAddress**, **DefaultEmailSubject**, and **DefaultEmailMessage** properties can be used for this. By default, these properties are empty.

Default.aspx

© 2003-2024 Stimulsoft

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   DefaultEmailAddress="recipient_address@gmail.com"
   DefaultEmailSubject="New Invoice"
   DefaultEmailMessage="Please check the new invoice in the attachment"

   ShowSendEmailButton="true"
   OnEmailReport="StiWebViewer1_EmailReport">
</cc1:StiWebViewer>
...
```

3.1.16 Calling Designer from Viewer

The **HTML5 Viewer** component has the ability to call the report designer. The special **Design** button in the toolbar of the viewer (the button is disabled by default) should be used. To use this feature, you should set the **ShowDesignButton** property to **true** and define the **OnDesignReport** event handler.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowDesignButton="true"
   OnDesignReport="StiWebViewer1_DesignReport">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_DesignReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = e.Report;
   this.Response.Redirect("Designer.aspx?report=" + report.ReportName);
}
...
```

Information

The viewer does not run the designer itself. It only calls the specified event and passes the previewed report as arguments. In the event, you can set the redirect to the ASPX page, on which the report designer is placed.

3.1.17 Caching

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to use the server cache to store rendered reports. If you do not use caching, you should load the report, connect data, and render it again every time you request a page. If you use caching, the previously rendered report will be loaded from the cache every time you refresh the page.

When using caching, it should be taken into account that every report saved in the cache takes up server memory and, with a large number of requests to reports, this can become a critical issue. Therefore, you need to choose between two options: low memory requirements but high in performance or low-performance requirements but high in memory.

You can manage caching with the following properties.

The CacheMode property

This property of the viewer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled. Each action of the viewer requires loading the report from the file and, if it is a report template, then render it;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this cache (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** – for caching, the current session is used. The report is saved as a packed string in this cache.

The CacheItemPriority property

© 2003-2024 Stimulsoft

This property sets the priority of the report stored in the server's cache. It affects the automatic clearing of the server memory in case of a lack of memory. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes for which you want to save the report in the server cache. If you use caching and the requested report is not found in the cache (the objects storage time has expired), then it will be requested again using a special **OnGetReport** event, then connect the report data and render it.

StiCacheHelper

The **HTML5 Viewer** component provides the ability to define your methods of working with report caching. For this purpose, a special class **StiCacheHelper** is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetReport** and **SaveReport**.

Default.aspx.cs

```
...
public partial class _Default : Page
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override StiReport GetReport(string guid)
      {
         string path =
         Path.Combine(HttpContext.Current.Server.MapPath(string.Empty),
         "CacheFiles", guid);
         if (File.Exists(path))
         {
            StiReport report = new StiReport();
            string packedReport = File.ReadAllText(path);
            if (guid.EndsWith("template"))
            report.LoadPackedReportFromString(packedReport);
            else report.LoadPackedDocumentFromString(packedReport);

            return report;
```

© 2003-2024 Stimulsoft

```
        }
        return null;

        //return base.GetReport(guid);
      }

      public override void SaveReport(StiReport report, string guid)
      {
        string packedReport = guid.EndsWith("template") ?
        report.SavePackedReportToString() :
        report.SavePackedDocumentToString();
        string path =
        Path.Combine(HttpContext.Current.Server.MapPath(string.Empty),
        "CacheFiles", guid);
        File.WriteAllText(path, packedReport);

        //base.SaveReport(report, guid);
      }
    }

    static _Default()
    {
      StiWebViewer.CacheHelper = new StiMyCacheHelper();
    }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as a value of the static **StiWebViewer.CacheHelper** property in the ASPX page constructor.

Information

If report caching is disabled (the **CacheMode** property of the viewer is set to **None**), the specified class will not be used.

3.1.18 Export and Printing from Code

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text. Please note that the print option is available only for reports and not for dashboards.

© 2003-2024 Stimulsoft

HTML5 Viewer is used to print reports in various ways and export reports to various formats. These actions are performed using the viewer menu. If you want to print or export a report using the code, for example, in the event of pressing the button, you can use the special **StiReportResponse** class. This class contains a set of static methods that allow you to print or export a report from the code, and the report viewer is not required.

Default.aspx

```
...
<asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Print Report" />
<asp:Button ID="Button2" runat="server" onclick="Button2_Click"
Text="Export Report" />
...
```

Default.aspx.cs

```
...
private StiReport LoadSimpleList()
{
   DataSet dataSet = new DataSet();
   dataSet.ReadXml(Server.MapPath("Reports/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));
   report.RegData(dataSet);

   return report;
}

protected void Button1_Click(object sender, EventArgs e)
{
   StiReport report = LoadSimpleList();

   StiReportResponse.PrintAsPdf(report);
   //StiReportResponse.PrintAsHtml(report);
}

protected void Button2_Click(object sender, EventArgs e)
{
   StiReport report = LoadSimpleList();

   StiReportResponse.ResponseAsPdf(report);
   //StiReportResponse.ResponseAsExcel2007(report);
   //StiReportResponse.ResponseAsText(report);
   //StiReportResponse.ResponseAsJson(report);
}
...
```

© 2003-2024 Stimulsoft

The **StiReportResponse** class contains methods for printing in PDF and HTML formats, as well as methods to export the report in any of the supported formats. As arguments, methods can take various export settings, displaying modes and options for saving received files.

3.1.19 Viewer Events

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component supports events that allow you to execute necessary operations before specific actions, such as printing and exporting, sending reports by email, interactivity, etc. Below is a sample of processing viewer events.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  OnExportReport="StiWebViewer1_ExportReport">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void StiWebViewer1_ExportReport(object sender,
StiExportReportEventArgs e)
{
  if (e.Format == StiExportFormat.Pdf)
  {
    StiPdfExportSettings pdfSettings = e.Settings as
    StiPdfExportSettings;
    pdfSettings.ImageQuality = 50;
    pdfSettings.ImageResolution = 50;
    pdfSettings.ImageCompressionMethod =
    StiPdfImageCompressionMethod.Jpeg;
  }
}
...
```

© 2003-2024 Stimulsoft

List of events

Name	Description
OnGetReport	The event occurs when requesting a report for preview.
OnGetReportData	The event occurs when connecting data of a report before it is rendered.
OnPrintReport	The event occurs when printing reports. This is not relevant when viewing dashboards.
OnExportReport	The event occurs when exporting reports.
OnExportReportResponse	The event occurs after exporting reports before saving the exported file.
OnEmailReport	The event occurs when sending a report by Email. This is not relevant when viewing dashboards.
OnInteraction	The event occurs when interactive actions of the viewer, such as using report variables, dynamic collapsing, and sorting.
OnDesignReport	The event occurs when pressing the Design button on the toolbar of the viewer.
OnViewerEvent	The event occurs when any action of the viewer.

3.1.20 Timeout

When working with the **StiWebViewer** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
  CacheTimeout="10">
</cc1:StiWebViewer>
...
```

© 2003-2024 Stimulsoft

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiWebViewer1.CacheTimeout = 10;
}
...
```

Using the cache will increase the speed of the report viewer. See the chapter Caching for more information

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   RequestTimeout="10">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiWebViewer1.RequestTimeout = 30;
}
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection and data sources in the report.

© 2003-2024 Stimulsoft

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   StiWebViewer1.Report = report;
}
...
```

3.1.21 Viewer Settings

HTML5 Viewer can be setup using the component properties which can be set on the ASPX page using C#/VB code. Below are samples of using the viewer properties.

Default.aspx

```
...
<cc1:StiWebViewer ID="StiWebViewer1" runat="server"
   ShowPrintButton="false"
   ScrollbarsMode="true"
   ShowTooltips="false"
   ShowExportToDbf="false"
   ShowExportToDif="false"
   Zoom="75"
   Theme="Office2022WhiteTeal">
</cc1:StiWebViewer>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiWebViewer1.ShowPrintButton = false;
   StiWebViewer1.ScrollbarsMode = true;
   StiWebViewer1.ShowTooltips = false;
   StiWebViewer1.ShowExportToDbf = false;
   StiWebViewer1.ShowExportToDif = false;
   StiWebViewer1.Zoom = 75;
   StiWebViewer1.Theme = StiTheme.Office2022WhiteTeal;
   StiWebViewer1.ReportDisplayMode = StiReportDisplayMode.Auto;
}
...
```

Please note that all dashboard elements have their own save options and full-screen

© 2003-2024 Stimulsoft

buttons for preview. There are no special options to control displaying them, but they can be disabled through the properties of the element. The code below should be added after loading the report before passing it to the viewer.

Default.aspx.cs

```
...
var dbsElementInteraction = (report.GetComponentByName("RegionMap1") as
Stimulsoft.Report.Dashboard.IStiElementInteraction).DashboardInteraction;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowFullScreenButton =
false;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowSaveButton = false;
...
```

Basic settings

Name	Description
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default width is 100%.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the automatic height is set depending on the size of the report page, or 650 pixels in the view mode of the viewer with scrollbars.

Work with server

Name	Description
RequestTimeout	Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For

© 2003-2024 Stimulsoft

	big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled; the report will be reloaded each time using the **OnGetReport** event; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as a repository, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of a shortage. The lower the priority is, the greater is the chance of removing information from memory.
UseRelativeUrls	Sets the viewer mode in which relative references are used for AJAX requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value that specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The

© 2003-2024 Stimulsoft

	report stored in the cache or server session will be automatically re-saved after a certain period of time if the designer is idle (every 3 minutes). By default, the property is set to **true**.
PassQueryParametersForResources	Enables transferring of all parameters of the URL request when generating links to the resources of the viewer. If it is set to **false**, only the necessary parameters are used to query the resources of the viewer, which contributes to the correct work of the browser cache. By default, the property is set to **true**.
PassQueryParametersToReport	Enables using all the URL parameters of the request as the variable values. The variables' names must match the parameters. The default value of the property is **false**.
PassFormValues	Enables passing the values of the POST form to the client-side if these values are required to be used in the events of the viewer. When the property is enabled, the helper method - **GetFormValues()** - returns a collection of parameters of a form. By default, the property is set to **false**.
ShowServerErrorPage	Enables displaying an HTML page with the details of the error that occurred on the server-side. When the property is enabled, the details of the error will be displayed in the viewer window. If the property is disabled, only the numeric error code and a short error text in the dialog box will be displayed. By default, the property is set to **true**.
UseCompression	Enables compression of the viewer requests into the GZip stream. That allows to decrease the amount of internet traffic but slows down the viewer slightly. The default value of the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server-side. The following resources are

© 2003-2024 Stimulsoft

	supported: scripts, styles, and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
UseLocalizedCache	Sets a value that enables the use of a different cache depending on the selected localization. The default value of the property is **false**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
Theme	Specifies the theme of the viewers' layout. The list of available themes can be found in the **StiTheme** enumeration. The default value is **Office2022WhiteBlue**.
CustomCss	Sets the path to the CSS file of the viewer's styles. The standard styles of the chosen theme will not be loaded if this property has got a value. The default value of the property is an empty string.
Localization	Specifies the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used, which is built into the viewer and does not require additional XML files.
BackgroundColor	Sets the background color of the viewer. By default, it is set to **White**.
PageBorderColor	Sets the border color of the viewer. By default, it is set to **Gray**.
RightToLeft	Sets the **Right to Left** mode for viewer controls.

© 2003-2024 Stimulsoft

	By default, the property is set to **false**.
FullScreenMode	Sets the full-screen display mode of the viewer. By default, the property is set to **false**.
ScrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to **false**.
OpenLinksWindow	Sets the target window for opening links contained in the report. By default, it is set to **Blank** (new window).
OpenExportedReportWindow	Sets the target window for opening the export file from the viewer. By default, it is set to **Blank** (new window).
DesignWindow	Sets the target window for opening the report designer. By default, it is set to **Self** (current window).
ShowTooltips	Enables displaying tips for the viewer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enables displaying links to online documentation for the viewer controls. By default, the property is set to **true**.
ShowDialogsHelp	Sets a value which indicates that show or hide the help button in dialogs. By default, the property is set to **true**.
PageAlignment	Sets the position of the report page in the viewer window. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – the page will be aligned left; › **Center** – the page will be centered (default value); › **Right** – the page will be aligned right.
ShowPageShadow	Enables displaying shadow for report pages. By default, the property is set to **true**.
BookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default, the property is set

© 2003-2024 Stimulsoft

	to **false**.
BookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
ParametersPanelPosition	Specifies the position of the report parameters panel. It can take one of the following **StiParametersPanelPosition** enumeration values: › **Top** - the panel will be docked to the top margin (default value); › **Left** – the panel will be docked to the left margin.
ParametersPanelMaxHeight	Sets the maximum height of the parameters bar in pixels. By default, the maximum height is 300 pixels.
ParametersPanelColumnsCount	Sets the number of columns to display report parameters. By default, there are 2 columns.
ParametersPanelSortDataItems	Gets or sets a value which indicates that variable items will be sorted. By default, the property is set to **true**.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
InterfaceType	Sets the type of interface used for the viewer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the viewer's interface is determined automatically depending of the device that is report is displayed on. That is the default value. › **Mouse** – the standard interface with a mouse control will be used for all the screen types. › **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been

© 2003-2024 Stimulsoft

	increased in size to simplify the control of the viewer and to improve its usability. › **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.
AllowMobileMode	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to **false**, then the mobile view will not be used. If the option is set to **true**, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option is set to **true**.
ChartRenderType	Sets the chart displaying mode on the report page. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages. It can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN

© 2003-2024 Stimulsoft

	markup.
DatePickerFirstDayOfWeek	Sets the first day of the week for the date picker. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Monday** – the first day of the week is Monday (default value); › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
AllowTouchZoom	Sets ability to change the scale of the report page by using the two-fingers gesture (Pinch to Zoom) for the touch-screens. The default value of the property is **true**.
ShowReportIsNotSpecifiedMessage	Sets a value which indicates that 'The report is not specified' message will be shown. The default value of the property is **true**.
ImagesQuality	Gets or sets the image quality that will be used on the viewer page. It has the following values: › **StiImagesQuality.Low** - low quality, used to speed up loading reports and saves memory; › **StiImagesQuality.Normal** - normal quality, suitable for most cases (default value); › **StiImagesQuality.High** - high quality, used for ultra high-definition displays, but may slow down the loading of pages.
PrintToPdfMode	Sets the Print to PDF mode. It has the following values: › **StiPrintToPdfMode.Hidden** - hidden print mode (default value); › **StiPrintToPdfMode.Popup** - the PDF document will be displayed before printing in a pop-up window.
CombineReportPages	Sets a value which indicates that if a report contains several pages, then they will be combined in preview. By default, the property is

© 2003-2024 Stimulsoft

	set to **false**.
SaveMenuImageSize	Sets a value which indicates images size of the save menu in the viewer. The default value of the property is **true**.

Toolbar

Name	Description
ShowToolbar	Enables displaying the viewer toolbar. By default, the property is set to **true**.
ToolbarDisplayMode	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the **StiToolbarDisplayMode** enumeration values: › **Simple** - all controls are located on the same control panel (default value); › **Separated** - the control panel is split into top and bottom panels.
ToolbarBackgroundColor	Sets the color of the viewer toolbar. By default, color of the selected theme is used.
ToolbarBorderColor	Sets the color of the borders of the Viewer toolbar. By default, color of the selected theme is used.
ToolbarFontColor	Sets the text color for the toolbar and the viewer menu. By default, color of the selected theme is used.
ToolbarFontFamily	Sets the font for the toolbar and the viewer menu. The default font of the selected theme is used.
ToolbarAlignment	Sets the alignment mode for the controls on the viewer toolbar. It can take one of the following values of the **StiContentAlignment** enumeration:

© 2003-2024 Stimulsoft

	› **Left** – elements will be aligned left; › **Center** – elements will be centered; › **Right** – elements will be aligned right; › **Default** – the alignment depends on the RightToLeft property (default value).
ShowButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default, the property is set to **true**.
ShowPrintButton	Enables showing the button - **Print** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports or dashboards.. By default, the property is set to **true**.
ShowSendEmailButton	Enables showing the button - **Send Email** - on the toolbar of the viewer. By default, the property is set to **false**. Also, you should add the OnEmailReport event handler.
ShowFindButton	Enables showing the button - **Find** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowBookmarksButton	Enables showing the button - **Bookmarks** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
ShowParametersButton	Enables showing the button - **Parameters** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.
ShowResourcesButton	Enables showing the button - **Resources** - on

© 2003-2024 Stimulsoft

	the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.
ShowEditorButton	Enables showing the button - **Editor** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowFullScreenButton	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports or dashboards.. By default, the property is set to **true**.
ShowFirstPageButton	Enables showing the button - **First Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowPreviousPageButton	Enables showing the button - **Previous Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowCurrentPageControl	Enables showing the current report page indicator. By default, the property is set to **true**.
ShowNextPageButton	Enables showing the button - **Next Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowLastPageButton	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowZoomButton	Enables showing the report zoom button. By default, the property is set to **true**.
ShowViewModeButton	Enables showing the button for selecting the display mode of report pages. By default, the property is set to **true**.
ShowDesignButton	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports or dashboards.. By default, the property is set to **false**.
ShowAboutButton	Enables showing the button - **About** - on the

© 2003-2024 Stimulsoft

	toolbar of the viewer. By default, the property is set to **true**.
ShowRefreshButton	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to **true**.
ShowPinToolbarButton	Enables displaying of the **Pin Toolbar** button on the viewer's toolbar. The button is available only in the Mobile mode of the viewer's interface. The default value of the property is **true**.
PrintDestination	Sets the report printing mode. It can take one of the following values of the **StiPrintDestination** enumeration: › **Default** – a menu with a choice of printing modes will be displayed (default value); › **Pdf** – printing will be done in the PDF format; › **Direct** – printing will be done to the HTML format directly to the printer, the system print dialog will be displayed; › **PopupWindow** – printing will be done in the HTML format via the preview window of the report.
ViewMode	Sets the mode for displaying report pages. It can take one of the following values of the **StiWebViewMode** enumeration values: › **SinglePage** - displays one page of the report selected in the toolbar of the viewer (default value); › **Continuous** - displays all pages of the report; › **MultiplePages** - displays all report pages as a table.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of the following values:

© 2003-2024 Stimulsoft

	› **StiZoomMode.PageWidth** – when the viewer runs, the zoom, necessary to display the report by the page width, will be set; › **StiZoomMode.PageHeight** – when the viewer runs, the zoom, required to display the page height of the report, will be set.
MenuAnimation	Enables animation when the viewer menu shows/hides. By default, the property is set to **true**.
ShowMenuMode	Sets the display mode of the viewer menu. It can take one of the following values of the **StiShowMenuMode** enumeration: › **Click** – shows menu by mouse click (default value); › **Hover** – shows menu by hovering the mouse cursor.
AutoHideToolbar	Enables auto-hiding of the viewer's toolbar. The property will work only for the Mobile mode of the viewer's interface. The default value of the property is **false**.

Samples

See samples how to add a custom button on the toolbar of the viewer.

Export report

Name	Description
DefaultExportSettings	This group of properties is used to specify the default export settings for each export type. These settings will be applied to the export dialogs when the viewer runs or to the report, if export dialogs are disabled.

© 2003-2024 Stimulsoft

StoreExportSettings	Enables saving selected settings in the export dialogs. Settings will be stored in browser cookies. By default, the property is set to **true**.
ShowExportDialog	Enables showing export options dialog box. If the property is set to **false**, the export will be done with the default settings. By default, the property is set to **true**.
ShowExportToDocument	Enables the export menu item - **Document File**. By default, the property is set to **true**.
ShowExportToPdf	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToXps	Enables the export menu item - **Microsoft XPS File**. By default, the property is set to **false**.
ShowExportToPowerPoint	Enables the export menu item - **Microsoft PowerPoint 2007/2010 File**. By default, the property is set to **true**.
ShowExportToHtml	Enables the export menu item - **File**. By default, the property is set to **true**.
ShowExportToHtml5	Enables the export menu item - **HTML5 File**. By default, the property is set to **true**.
ShowExportToMht	Enables the export menu item - **MHT Web Archive**. By default, the property is set to **true**.
ShowExportToText	Enables the export menu item - **Text File**. By default, the property is set to **true**.
ShowExportToRtf	Enables the export menu item - **Rich Text File**. By default, the property is set to **true**.
ShowExportToWord2007	Enables the export menu item - **Microsoft Word 2007/2010 File**. By default, the property is set to **true**.
ShowExportToOpenDocumentWriter	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToExcel	Enables the export menu item **Microsoft Excel File**. By default, the property is set to **true**.
ShowExportToExcelXml	Enables the export menu item - **Microsoft Excel Xml File**. By default, the property is set to **true**.
ShowExportToExcel2007	Enables displaying the **Microsoft Excel 2007/2010 File** export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToOpenDocumentCalc	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to **true**.
ShowExportToCsv	Enables the export menu item - **CSV File**. By default, the property is set to **true**.
ShowExportToDbf	Enables the export menu item - **DBF File**. By default, the property is set to **true**.
ShowExportToXml	Enables the export menu item - **XML File**. By default, the property is set to **true**.
ShowExportToDif	Enables the export menu item - **Data Interchange Format (DIF) File**. By default, the property is set to **true**.
ShowExportToSylk	Enables the export menu item - **Symbolic Link (SYLK) File**. By default, the property is set to **true**.
ShowExportToJson	Enables the export menu item - **JSON File**. By default, the property is set to **true**.
ShowExportToImageBmp	Enables displaying the **BMP Image** export menu item when viewing reports, and the **BMP Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageGif	Enables displaying the **GIF Image** export menu item when viewing reports, and the **GIF Image** item when viewing dashboards. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToImageJpeg	Enables displaying the **JPEG Image** export menu item when viewing reports, and the **JPEG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePcx	Enables displaying the **PCX Image** export menu item when viewing reports, and the **PCX Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePng	Enables displaying the **PNG Image** export menu item when viewing reports, and the **PNG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageTiff	Enables displaying the **TIFF Image** export menu item when viewing reports, and the **TIFF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageMetafile	Enables displaying the **Windows Metafile** export menu item when viewing reports, and the **Windows Metafile** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvg	Enables displaying the **Scalable Vector Graphics (SVG) File** export menu item when viewing reports, and the **Scalable Vector Graphics (SVG) File** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvgz	Enables displaying the **Compressed SVG (SVGZ) File** export menu item when viewing reports, and the **Compressed SVG (SVGZ) File** item when viewing dashboards. By default, the property is set to **true**.
ShowOpenAfterExport	Enables displaying the **Open After Export** parameter in export settings menu. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

Email

Name	Description
ShowEmailDialog	Enables displaying settings for sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server-side in the **OnEmailReport** event. By default, the property is set to **true**.
ShowEmailExportDialog	Enables displaying export options dialog box when sending email. If the property is set to **false**, the export will be done with the default settings. By default, the property is set to **true**.
DefaultEmailAddress	Sets the default recipient email, i.e. the address to which the email with the attached report will be sent.
DefaultEmailSubject	Sets the default email subject (header).
DefaultEmailMessage	Sets the default email message (text).

3.2 HTML5 Designer

YouTube

Watch videos for working with ASP.NET HTML5 Designer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

Samples

See on GitHub examples of working with the ASP.NET HTML5 Designer component. All samples are separate projects grouped into one solution for Visual Studio.

The **HTML5 Designer (StiWebDesigner)** component is designed to create reports in the web browser. You do not need to install the .NET Framework, ActiveX components, or any special plug-ins on the client-side. All you need is any modern

© 2003-2024 Stimulsoft

Web browser.

With the help of **HTML5 Designer**, you can create, edit, save and preview reports on any computer with any operating system installed. Since the designer only uses HTML and JavaScript technologies, it can be run on devices without Flash or Silverlight support - tablets, smartphones. Also, the designer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Designer** component uses the AJAX technology to perform all actions on reports, which allows you to get rid of reloading the entire page, save Web traffic, and speed up work.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use **HTML5 Designer** in a Web-project, the installation of the Stimulsoft.Reports.Web NuGet package is required:
- Select 'Manage NuGet Packages...' menu item in the project's pop-up menu;
- In the 'Browse' tab, type 'Stimulsoft.Reports.Web' in the search textbox;
- Click the Stimulsoft.Report.Web package, select the version of the package, and click **Install**. If the package should be updated, use the **Update** button.

If for some reason, that is not possible, the following assemblies should be added to the project:
- Stimulsoft.Base.dll
- Stimulsoft.Report.dll
- Stimulsoft.Report.Check.dll
- Stimulsoft.Report.Helper.dll
- Stimulsoft.Report.Web.dll
- Stimulsoft.Report.WebDesign.dll

To add the ability to create and edit dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web (this package is associated with the package

© 2003-2024 Stimulsoft

Stimulsoft.Reports.Web. If it is missed, it will be installed automatically):
› Select "Manage NuGet Packages ..." in the context menu of the project;
› Specify Stimulsoft.Dashboards.Web in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

If for some reason this is not possible, you should additionally add the following assemblies to the project:
› Stimulsoft.Dashboard.dll
› Stimulsoft.Dashboard.Drawing.dll
› Stimulsoft.Dashboard.Export.dll

- How this Works?
- Activation
- Editing Reports and Dashboards
- Creating New Reports and New Dashboards
- Saving Report and Dashboard
- Preview
- Settings
- Additional Functionality of Preview
- Timeout
- Localization
- Using Themes
- Caching
- Events

3.2.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the designer, you need to place the **StiWebDesigner** component on the ASPX page, set the necessary properties, and, if necessary, set the necessary event handlers. When the report designer runs, the following actions occur:

© 2003-2024 Stimulsoft

› The .NET component generates HTML and JavaScript code that is necessary for displaying and running the designer;
› When the component is output, the JavaScript method is launched. It requests the report template on the server side displays it in the designer window;
› Various actions in the designer (for example, report preview, saving the report template, export reports, sorting, drill-down, etc.) call a particular action on the server-side. You can perform the necessary manipulations with the report.

3.2.2 Activation

YouTube

Watch videos that show how to activate the ASP.NET HTML5 Designer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiWebDesigner** component.

Default.aspx.cs

```
...
public partial class _Default : Page
{
   static _Default()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
      var path = HttpContext.Current.Server.MapPath("license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

© 2003-2024 Stimulsoft

3.2.3 Editing Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To edit a report template, you need to add the **StiWebDesigner** component to the ASPX page and assign a loaded report template to it.

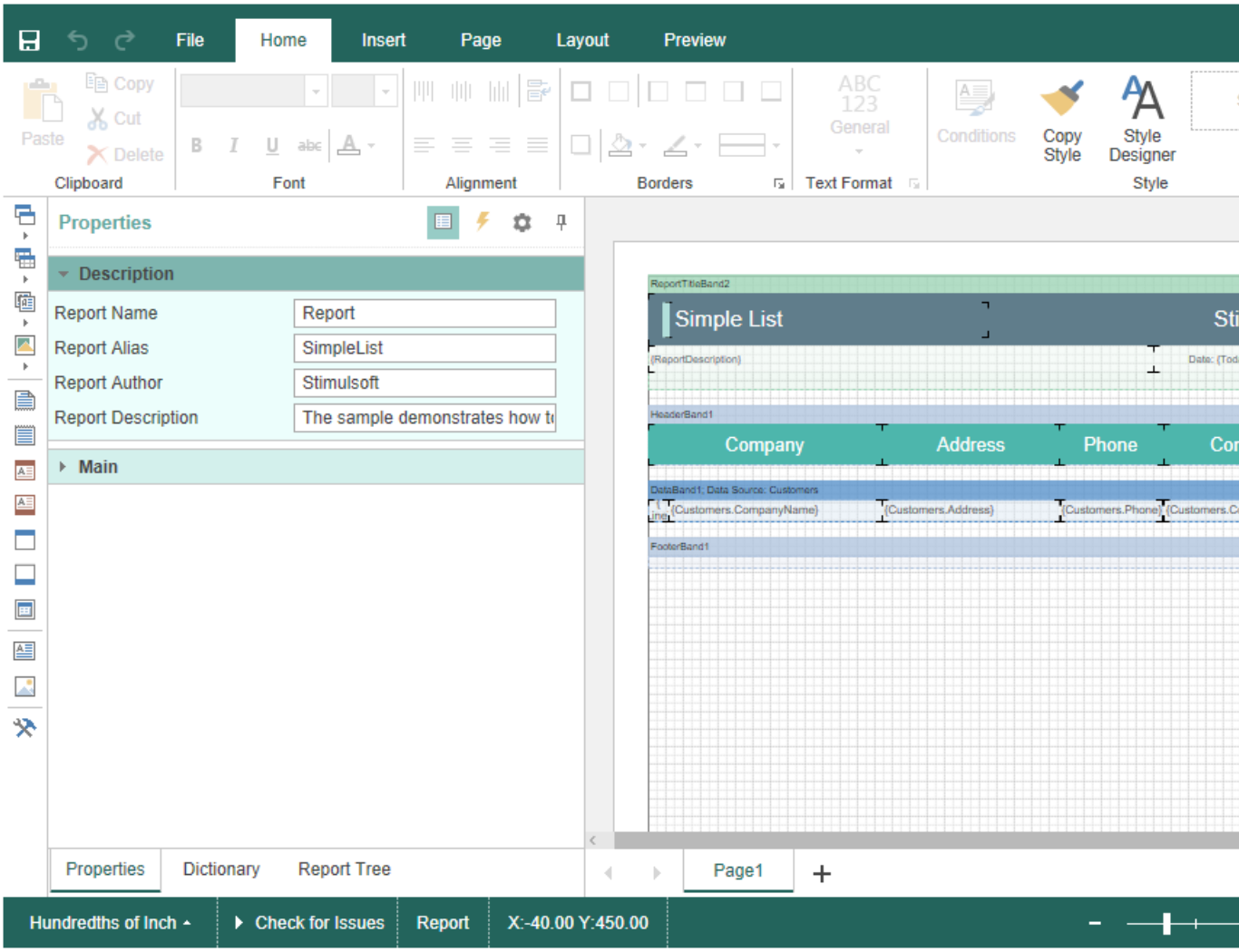
Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));
   //report.Load(Server.MapPath("Reports/Dashboard.mrt"));

   StiWebDesigner1.Report = report;
}
...
```

© 2003-2024 Stimulsoft



Also, **HTML5 Designer** has a special **OnGetReport** event that you can use to assign a report template. In this case, you need to load the report in the event handler.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
  OnGetReport="StiWebDesigner1_GetReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_GetReport(object sender,
StiReportDataEventArgs e)
```

© 2003-2024 Stimulsoft

```
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));

   e.Report = report;
}
...
```

Information

The **OnGetReport** event will be called regardless of whether the report was previously assigned or not. If the report is already assigned to the designer, then, in the event arguments, the **e.Report** property will contain the loaded report object. You can change it or assign a new report.

By default, **HTML5 Designer** uses the entire area of the browser window to edit the report. To display a component in a specific HTML page with the specific position and dimensions, it is enough to set its width and height using the **Width** and **Height** properties.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   Width="1000px" Height="800px">
</cc1:StiWebDesigner>
...
```

3.2.4 Creating New Reports and New Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the designer with a new (empty) report, no action is required. When the component is loaded, the new report will be created automatically. If necessary, you can create a new report object, preload the data, or perform other necessary actions.

© 2003-2024 Stimulsoft

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnGetReport="StiWebDesigner1_GetReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_GetReport(object sender,
StiReportDataEventArgs e)
{
   e.Report = new StiReport();
   //var newDashboard = StiReport.CreateNewDashboard();
}
...
```

You can also create a new report using the main menu of the designer. The **OnCreateReport** event is used to load data for a new report or perform other necessary actions. This event will be called when you create a new empty report or a report using the wizard.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnCreateReport="StiWebDesigner1_CreateReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_CreateReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = new StiReport();
   //var newDashboard = StiReport.CreateNewDashboard();

   // Register data for the new report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("Data/Demo.xml"));
   report.RegData(data);
   //newDashboard.RegData(data);
   report.Dictionary.Synchronize();
   //newDashboard.Dictionary.Synchronize();
```

© 2003-2024 Stimulsoft

```
  e.Report = report;
  //e.Report = newDashboard;
}
...
```

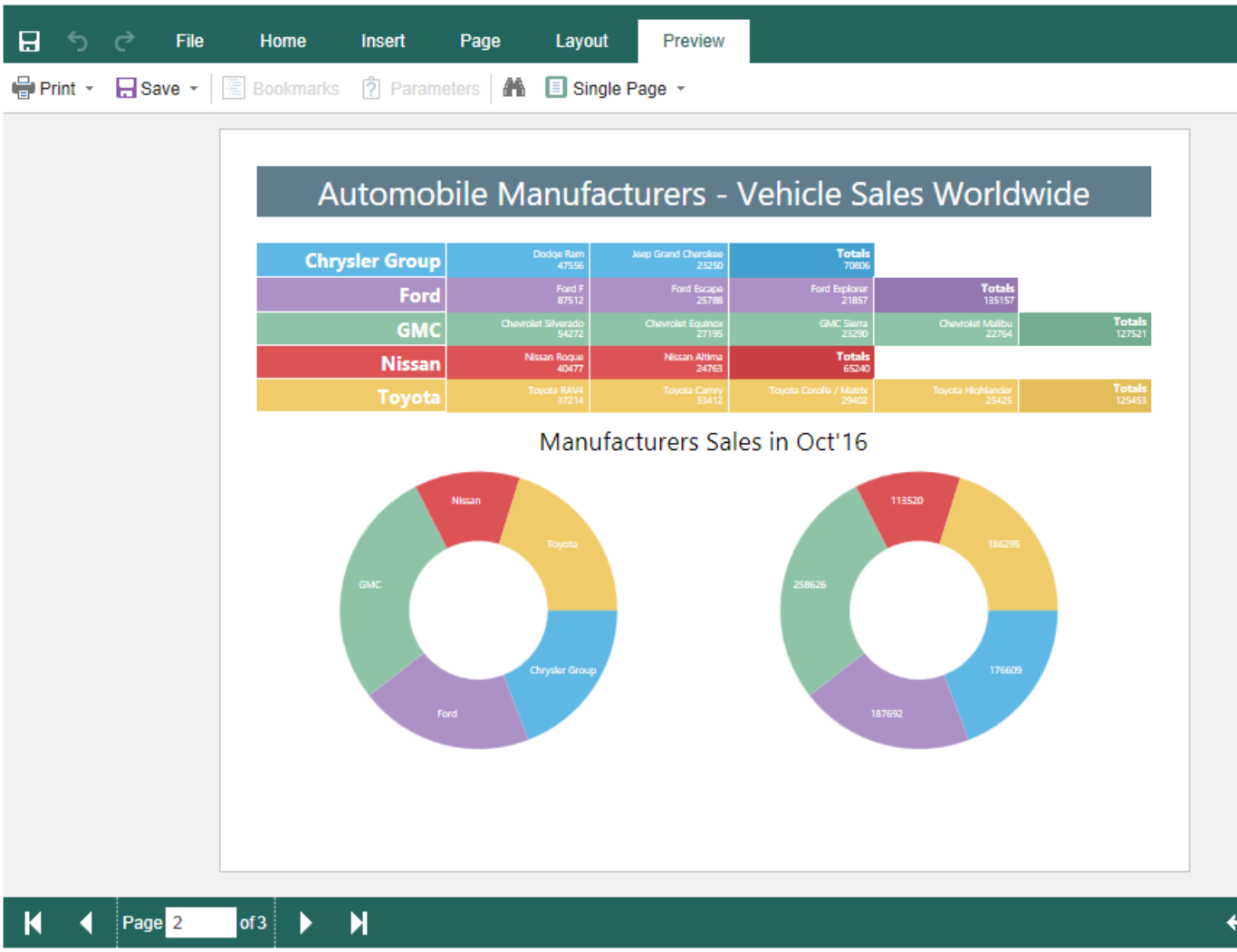
3.2.5 Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides the ability to preview reports. To preview the report, just go to the appropriate tab in the designer window. The report template will be transferred to the server-side, rendered, and displayed in the embedded viewer.

© 2003-2024 Stimulsoft



Before previewing the report, it is possible to perform any necessary actions, for example, connect data for the report. To do this, you can use a special **OnPreviewReport** event which will be called before previewing the report. In the arguments of the event, there will be a report to be previewed. The **OnPreviewReport** event is called before preparing and rendering a report for viewing till its saving to the cash.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
  OnPreviewReport="StiWebDesigner1_PreviewReport">
</cc1:StiWebDesigner>
...
```

© 2003-2024 Stimulsoft

Default.aspx.cs

```
...
protected void StiWebDesigner1_PreviewReport(object sender,
StiReportDataEventArgs e)
{
   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("Data/Demo.xml"));
   e.Report.RegData(data);
}
...
```

If you need to take actions on your report immediately before displaying the report, you can use the **OnGetPreviewReport** event, which is called after the request of the prepared report from the cash.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnGetPreviewReport="StiWebDesigner1_GetPreviewReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_GetPreviewReport(object sender,
StiReportDataEventArgs e)
{
   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("Data/Demo.xml"));
   e.Report.RegData(data);

   //report.IsRendered = false;
}
...
```

Information

So as in this event, a prepared report for viewing is transferred. If you need to render again, you should set the **report.IsRendered = false** flag.

3.2.6 Additional Features of Preview

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The preview window of the **HTML5 Designer** component has a fully functional interactive HTML5 Viewer that can print and export reports, supports working with report parameters, dynamic sorting, interactive reports, collapsing, etc. To use these features, you do not need any additional settings for the report designer.

In any of the above actions, you can work with the report template, such as changing its properties and parameters and connecting new data for rendering.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnExportReport="StiWebDesigner1_ExportReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_ExportReport(object sender,
StiReportDataEventArgs e)
{
   e.Report.ReportName = "MyReportName";
   e.Report.ReportAlias = "MyReportAlias";
}
...
```

Information

Suppose you do not need any additional options to preview the report (for example, exporting or printing a report). In that case, you can disable them using the appropriate properties of the **HTML5 Designer** component.

3.2.7 Saving Reports and Dashboards

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides two ways of saving the report, which are available in the main menu and the main panel of the designer - **Save Report** and **Save As**. In turn, each of these ways has its modes and settings.

Saving a report and dashboard on the server-side

To save the edited report on the server-side, you need to set the **OnSaveReport** special event, which will be called when you select the **Save Report** menu item or click the Save button on the main panel of the designer.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnSaveReport="StiWebDesigner1_SaveReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_SaveReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = e.Report;

   // Save the report template
   // ...
}
...
```

By default, after saving the report, the designer continues working without displaying any messages. If necessary, after saving the report, it is possible to display a dialog box with an error or a text message. For this purpose, the special properties - **e.ErrorCode** and **e.ErrorString** in the arguments of the event are used.

© 2003-2024 Stimulsoft

Default.aspx.cs

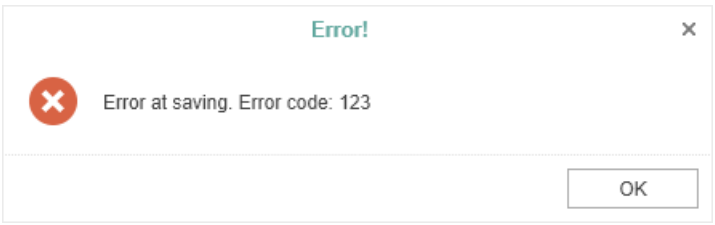
```
...
protected void StiWebDesigner1_SaveReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = e.Report;

   // Save the report template
   // ...

   e.ErrorCode = 123;
   //e.ErrorString = "Some message after saving";
}
...
```

You can get a report name from the designer save dialog or an original report name.

Default.aspx.cs

```
...
protected void StiWebDesigner1_SaveReport(object sender,
StiSaveReportEventArgs e)
{
   //Report name from the designer save dialog
   var reportName = e.FileName;

   //Original report name from properties
   var reportName = e.Report.ReportName;
}
...
```

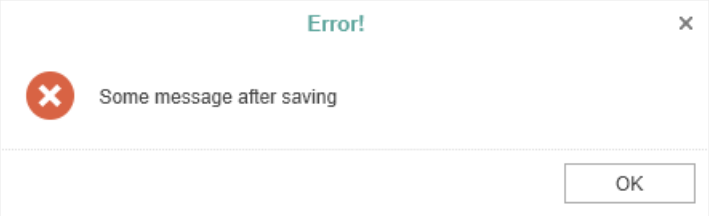
If you set a positive integer value for the **e.ErrorCode** property, the user will see the error message of saving the report and the error code, where the error code is the integer value.



If you set a string value for the **e.ErrorString** property, a dialog with the specified

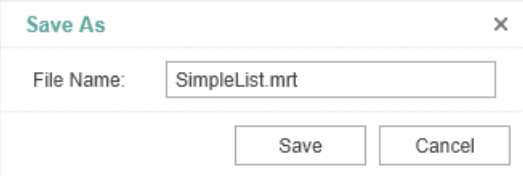
© 2003-2024 Stimulsoft

text will be displayed. The text can contain both a save error message or a warning, or any other message.



Saving reports and dashboards on the client side

To save the edited report on the client-side as a file, no additional designer settings are required. It is enough to click the **Save As** main menu item. The dialog box will be displayed. In this dialog, you can change the name of the report file. The file will be saved to the local disk of the computer.



The **HTML5 Designer** component provides the ability to change the behavior of the specified save option. For this purpose, the special **OnSaveReportAs** event is used in the designer. If you use this event, the report will be saved on the server-side. The work of this event will be similar to the **OnSaveReport** event.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnSaveReportAs="StiWebDesigner1_SaveReportAs">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_SaveReportAs(object sender,
StiReportDataEventArgs e)
{
   StiReport report = e.Report;
```

```
    // Save the report template
    // ...
}
...
```

Saving settings

The report is saved in the background mode without reloading the page in the web browser window. Suppose you need to control the process of saving the report visually. In that case, you should change the value of the **SaveReportMode** (or **SaveReportAsMode**) property of the designer to one of the three specified values - **Hidden** (default value), **Visible**, or **NewWindow**.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnSaveReportAs="StiWebDesigner1_SaveReportAs"
   SaveReportAsMode="Visible">
</cc1:StiWebDesigner>
...
```

If the **SaveReportMode** property is set to **Visible**, the report save event will be called in the current browser window in the normal (visible) mode using the POST request. If the **SaveReportMode** property is set to **NewWindow**, the report save event will be called in a new window of the web browser. By default, this property is set to **Hidden** - the report save event is called in the background using the AJAX request and is not displayed in the browser window. The same values and behavior are applicable to the **SaveReportAsMode** property.

3.2.8 Localization

The **HTML5 Designer** component supports the complete localization of its interface. Use the special **Localization** property to localize the report designer interface. As a value of this property, you should specify the path to the localization XML file (relative or absolute).

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   Localization="Localization/en.xml">
</cc1:StiWebDesigner>
```

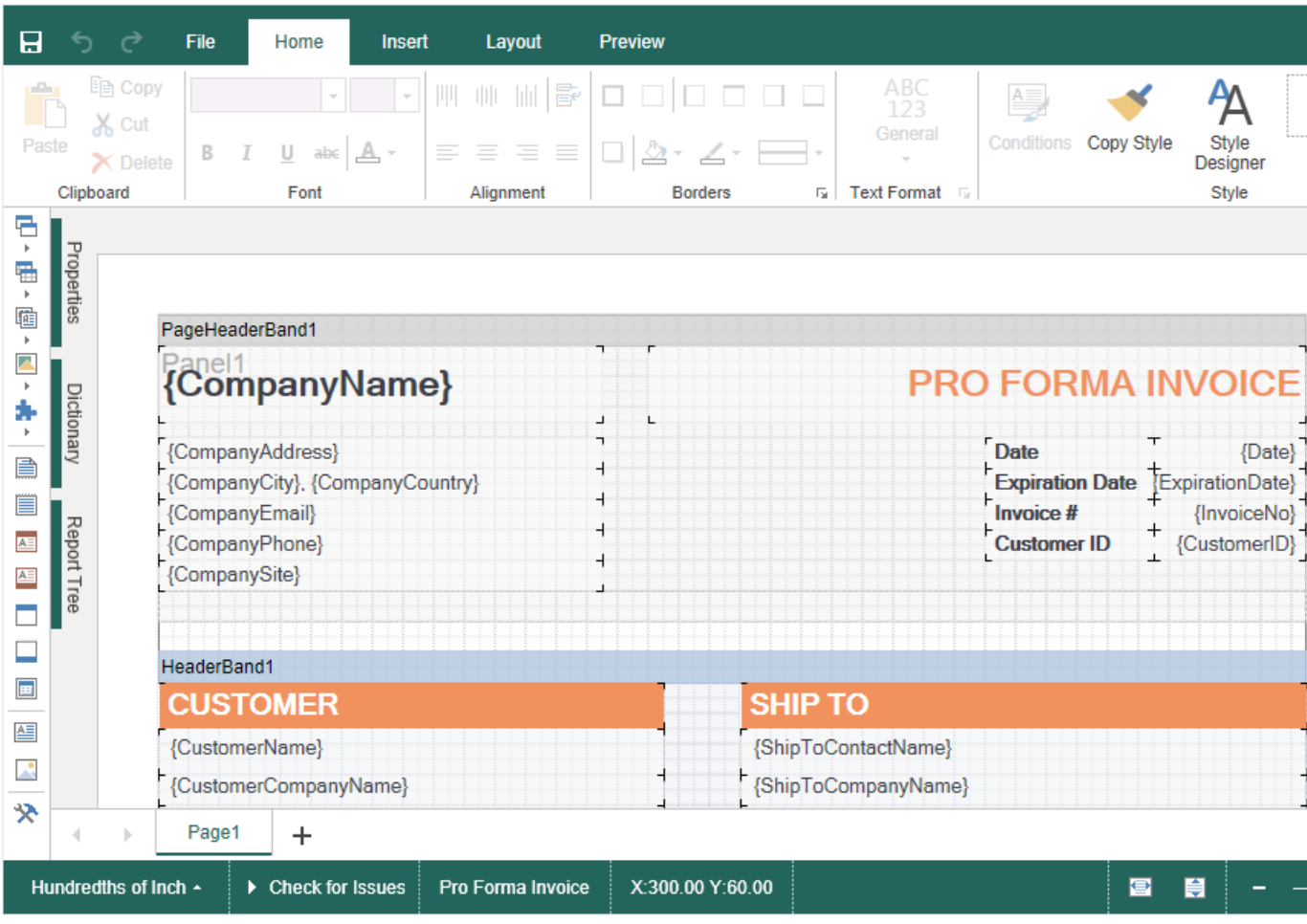
© 2003-2024 Stimulsoft

```
...
```

The interface of the report designer allows you to select the necessary localization from an accessible list. To do this, the value of the **LocalizationDirectory** property must be the folder in which the localization XML files are stored.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   Localization="Localization/en.xml"
   LocalizationDirectory="Localization">
</cc1:StiWebDesigner>
...
```



Information

© 2003-2024 Stimulsoft

If the value for the **Localization** property is set, then when you run the report designer, the localization specified in this property will always be applied. If the property value is not set, the localization selected from the list of available localizations in the report designer panel will be automatically loaded.

3.2.9 Using Themes

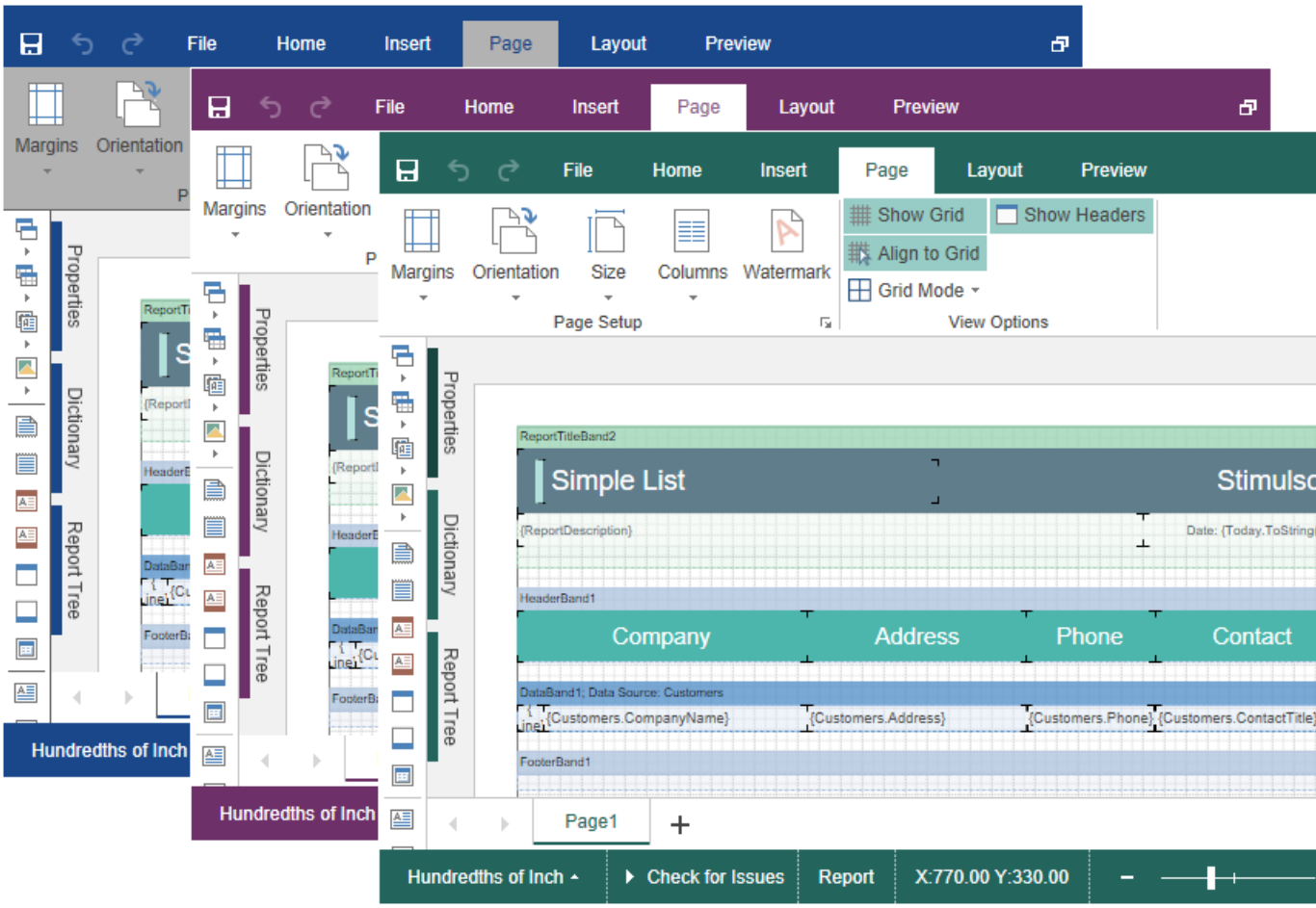
In the **HTML5 Designer** component, you can change the appearance of visual controls. To change the theme, you should use the **Theme** property.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
  Theme="Office2022WhiteTeal">
</cc1:StiWebDesigner>
...
```

There are currently **2 themes** available with different color accents. As a result, **more than 50** variants of the appearance are available. This allows you to customize the appearance of the designer for almost any design of the Web project.

© 2003-2024 Stimulsoft



3.2.10 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

HTM5 Designer uses the server cache to store the editable report template. It is necessary because the client part of the designer contains only a visual representation of components of a report template. The report object itself with all the parameters and properties is stored on the server-side.

You can manage caching with the following properties.

© 2003-2024 Stimulsoft

The CacheMode property

This property of the designer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** - for caching, the current session is used, in which the report is saved as a packed string.

The CacheItemPriority property

This property sets the priority of the report stored in the cache of the server. It affects the automatic clearing of the server memory in case of memory shortage. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes you want to store the report in the server cache. If, when using caching, the requested report is not found in the cache (time of storing this report expired), then it will be requested again using the special **OnGetReport** event. In this case, the unsaved changes may be lost.

The **HTML5 Designer** component provides the ability to specify its methods for working with report caching. For this purpose, a special **StiCacheHelper** class is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetReport** and **SaveReport**.

© 2003-2024 Stimulsoft

Default.aspx.cs

```
...
public partial class _Default : Page
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override object GetObject(string guid)
      {
         string path =
         Path.Combine(HttpContext.Current.Server.MapPath(string.Empty),
         "CacheFiles", guid);
         if (File.Exists(path))
         {
            byte[] cacheData = File.ReadAllBytes(path);
            return StiCacheHelper.GetObjectFromCacheData(cacheData);
         }
         return null;

         //return base.GetObject(guid);
      }

      public override void SaveObject(object obj, string guid)
      {
         byte[] cacheData = StiCacheHelper.GetCacheDataFromObject(obj);
         string path =
         Path.Combine(HttpContext.Current.Server.MapPath(string.Empty),
         "CacheFiles", guid);
         File.WriteAllBytes(path, cacheData);

         //base.SaveObject(obj, guid);
      }

      public override void RemoveReport(string guid)
      {
         string path =
         Path.Combine(HttpContext.Current.Server.MapPath(string.Empty),
         "CacheFiles", guid);
         if (File.Exists(path))
         {
            File.Delete(path);
         }
      }
   }

   static _Default()
   {
      StiWebDesigner.CacheHelper = new StiMyCacheHelper();
   }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as the value of the **StiWebDesigner.CacheHelper** static property in the ASPX

© 2003-2024 Stimulsoft

page constructor.

3.2.11 Designer Events

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component supports events that allow you to execute necessary operations before specific actions, such as creating and editing report templates, previewing, printing and exporting, interactivity, etc. Below is a sample for processing designer events.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   OnGetReport="StiWebDesigner1_GetReport"
   OnCreateReport="StiWebDesigner1_CreateReport"
   OnSaveReport="StiWebDesigner1_SaveReport">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void StiWebDesigner1_GetReport(object sender,
StiReportDataEventArgs e)
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));

   e.Report = report;
}

protected void StiWebDesigner1_CreateReport(object sender,
StiReportDataEventArgs e)
{
   DataSet data = new DataSet();
   data.ReadXmlSchema(Server.MapPath("Data/Demo.xsd"));
   data.ReadXml(Server.MapPath("Data/Demo.xml"));

   e.Report.RegData(data);
   e.Report.Dictionary.Synchronize();
}
```

© 2003-2024 Stimulsoft

```
protected void StiWebDesigner1_SaveReport(object sender,
StiReportDataEventArgs e)
{
   try
   {
      e.Report.Save(Server.MapPath("Reports/" + e.Report.ReportName +
      ".mrt"));
   }
   catch (Exception ex)
   {
      e.ErrorString = ex.Message;
   }
}
...
```

Events

Name	Description
OnGetReport	The event occurs when requesting a report for editing.
OnCreateReport	The event occurs when creating new reports from the designer menu.
OnOpenReport	The event occurs when you open a report from the designer menu. In the arguments of the event, the loaded report will be sent.
OnPreviewReport	The event occurs when going to the preview tab, and when interactive activities such as using report variables, dynamic collapsing, drill-down, and sorting a report when previewing it.
OnSaveReport	The event occurs when clicking the Save button on the panel or from the main menu of the designer.
OnSaveReportAs	The event occurs when clicking the Save As button from the main menu of the designer. If the event is not specified, the report will be saved to the local disk.

© 2003-2024 Stimulsoft

OnExportReport	The event occurs when exporting reports.
OnExportReportResponse	The event occurs when after exporting reports before saving the exported report file.
OnExit	The event occurs when clicking the Exit button in the main menu of the designer.

3.2.12 Timeout

When working with the **StiWebDesigner** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property
Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
  CacheTimeout="10">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
  StiWebDesigner1.CacheTimeout = 10;
}
...
```

Using cache will increase the speed of the report designer. See the chapter Caching for more information

© 2003-2024 Stimulsoft

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Default.aspx

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   RequestTimeout="10">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiWebDesigner1.RequestTimeout = 30;
}
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection and data sources in the report.

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   StiWebDesigner1.Report = report;
}
...
```

© 2003-2024 Stimulsoft

3.2.13 Add custom functions

Information

See on GitHub example of adding a custom function in the ASP.NET HTML5 Designer component.

You can add a custom function to the **Dictionary** in the report designer when integrating it into your application. After adding the custom function, you can use this in creating reports and dashboards. Below is the example of adding a function for calculating the sum total.

Default.aspx.cs

```
...
public static decimal MySum(object value)
{
   if (!ListExt.IsList(value))
   return Stimulsoft.Base.Helpers.StiValueHelper.TryToDecimal(value);

   return Stimulsoft.Data.Functions.Funcs.SkipNulls(ListExt.ToList(value))
   .TryCastToDecimal()
   .Sum();
}
...
static _Default()
{
   StiFunctions.AddFunction("MyCategory", "MySum",
   "description", typeof(_Default),
   typeof(decimal), "Calculates a sum of the specified set of values.",
   new[] { typeof(object) },
   new[] { "values" },
   new[] { "A set of values" }).UseFullPath = false;
}
...
```

3.2.14 Settings

The **HTML5 Designer** can be configured using the component properties that can be set on the ASPX page or using the C#/VB code. Below are examples of setting designer properties.

Default.aspx

© 2003-2024 Stimulsoft

```
...
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
   ShowClone="false"
   ShowCrossDataBand="false"
   ShowFileMenuOpen="false"
   PermissionBusinessObjects="None"
   PermissionDataConnections="View"
   DefaultUnit="Centimeters"
   Localization="Localization/de.xml"
   Theme="Office2022WhiteTeal">
</cc1:StiWebDesigner>
...
```

Default.aspx.cs

```
...
protected void Page_Load(object sender, EventArgs e)
{
   StiWebDesigner1.ShowClone = false;
   StiWebDesigner1.ShowCrossDataBand = false;
   StiWebDesigner1.ShowFileMenuOpen = false;
   StiWebDesigner1.PermissionBusinessObjects = StiDesignerPermissions.None;
   StiWebDesigner1.PermissionDataConnections = StiDesignerPermissions.View;
   StiWebDesigner1.DefaultUnit = StiReportUnitType.Centimeters;
   StiWebDesigner1.Localization = "Localization/de.xml";
   StiWebDesigner1.Theme = StiDesignerTheme.Office2022WhiteTeal;
   StiWebDesigner1.ReportDisplayMode = StiReportDisplayMode.Auto;
}
...
```

Basic Settings

Name	Description
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default width is 100%. By default, the component is expanded to the entire area of the browser window.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default height is 800

© 2003-2024 Stimulsoft

	pixels. By default, the component is expanded to the entire area of the browser window.

Server

Name	Description
RequestTimeout	Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled in **HTML5 Designer**; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as a repository, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of a lack of memory. The lower the priority is, the greater is the chance of removing information from

© 2003-2024 Stimulsoft

	memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or server session will be automatically re-saved after a certain period of time if the designer is idle (about every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the designer mode in which relative URLs are used for requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value that specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the designer. If **false**, only the necessary parameters are used to request the resources of the designer. This corresponds to the correct operation of the browser cache. By default, the property is set to **true**.
ShowServerErrorPage	Sets a value that enables or disables the display of the detailed server error in the preview. The default value is **true**.
UseCompression	Enables compression of designer requests in the GZip stream. This allows you to reduce the amount of Internet traffic but slows down the designer. By default, the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server-side. The following resources are supported: scripts, styles, and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the

© 2003-2024 Stimulsoft

	correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
Theme	Specifies the theme of the designer layout. The list of available themes can be found in the **StiDesignerTheme** enumeration. The default value is **Office2022WhiteBlue**.
CustomCss	Specifies the path to the CSS file of styles for the report designer. If this property is set, the standard styles of the selected theme will not be loaded. The default value is an empty value.
Localization	Specifies the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used, which is built into the viewer and does not require additional XML files.
LocalizationDirectory	Specifies the path to the directory with XML localization files. The localization files located in the specified folder will be loaded to the localization list in the designer panel.
DefaultUnit	Sets the units for the size of the report and all its components. By default, centimeters are used.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the **StiZoomMode** enumeration: › **PageWidth** – when the designer runs, the zoom, necessary to display the report by the page width, will be set; › **PageHeight** – when the designer runs, the zoom, required to display the page height of

© 2003-2024 Stimulsoft

	the report, will be set.
ShowAnimation	Enables animation for various elements of the designer interface. By default, the property is set to **true**.
ShowTooltips	Enables displaying tooltips for designer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enable displaying links to online documentation in tooltips for designer controls. By default, the property is set to **true**.
ShowDialogsHelp	Enables displaying links to online documentation on the titles of dialog forms of the designer. By default, the property is set to **true**.
InterfaceType	Sets the type of interface used for the designer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the interface type of the designer will be selected automatically depending on the device used (default value); › **Mouse** – forced use of the interface to control the designer with the mouse; › **Touch** – forced use of the Touch interface to control the designer via the touch screen (mobile devices), also in this mode, the interface elements are increased.
DatePickerFirstDayOfWeek	Sets the first day of the week for the select date item. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Monday** – the first day of the week is Monday (default value); › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.

© 2003-2024 Stimulsoft

ShowReportTree	Enables displaying the tree of report components. By default, the property is set to **true**.
ChartRenderType	Gets or sets the type of the chart in the preview. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages in the preview tab. Can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
ParametersPanelSortDataItems	Sets a value that indicates that variable items will be sorted. By default, the property is set to **true**.
CloseDesignerWithoutAsking	Sets a value that indicates that the designer will be closed without asking. By default, the

© 2003-2024 Stimulsoft

	property is set to **true**.
ShowSystemFonts	Sets visibility of the system fonts in the fonts list. By default, the property is set to **true**.
ShowNewPageButton	Sets visibility of the **New Page** button in the designer. By default, the property is set to **true**.
ShowNewDashboardButton	Sets visibility of the **New Dashboard** button in the designer. By default, the property is set to **true**.
WizardTypeRunningAfterLoad	Calls the Report wizard after starting the report designer. It may have one of the following **StiWizardType** enumeration values: › **None** - runs the report designer without running the report wizard; › **StandardReport** - runs the Standard wizard; › **MasterDetailReport** - runs the Master-Detail wizard; › **LabelReport** - runs the Label wizard; › **InvoicesReport** - runs the Invoice wizard; › **OrdersReport** - runs the Order wizard; › **QuotationReport** - runs the Quote wizard.

Behavior

Name	Description
ShowSaveDialog	Enables displaying the dialog to insert a report name when it is saved. The name of the report will be transferred in the parameters of the report designer. By default, the property is set to **true**.
UndoMaxLevel	Sets the maximum number to cancel actions with the report (the Undo/Redo function). A big value of this property will consume memory on the server side to store the undo parameters. The default value is **6**.

© 2003-2024 Stimulsoft

AllowChangeWindowTitle	Allows using the title of the browser window to display the file name of the edited report. By default, the property is set to **true**.
SaveReportMode	Sets the mode to save the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
SaveReportAsMode	Sets the mode for saving the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
CheckReportBeforePreview	Sets the value that allows running the report checker before preview.

FileMenu

Name	Description
Visible	Enables displaying the main menu of the report designer. By default, the property is set to **true**.
ShowNew	Enables showing the main menu item - **New**. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
ShowFileMenuNewReport	Sets a visibility of the new report button in the designer. By default, the property is set to **true**.
ShowFileMenuNewDashboard	Sets a visibility of the new dashboard button in the designer. By default, the property is set to **true**.
ShowOpen	Enables showing the main menu item - **Open**. By default, the property is set to **true**.
ShowSave	Enables showing the main menu item - **Save**. By default, the property is set to **true**.
ShowSaveAs	Enables showing the main menu item - **Save As**. By default, the property is set to **true**.
ShowClose	Enables showing the main menu item - **Close**. By default, the property is set to **true**.
ShowExit	Enables showing the main menu item - **Exit**. By default, the property is set to **true**.
ShowReportSetup	Enables showing the main menu item - **Report Setup**. By default, the property is set to **true**.
ShowOptions	Enables showing the main menu item - **Options**. By default, the property is set to **true**.
ShowInfo	Enables showing the main menu item - **Info**. By default, the property is set to **true**.
ShowAbout	Enables showing the main menu item - **About**. By default, the property is set to **true**.
ShowHelp	Enables showing the main menu item - **Help**. By default, the property is set to **true**.

Dictionary

Name	Description
Visible	Enables showing the data dictionary of the report. By default, the property is set to **true**.
UseAliases	Allows you to use aliases in the data dictionary.

© 2003-2024 Stimulsoft

	It has the three values of the **StiUseAliases** enumeration: › **Auto** - defines the mode of using aliases from a saved value in cookies (default value); › **True** - sets the mode of using aliases in the data dictionary; › **False** - disables the mode of using aliases in the data dictionary.
NewReportDictionary	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It has the three values of the **StiNewReportDictionary** enumeration: › **Auto** - defines the mode to create or join the data dictionary from a saved value in cookies (default value); › **DictionaryNew** - sets the mode to create a new data dictionary when creating a new report; › **DictionaryMerge** - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
ShowDictionaryContextMenuProperties	Sets a visibility of the **Properties** item in the dictionary context menu. By default, the property is set to **true**.
ShowDictionaryActions	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.
PermissionDataConnections	Sets the available actions to connect data to the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataSources	Sets available actions on report data sources. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataColumns	Sets the available actions on data columns in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.

© 2003-2024 Stimulsoft

PermissionBusinessObjects	Sets the available actions on the business objects in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataRelations	Sets the available actions to linking data in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionVariables	Sets available actions on report variables. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionResources	Sets the available actions for the resources in the Report Dictionary. Takes one or several values from the **StiDesignerPermissions** enumeration.
PermissionSqlParameters	Sets the available actions for the parameters of the SQL queries for the Report DataSources. Takes one or several values from **StiDesignerPermissions** enumeration.
DataTransformationsPermissions	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

The table below shows all available values for the **StiDesignerPermissions** enumeration, which can be set for the dictionary elements of the report.

Value	Description
None	Disables any action on the item of the data dictionary.
All	Allows any action on the item of the data dictionary.
Create	Allows creating a specific data dictionary item.
Delete	Allows deleting a specific data dictionary item.
Modify	Allows modifying a specific data dictionary item.
View	Allows viewing a specific data dictionary item.

© 2003-2024 Stimulsoft

ModifyView	Allows modifying and viewing a specific data dictionary item.

Toolbar

Name	Description
ShowToolbar	Enables displaying the toolbar of the designer. By default, the property is set to **true**.
ShowSetupToolboxButton	Enables displaying the button to call the dialog box with settings for the side toolbar. By default, the property is set to **true**.
ShowInsertButton	Enables displaying the **Insert** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowLayoutButton	Enables displaying the tab **Layout** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPageButton	Enables displaying the tab **Page** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPreviewButton	Enables displaying the tab **Preview** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the designer. By default, the property is set to **true**.
ShowAboutButton	Enables displaying the **About** on the toolbar of the designer. By default, the property is set to **false**.

PropertiesGrid

Name	Description

© 2003-2024 Stimulsoft

Visible	Enables displaying the property panel. By default, the property is set to **true**.
Width	Sets the width of the property panel. By default, the width is set to **370 px**.
LabelWidth	Specifies the width of the labels on the properties panel. By default, the width is set to **160 px**.
PropertiesGridPosition	Sets **Left** or **Right** position of the properties grid in the designer. It has the three values of the **StiPropertiesGridPosition** enumeration: › **Left**; › **Right**.
ShowPropertiesWhichUsedFromStyles	Sets a visibility of the properties which used from styles in the designer. By default, the property is set to **false**.

Components

Name	Description
ShowText	Enables displaying the **Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowTextInCells	Enables displaying the **Text in Cells** component in the insert menu for report components. By default, the property is set to **true**.
ShowRichText	Enables displaying the **Rich Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowImage	Enables displaying the **Image** component in the insert menu for report components. By default, the property is set to **true**.
ShowBarCode	Enables displaying the **Bar Code** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowShape	Enables displaying the **Shape** component in the insert menu for report components. By default, the property is set to **true**.
ShowHorizontalLinePrimitive	Enables displaying the **Horizontal Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowVerticalLinePrimitive	Enables displaying the **Vertical Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowRectanglePrimitive	Enables displaying the **Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowRoundedRectanglePrimitive	Enables displaying the **Rounded Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowPanel	Enables displaying the **Panel** component in the insert menu for report components. By default, the property is set to **true**.
ShowClone	Enables displaying the **Clone** component in the insert menu for report components. By default, the property is set to **true**.
ShowCheckBox	Enables displaying the **Check Box** component in the insert menu for report components. By default, the property is set to **true**.
ShowSubReport	Enables displaying the **Sub Report** component in the insert menu for report components. By default, the property is set to **true**.
ShowZipCode	Enables displaying the **Zip Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowTable	Enables displaying the **Table** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowCrossTab	Enables displaying the **Cross-Tab** component in the insert menu for report components. By default, the property is set to **true**.
ShowChart	Enables displaying the **Chart** component in the insert menu for report components. It affects on all chart types. By default, the property is set to **true**.
ShowMap	Enables displaying the **Map** component in the insert menu for report components. By default, the property is set to **false**.
ShowGauge	Enables displaying the **Gauge** component in the insert menu for report components. By default, the property is set to **false**.
ShowSparkline	Enables displaying the **Sparkline** component in the insert menu for report components. By default, the property is set to **true**.
ShowMathFormula	Enables displaying the **Math Formula** component in the insert menu for report components. By default, the property is set to **true**.
ShowElectronicSignature	Enables displaying the **Electronic Signature** component in the insert menu for report components. By default, the property is set to **true**.
ShowPdfDigitalSignature	Enables displaying the **PDF Digital Signature** component in the insert menu for report components. By default, the property is set to **true**.

Bands

Name	Description
ShowReportTitleBands	Enables displaying the **Report Title** item in the **Bands** menu of the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowReportSummaryBand	Enables displaying the **Report Summary** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageHeaderBand	Enables displaying the **Page Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageFooterBand	Enables displaying the **Page Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupHeaderBand	Enables displaying the **Group Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupFooterBand	Enables displaying the **Group Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHeaderBand	Enables displaying the **Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowFooterBand	Enables displaying the **Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnHeaderBand	Enables displaying the **Column Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnFooterBand	Enables displaying the **Column Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowDataBand	Enables displaying the **Data** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHierarchicalBand	Enables displaying the **Hierarchical** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowChildBand	Enables displaying the **Child** item in the **Bands** menu of the designer. By default, the property is

© 2003-2024 Stimulsoft

	set to **true**.
ShowEmptyBand	Enables displaying the **Empty** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowOverlayBand	Enables displaying the **Overlay** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowTableOfContents	Enables displaying the **Table of Contents** item in the **Bands** menu of the designer. By default, the property is set to **true**.

DashboardElements

Name	Description
ShowTableElement	Enables displaying the **Table** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowCardsElement	Enables displaying the **Cards** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowChartElement	Enables displaying the **Chart** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowGaugeElement	Enables displaying the **Gauge** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPivotTableElement	Enables displaying the **Pivot** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowIndicatorElement	Enables displaying the **Indicator** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowProgressElement	Enables displaying the **Progress** element in the Dashboard Elements menu of the designer. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
ShowRegionMapElement	Enables displaying the **Region Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowOnlineMapElement	Enables displaying the **Online Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowImageElement	Enables displaying the **Image** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTextElement	Enables displaying the **Text** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPanelElement	Enables displaying the **Panel** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowShapeElement	Enables displaying the **Shape** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowButtonElement	Enables displaying the **Button** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowListBoxElement	Enables displaying the **List Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowComboBoxElement	Enables displaying the **Combo Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewElement	Enables displaying the **Tree View** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewBoxElement	Enables displaying the **Tree View Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowDatePickerElement	Enables displaying the **Date Picker** element in

© 2003-2024 Stimulsoft

	the Dashboard Elements menu of the designer. By default, the property is set to **true**.

CrossBands

Name	Description
ShowCrossGroupHeaderBand	Enables displaying the **Cross Group Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossGroupFooterBand	Enables displaying the **Cross Group Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossHeaderBand	Enables displaying the **Cross Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossFooterBand	Enables displaying the **Cross Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossDataBand	Enables displaying the **Cross Data** section in the insert menu for report components. By default, the property is set to **true**.

Dashboards

Name	Description
ShowNewDashboardButton	Sets a visibility of the **New Dashboard** button in the designer. By default, the property is set to **true**.

Pages

© 2003-2024 Stimulsoft

Name	Description
ShowNewPageButton	Sets a visibility of the **New Page** button in the designer. By default, the property is set to **true**.

When designing a report or dashboard in the report designer, you can also define **ExportOptions**, **EmailOptions**, and **PreviewToolbarOptions** on the **Preview** tab. These options are similar to the report viewer options.

4 Reports and Dashboards for ASP.NET MVC

ASP.NET MVC is a technology for creating web applications and web services using the **Model-View-Controller** template. Stimulsoft provides tools for creating and displaying reports and dashboards on any device using this technology.

Tools for creating and editing reports:	Tools for viewing and converting reports:	Tools for creating and editing dashboards:	Tools for viewing and converting dashboards:
HTML5 designer	HTML5 viewer	HTML5 designer	HTML5 viewer

4.1 HTML5 Viewer

YouTube

Watch videos for the ASP.NET MVC HTML5 Viewer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

Samples

See on GitHub examples for working with the ASP.NET MVC HTML5 Viewer

© 2003-2024 Stimulsoft

component. All samples are separate projects grouped into one solution for Visual Studio.

The **HTML5 Viewer** (**StiMvcViewer**) component is designed for viewing reports in the web browser. You do not need to install the .NET Framework, ActiveX components, or any special plug-ins on the client-side. All you need is any modern Web browser.

With the help of the **HTML5 Viewer**, you can view, print, and export reports on any computer with any operating system installed. Since the viewer only uses HTML and JavaScript technologies, it can be run on devices with no Flash or Silverlight support - tablets, smartphones. Also, Viewer supports the following interfaces: Mobile Interface and Touch Interface. These interfaces will be switched on automatically when the mobile devices are used or for the touchscreen displays.

The **HTML5 Viewer** component uses the AJAX technology to perform all actions (uploading a report, paging, scaling, interactivity in reports, etc.), allowing you to get rid of reloading the entire page and save Web traffic, and speed up work.

The **HTML5 Viewer** supports many themes, animated interface, bookmarks, interactive reports, editing of report elements on the page, full-screen mode, search panel, and other necessary features for viewing reports.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Viewer** in a Web project, you should install the NuGet package of Stimulsoft.Reports.Web:

- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Reports.Web in the search bar on the Browse tab;
- Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

© 2003-2024 Stimulsoft

If this is not possible, you should add the following assemblies to the project:
- Stimulsoft.Base.dll
- Stimulsoft.Report.dll
- Stimulsoft.Report.Check.dll
- Stimulsoft.Report.Helper.dll
- Stimulsoft.Report.Mvc.dll
- Stimulsoft.Report.Web.dll
- Stimulsoft.Report.WebDesign.dll

To add the ability to view and export dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web (this package is associated with the package Stimulsoft.Reports.Web. If it is missed, it will be installed automatically):
- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Dashboards.Web in the search bar on the Browse tab;
- Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

If for any reason, this is not possible, you should additionally add the following assemblies to the project:
- Stimulsoft.Dashboard.dll
- Stimulsoft.Dashboard.Drawing.dll
- Stimulsoft.Dashboard.Export.dll

- How this works?
- Activation
- Showing Reports and Dashboards
- Connecting data
- Localization
- Printing Reports
- Exporting Reports and Dashboards
- Viewing Modes
- Interactive Reports
- Timeout
- Editing Rendered Reports
- Sending Reports by Email
- Calling Designer from Viewer
- Caching
- Using Themes
- Basic Features

© 2003-2024 Stimulsoft

- Work with Parameters
- Work with Bookmarks
- Viewer Settings
- Additional Methods
- Export and Printing from Code

4.1.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the viewer, you need to place the **StiMvcViewer** component on the page, set the necessary settings, and set the necessary actions in the view controller. When the report viewer runs, the following actions occur:

› The .NET component generates HTML and JavaScript code that is necessary for displaying and running the viewer;

› When the component is output, the JavaScript method is launched. It requests the first page of the report on the server-side or the entire report (depending on the selected mode) and the required report parameters;

› Each action in the viewer (for example, paging, printing or exporting a report, etc.) calls a certain action on the server-side, in which you can perform the necessary manipulations with the report;

› To speed up the work, the viewer saves the report in cache or server session, which makes it impossible to re-build the report.

4.1.2 Activation

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiMvcViewer** component.

HomeController.cs

```
...
public class HomeController : Controller
{
   static HomeController()
```

© 2003-2024 Stimulsoft

```
    {
        //Activation with using license code
        Stimulsoft.Base.StiLicense.Key = "Your activation code...";

        //Activation with using license file
        var path = System.Web.HttpContext.Current.Server.MapPath("~/Content/
        license.key");
        Stimulsoft.Base.StiLicense.LoadFromFile(path);
    }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

4.1.3 Showing Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

To show the report, you need to add the **StiMvcViewer** component to the page, set it to the minimum required properties, and specify the necessary actions in the view controller.

© 2003-2024 Stimulsoft

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         GetReport = "GetReport",
         ViewerEvent = "ViewerEvent"
      }
})
...
```

HomeController.cs

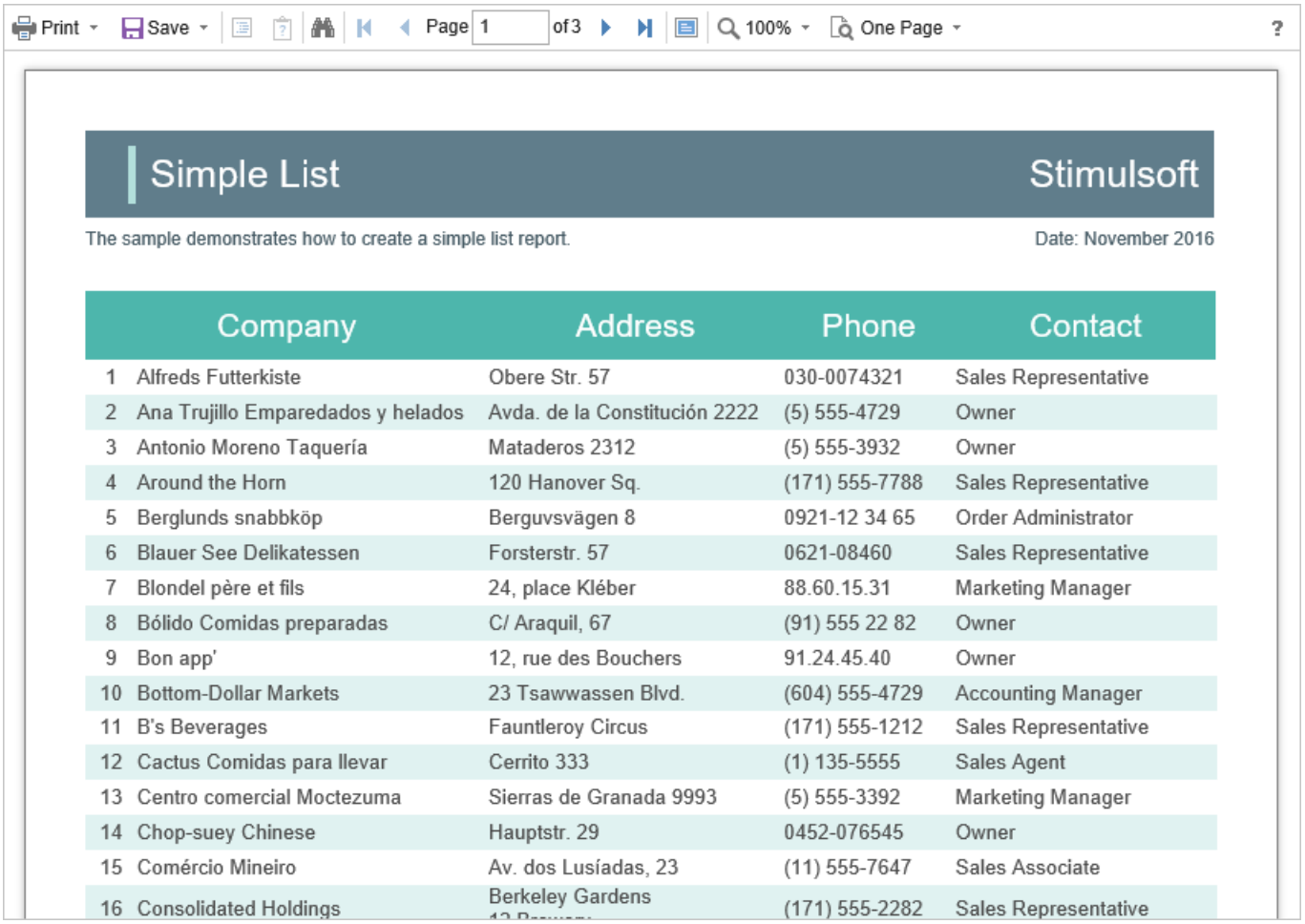
```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Content/SimpleList.mrt"));
   //report.Load(Server.MapPath("~/Content/Dashboard.mrt"));

   return StiMvcViewer.GetReportResult(report);
}

public ActionResult ViewerEvent()
{
   return StiMvcViewer.ViewerEventResult();
}
...
```

In the above example, the processing of two actions of the viewer is added. The **GetReport** action returns the report prepared for preview. The **ViewerEvent** action handles the viewer events.

Information

The **ViewerEvent** action is mandatory. Without it, the correct operation of the viewer is not possible.

© 2003-2024 Stimulsoft



	Company	Address	Phone	Contact
1	Alfreds Futterkiste	Obere Str. 57	030-0074321	Sales Representative
2	Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222	(5) 555-4729	Owner
3	Antonio Moreno Taquería	Mataderos 2312	(5) 555-3932	Owner
4	Around the Horn	120 Hanover Sq.	(171) 555-7788	Sales Representative
5	Berglunds snabbköp	Berguvsvägen 8	0921-12 34 65	Order Administrator
6	Blauer See Delikatessen	Forsterstr. 57	0621-08460	Sales Representative
7	Blondel père et fils	24, place Kléber	88.60.15.31	Marketing Manager
8	Bólido Comidas preparadas	C/ Araquil, 67	(91) 555 22 82	Owner
9	Bon app'	12, rue des Bouchers	91.24.45.40	Owner
10	Bottom-Dollar Markets	23 Tsawwassen Blvd.	(604) 555-4729	Accounting Manager
11	B's Beverages	Fauntleroy Circus	(171) 555-1212	Sales Representative
12	Cactus Comidas para llevar	Cerrito 333	(1) 135-5555	Sales Agent
13	Centro comercial Moctezuma	Sierras de Granada 9993	(5) 555-3392	Marketing Manager
14	Chop-suey Chinese	Hauptstr. 29	0452-076545	Owner
15	Comércio Mineiro	Av. dos Lusíadas, 23	(11) 555-7647	Sales Associate
16	Consolidated Holdings	Berkeley Gardens	(171) 555-2282	Sales Representative

If the report is not rendered before showing, the **HTML5 Viewer** component will automatically render it. Thus, you can use report templates, rendered reports, and reports in the form of classes to show the report.

HomeController.cs

```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   report.LoadDocument(Server.MapPath("~/Content/SimpleList.mdc"));

   return StiMvcViewer.GetReportResult(report);
}
...
```

HomeController.cs

```
...
```

© 2003-2024 Stimulsoft

```
public ActionResult GetReport()
{
   StiReport report = new StiReportCompiledClass();

   return StiMvcViewer.GetReportResult(report);
}
...
```

Since the dashboard is not a static document and requires data to work, the format of the rendered MDC document is not available for it. Instead, it is possible to use a snapshot of the report in the MRT format, which contains all the data necessary for the dashboard to work and can be correctly displayed in the viewer.

Default.aspx.cs

```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Reports/Snapshot.mrt"));

   return StiMvcViewer.GetReportResult(report);
}
...
```

Loading custom fonts

You can load custom fonts using the **StiFontCollection** class, having specified the file that contains a font. To do this, you should call the static method in the controller constructor to load a font.

ViewerController.cs

```
...
public class ViewerController : Controller
{
   static ViewerController()
   {
      Stimulsoft.Base.StiFontCollection.AddFontFile(Server.MapPath("~/
      fonts/my-font/font-name.ttf"));
   }
}
...
```

4.1.4 Connecting Data

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Data to a report can be connected in various ways. The easiest way is to store connection settings in the report template. You can also connect the data from the code. You can do this when the report is loaded in the **GetReport** action.

HomeController.cs

```
...
public ActionResult GetReport()
{
   DataSet ds = new DataSet();
   ds.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Content/TwoSimpleLists.mrt"));
   report.Dictionary.Databases.Clear();
   report.RegData("Demo", ds);

   return StiMvcViewer.GetReportResult(report);
}
...
```

Data for the report can be connected not only when the report is loaded. For example, you can connect new data at the moment of interactive actions in the viewer (applying report parameters, sorting, drill-down, collapsing). To do this, you should set the **Interaction** action for the **HTML5 Viewer** component and, in the action handler, connect the data for the current report. In the same way, you can connect data in other actions of the viewer.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         GetReport = "GetReport",
         ViewerEvent = "ViewerEvent",
         Interaction = "ViewerInteraction"
      }
```

© 2003-2024 Stimulsoft

```
})
...
```

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   DataSet data = new DataSet();
   data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));

   StiReport report = StiMvcViewer.GetReportObject();
   report.RegData("Demo", data);

   return StiMvcViewer.InteractionResult(report);
}
...
```

If you want to connect new data only for a certain interactive action of the viewer, for example, only when you apply report parameters you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side on any request and contain all necessary information and states of the client part of the viewer. To determine the type of action of the viewer, it is enough to check the **Action** property of the viewer parameters.

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   StiRequestParams requestParams = StiMvcViewer.GetRequestParams();
   if (requestParams.Action == StiAction.Variables)
   {
      DataSet data = new DataSet();
      data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));

      StiReport report = StiMvcViewer.GetReportObject();
      report.RegData("Demo", data);

      return StiMvcViewer.InteractionResult(report);
   }

   return StiMvcViewer.InteractionResult();
}
...
```

SQL data sources

© 2003-2024 Stimulsoft

The connection parameters to the SQL data source and any other ones can be stored in the report template. Suppose you want to set the connection parameters from the code before rendering the report (for example, for security reasons or depending on the authorized user). In that case, you can use the example below.

HomeController.cs

```
...
public ActionResult GetReport()
{
   OracleConnection connection = new OracleConnection("Data
   Source=Oracle8i;Integrated Security=yes");
   connection.Open();
   OracleDataAdapter adapter = new OracleDataAdapter();
   adapter.SelectCommand = new OracleCommand("SELECT * FROM Products",
   connection);

   DataSet dataSet = new DataSet("productsDataSet");
   adapter.Fill(dataSet, "Products");

   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Content/SqlSampleReport.mrt"));
   report.RegData("Products", dataSet);

   return StiMvcViewer.GetReportResult(report);
}
...
```

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection and data sources in the report.

HomeController.cs

```
...
public ActionResult GetReport()
{

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlDatabase)
   report.Dictionary.Databases["Connection"]).ConnectionString = @"Data
   Source=server;Integrated Security=True;Initial Catalog=DataBase";
   ((StiSqlSource)
```

© 2003-2024 Stimulsoft

```
    report.Dictionary.DataSources["DataSourceName"]).SqlCommand = "select *
    from Table where Column = 100";
    ((StiSqlSource)
    report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

    return StiMvcViewer.GetReportResult(report);
}
...
```

Information

For SQL data sources of other types, the connection is created similarly, and an adapter corresponding to the data source type is connected. For example, for the MS SQL data source, you should connect SqlDataAdapter. For OLE DB - OleDbDataAdapter is required. Also, you should specify a connection string that matches the connection type.

You can also use data for designing reports and dashboards obtained from OData storage. In this case, you can do the authorization using a username, user password, or token. Authorization parameters are specified in the connection string to the OData storage using the ";" separator.

HomeController.cs

```
...
public ActionResult GetReport()
{

    var report = new StiReport();

    //Authorization using a user account
    var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
    services.odata.org/V4/Northwind/
    Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Password;C
    lient_Id=Your Client ID", false, null);

    //Authorization using a user token
    var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
    services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your token",
    false, null);

    report.Dictionary.Databases.Add(oDataDatabase);
    oDataDatabase.Synchronize(report);

    //Query with data filter
    ((StiSqlSource)report.Dictionary.DataSources["Products"]).SqlCommand =
```

© 2003-2024 Stimulsoft

```
    "Products?$filter=ProductID eq 2";

    return StiMvcViewer.GetReportResult(report);
}
...
```

The table below shows the connection string templates for different types of data sources.

Data Source	Connection String Template
MS SQL	Integrated Security=False; Data Source=myServerAddress;Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
MySQL	Server=myServerAddress; Database=myDataBase;UserId=myUsername; Pwd=myPassword;
ODBC	Driver={SQL Server}; Server=myServerAddress;Database=myDataBase ; Uid=myUsername; Pwd=myPassword;
OLE DB	Provider=SQLOLEDB.1; Integrated Security=SSPI;Persist Security Info=False; Initial Catalog=myDataBase;Data Source=myServerAddress
Oracle	Data Source=TORCL;User Id=myUsername;Password=myPassword;
MS Access	Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Password=pass;Data Source=C:\ \myAccessFile.accdb;
PostgreSQL	Server=myServerAddress; Port=5432; Database=myDataBase;User Id=myUsername; Password=myPassword;
Firebird	User=SYSDBA; Password=masterkey; Database=SampleDatabase.fdb;DataSource=my ServerAddress; Port=3050; Dialect=3; Charset=NONE;Role=; Connection lifetime=15;

© 2003-2024 Stimulsoft

	Pooling=true; MinPoolSize=0;MaxPoolSize=50; Packet Size=8192; ServerType=0;
SQL CE	Data Source=c:\MyData.sdf; Persist Security Info=False;
SQLite	Data Source=c:\mydb.db; Version=3;
DB2	Server=myAddress:myPortNumber;Database=myDataBase;UID=myUsername;PWD=myPassword; Max Pool Size=100;Min Pool Size=10;
Infomix	Database=myDataBase;Host=192.168.10.10;Server=db_engine_tcp;Service=1492;Protocol=onsoctcp;UID=myUsername;Password=myPassword;
Sybase	Data Source=myASEserver;Port=5000;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
Teradata	Data Source=myServerAddress;User ID=myUsername;Password=myPassword;
VistaDB	Data Source=D:\folder \myVistaDatabaseFile.vdb4;Open Mode=ExclusiveReadWrite;
Universal(dotConnect)	Provider=Oracle;direct=true;data source=192.168.0.1;port=1521;sid=sid;user=user; password=pass
MongoDB	mongodb://<user>:<password>@localhost/test
OData	http://services.odata.org/v3/odata/OData.svc/
Other...	The table shows the most commonly used templates for the connection string. You can view various connection string options at the special website.

Data from XML, JSON, Excel files

Connecting to XML and JSON data sources can be stored in the report template. If you want to specify data files from the code, you can use the example below.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public ActionResult GetReport()
{
   DataSet data = new DataSet();
   data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Content/SimpleList.mrt"));
   report.RegData(data);

   return StiMvcViewer.GetReportResult(report);
}
...
```

HomeController.cs

```
...
public ActionResult GetReport()
{
   DataSet data
   = StiJsonToDataSetConverterV2.GetDataSetFromFile(Server.MapPath("~/
   Content/Data/Demo.json"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Content/SimpleList.mrt"));
   report.RegData(data);

   return StiMvcViewer.GetReportResult(report);
}
...
```

Information

The viewer has the possibility of obtaining data from an Excel file. To do this, you can use the following method.

```
DataSet dataSet = StiExcelConnector.Get().GetDataSet(new StiExcelOptions(ar
```

4.1.5 Localization

The **HTML5 Viewer** component supports the complete localization of its interface. To localize the report viewer interface, use the special **Localization** property. The value of this property should specify the path to the localization XML file (relative or absolute).

Index.cshtml

© 2003-2024 Stimulsoft

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Localization = "~/Content/Localization/en.xml"
})
...
```

When you load the report viewer, the localization file will be loaded automatically.

4.1.6 Using Themes

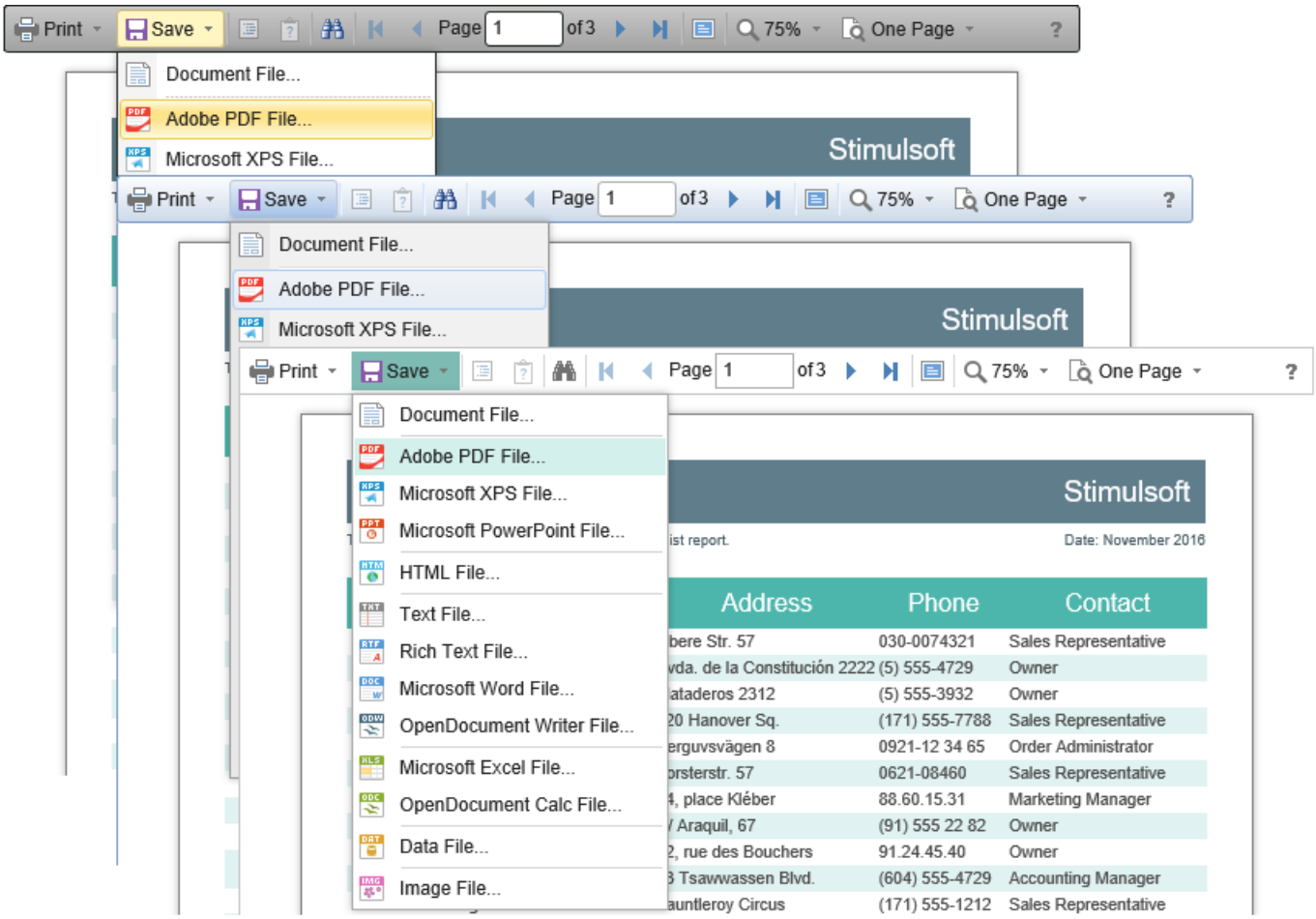
The **HTML5 Viewer** component can change the appearance of visual controls. To change the theme, use the **Theme** property, which can take one of the values of the **StiViewerTheme** enumeration.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Theme = StiTheme.Office2022WhiteTeal
})
...
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.

© 2003-2024 Stimulsoft



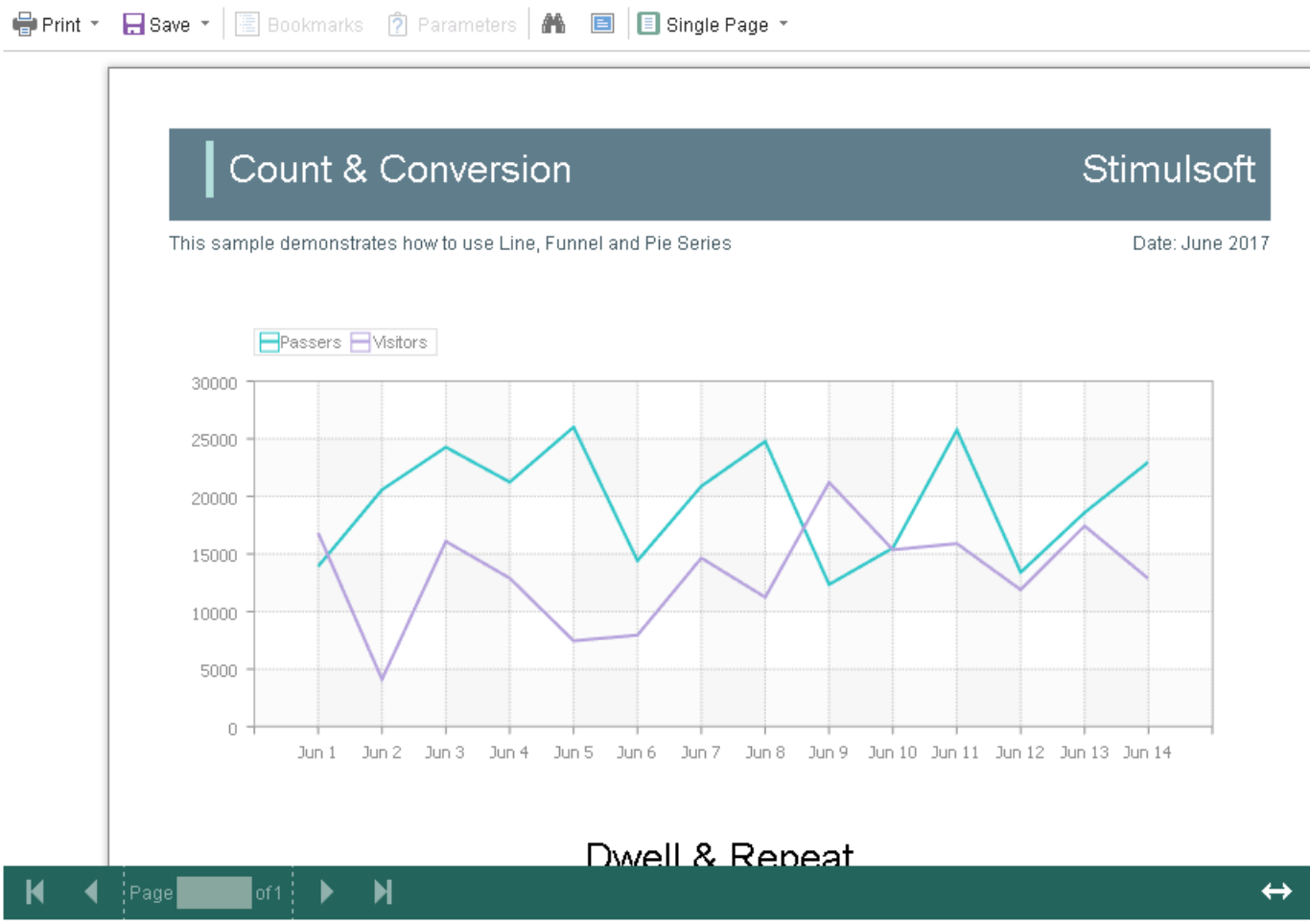
By default, the viewer has only the top toolbar on which all the report controls are located. If necessary, the toolbar can be split into top and bottom parts. The top panel will contain the menu for printing and exporting the report and the buttons for working with parameters and bookmarks. The bottom panel will contain controls to switch between the report pages and setting the zoom of pages. To enable this mode, enable the **ToolbarDisplayMode** property. It has values **Simple** and **Separated**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Appearance =
      {
         ScrollbarsMode = true
      },
      Toolbar =
      {
```

© 2003-2024 Stimulsoft

```
        DisplayMode = StiToolbarDisplayMode.Separated
    }
})
...
```



In addition, it is possible to set the appearance parameters for the main elements of the viewer. For example, you can change the font and color of the control panel inscriptions of the viewer, set the background of the viewer, set the color of page borders, etc. Below is a list of available properties that change the appearance of the viewer and their default values.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
```

© 2003-2024 Stimulsoft

```
    Appearance =
    {
      BackgroundColor = Color.White,
      PageBorderColor = Color.Blue,
      ShowPageShadow = true
    },
    Toolbar =
    {
      BackgroundColor = Color.White,
      BorderColor = Color.Gray,
      FontColor = Color.Black,
      FontFamily = "Arial"
    }
})
...
```

4.1.7 Basic Features

The main features of the viewer include the following operations:

- Displaying the report.
- Switching between the report pages.
- Changing the scale.
- Displaying the preview mode.

All specified operations are performed in the AJAX mode without restarting the browser page. For the correct work of these operations, you should define a special ViewerEvent action.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
  new StiMvcViewerOptions() {
    Actions =
    {
      ViewerEvent = "ViewerEvent"
    }
})
...
```

HomeController.cs

```
...
public ActionResult ViewerEvent()
{
  // Some code before viewer event
  // ...

  return StiMvcViewer.ViewerEventResult();
}
...
```

© 2003-2024 Stimulsoft

Information

This action is mandatory. Without it, the correct operation of the viewer is not possible.

The **ViewerEvent** action returns a prepared HTML page of the report (or set of pages), built taking into account the current state of the viewer. If necessary, you can change the parameters of the current report in the specified action and update the report data in case of interactive actions of the viewer.

4.1.8 Printing Reports

Information

Please note that the print option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides several options for printing a report. Each has its advantages and disadvantages.

Print to PDF
Printing will be done by exporting the report to PDF. The advantages are greater accuracy of positioning and printing of the report elements compared to other printing options. Among the drawbacks is the mandatory presence of a plug-in installed in a web browser for viewing PDF files (modern browsers have embedded PDF viewer and printer).

Print with Preview
The report will be printed in a separate pop-up browser window in HTML. The report can be previewed and then sent to the printer or copied to another location as text or HTML code. Advantages - cross-browser compatibility when printing, no need to install special plug-ins. The disadvantage is the relatively low accuracy of the position of the report elements due to the peculiarities of the implementation of HTML formatting.

© 2003-2024 Stimulsoft

Print without Preview

The report will be printed directly to the printer without preview. After selecting this menu item, the system print dialog is displayed. Since printing in this mode is carried out in the HTML format, the print quality is similar to printing a report with a preview.

Information

When printing to the **HTML format**, you should check the compliance of report page settings and printer parameters (paper size, orientation, margins, indents) and check your browser print settings, such as margins, headers, footers watermarks printing, color printing.

The print function does not require additional settings for the viewer. Before printing a report, you can define a special **PrintReport** action if you need to perform any actions.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         PrintReport = "PrintReport"
      }
})
...
```

HomeController.cs

```
...
public ActionResult PrintReport()
{
   // Some code before print
   // ...

   return StiMvcViewer.PrintReportResult();
}
...
```

Print setup

© 2003-2024 Stimulsoft

If you choose to print a report in the viewer panel, a menu with printing options is displayed. The **HTML5 Viewer** component is able to force the required printing mode. To do this, set the **PrintDestination** property to one of the following values of the **StiPrintDestination** enumeration.

- **Default** – the menu will be displayed (the default property value);
- **Pdf** – print to the PDF format;
- **Direct** – printing to the HTML format directly to the printer, the system print dialog will be displayed;
- **WithPreview** – print to the HTML format with preview in a pop-up window.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Toolbar =
      {
         PrintDestination = StiPrintDestination.Default
      }
})
...
```

The **HTML5 Viewer** component can completely disable report printing. To do this, set the value of the **ShowPrintButton** property to **false**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Toolbar =
      {
         ShowPrintButton = false
      }
})
...
```

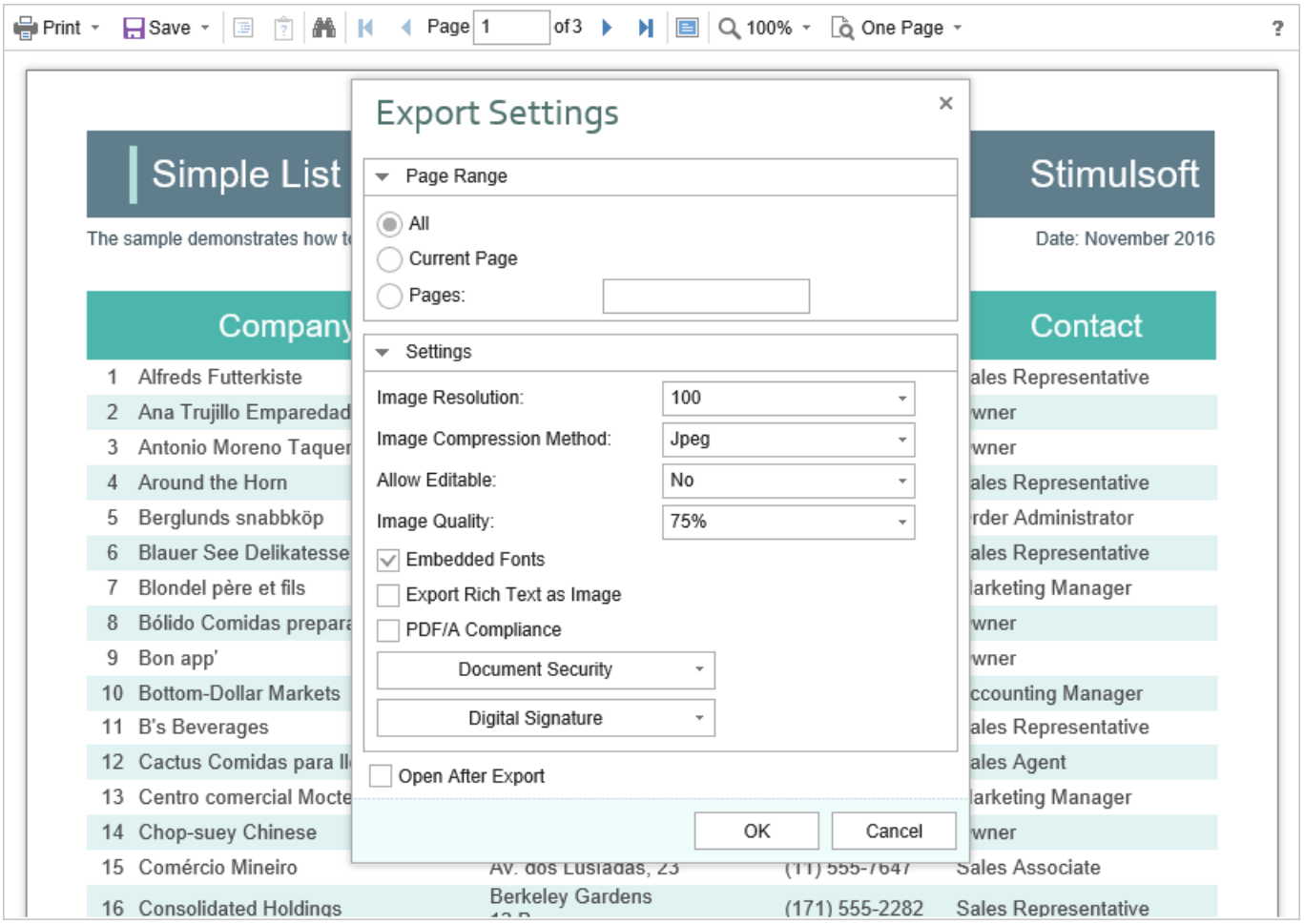
4.1.9 Exporting Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

The **HTML5 Viewer** component allows you to export the displayed report to three dozen various formats, such as **PDF**, **HTML**, **Word**, **Excel**, **XPS**, **RTF**, **images**, **text**, and others. You may export the dashboard to PDF, Excel, image files.



The export function does not require additional settings for the viewer. If you need to perform any actions before exporting the report, you can define a special **ExportReport** action.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
```

© 2003-2024 Stimulsoft

```
    Actions =
    {
      ExportReport = "ExportReport"
    }
})
...
```

HomeController.cs

```
...
public ActionResult ExportReport()
{
  // Some code before export
  // ...

  return StiMvcViewer.ExportReportResult();
}
...
```

Export settings

Each report export format of the **HTML5 Viewer** component has a lot of settings, and each setting has its default values. Sometimes you need to set other default values. For this purpose, a special **DefaultSettings** property of the viewer is used. It is a container for all the default export settings.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
  new StiMvcViewerOptions() {
    Exports =
    {
      DefaultSettings =
      {
        ExportToPdf =
        {
          ImageQuality = 0.75f,
          ImageFormat = Stimulsoft.Report.Export.StiImageFormat.Color
        },
        ExportToHtml =
        {
          ExportMode = Stimulsoft.Report.Export.StiHtmlExportMode.Div,
          UseEmbeddedImages = true
        }
      }
    }
})
...
```

© 2003-2024 Stimulsoft

If it is required, you can completely hide export dialogs. Exporting will always be done with default settings. For this, it is enough to set the value of the **ShowExportDialog** property to **false**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Exports =
      {
         ShowExportDialog = false
      }
})
...
```

The **HTML5 Viewer** component contains 30+ export formats, and sometimes you need to disable unwanted formats. This allows you to simplify UI and the use of the viewer. To disable unused export formats, it is enough to set the values for the corresponding properties of the viewer listed in the list below to **false**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Exports =
      {
         ShowExportToDocument = true,
         ShowExportToPdf = true,
         ShowExportToXps = true,
         ShowExportToPowerPoint = true,
         ShowExportToHtml = true,
         ShowExportToHtml5 = true,
         ShowExportToMht = true,
         ShowExportToText = true,
         ShowExportToRtf = true,
         ShowExportToWord2007 = true,
         ShowExportToOpenDocumentWriter = true,
         ShowExportToExcel = true,
         ShowExportToExcelXml = true,
         ShowExportToExcel2007 = true,
         ShowExportToOpenDocumentCalc = true,
         ShowExportToCsv = true,
         ShowExportToDbf = true,
         ShowExportToXml = true,
         ShowExportToDif = true,
         ShowExportToSylk = true,
         ShowExportToImageBmp = true,
```

© 2003-2024 Stimulsoft

```
        ShowExportToImageGif = true,
        ShowExportToImageJpeg = true,
        ShowExportToImagePcx = true,
        ShowExportToImagePng = true,
        ShowExportToImageTiff = true,
        ShowExportToImageMetafile = true,
        ShowExportToImageSvg = true,
        ShowExportToImageSvgz = true
      }
})
...
```

The **HTML5 Viewer** component can completely disable the export menu. To do this, set the value of the **ShowSaveButton** property to **false**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Toolbar =
      {
         ShowSaveButton = false
      }
})
...
```

4.1.10 Viewing Modes

The **HTML5 Viewer** component has two modes for displaying reports - with and without scrollbars. By default, the view mode without scrollbars is set. To enable the scrollbar view mode, set the value of the **ScrollbarsMode** property to **true**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Appearance =
      {
         ScrollbarsMode = true
      }
})
...
```

In the first mode (without scrollbars), the viewer displays a page or report as a whole, automatically stretching the preview space. If the width and height are specified, the viewer will truncate the page that is out of bounds. In the second mode, unlike the

© 2003-2024 Stimulsoft

first one, when the page is out of bounds of the viewer's size, no truncation will be performed. Scrollbars will appear, using which you can view the entire page or report.

Information

In the report mode with scrollbars, you should set the height of the viewer. Otherwise, the default height will be set to **650 pixels**.

The **HTML5 Viewer** component provides the full-screen report and dashboard mode. By default, the standard view mode is enabled, the viewer has the specified dimensions in the settings. To enable the full-screen mode, set the **FullScreenMode** property to **true**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Appearance =
      {
         FullScreenMode = true
      }
})
...
```

Also, to enable or disable the full-screen mode, you can use the corresponding button on the control panel of the viewer.

The **HTML5 Viewer** component has three modes to display reports - page-by-page, entire report, and tabular display of report pages. To control the modes, the **ViewMode** property is used. It can have one of the specified values - **SinglePage**, **Continuous**, **MultiplePages**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Toolbar =
      {
```

© 2003-2024 Stimulsoft

```
        ViewMode = StiWebViewMode.OnePage
      }
})
...
```

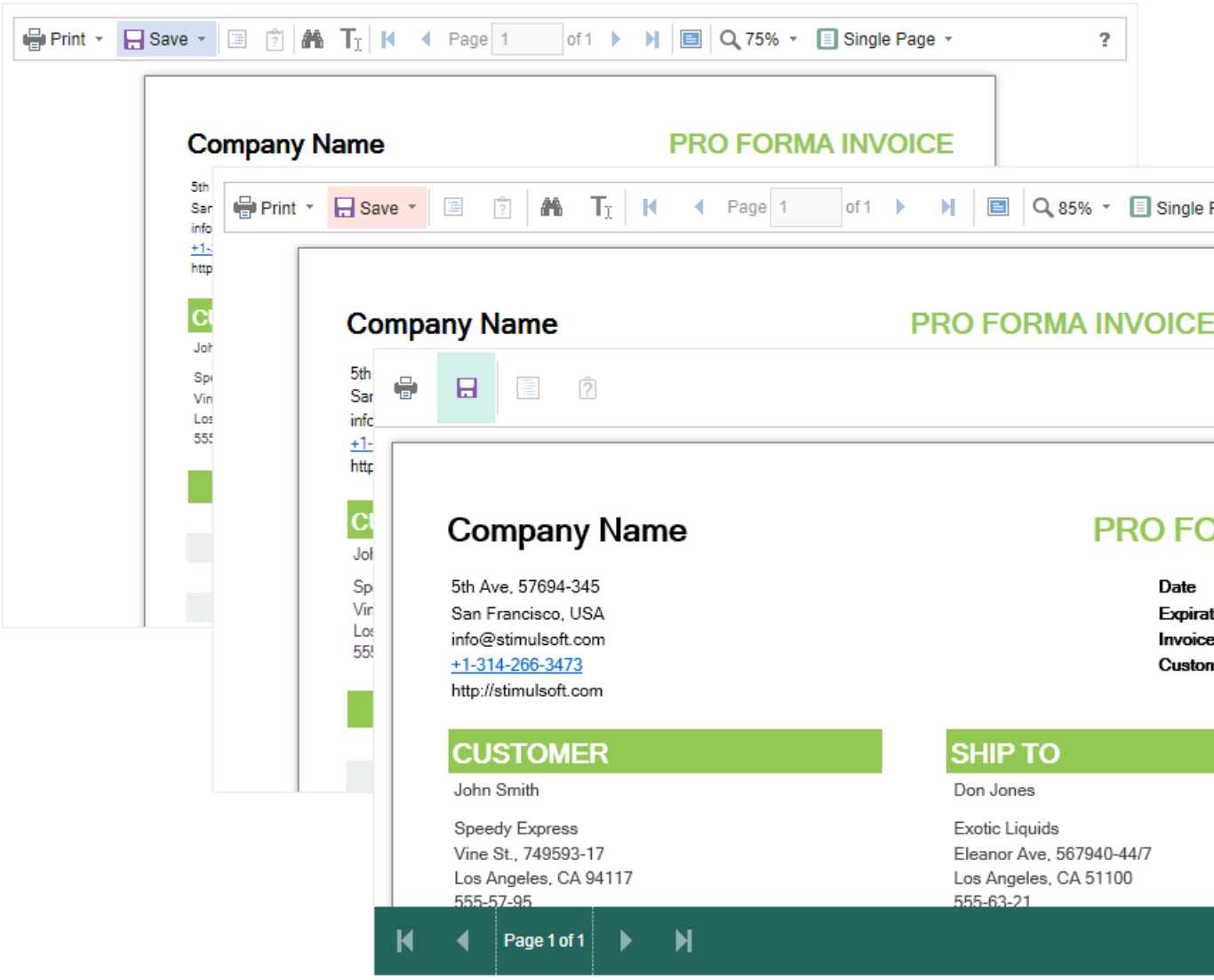
HTML5 Viewer component supports interaction on a regular PC display and works with a touchscreen of screens and mobile devices. **InterfaceType** property allows controlling the interface modes. The property can have one of the following values:

- **Auto** – the viewer's interface is determined automatically depending on the device the report is displayed on. This is the default value.
- **Mouse** – the standard interface with a mouse control will be used for all the screen types.
- **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability.
- **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with smartphones.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Appearance =
      {
         InterfaceType = StiInterfaceType.Auto
      }
})
...
```

© 2003-2024 Stimulsoft



4.1.11 Work with Parameters

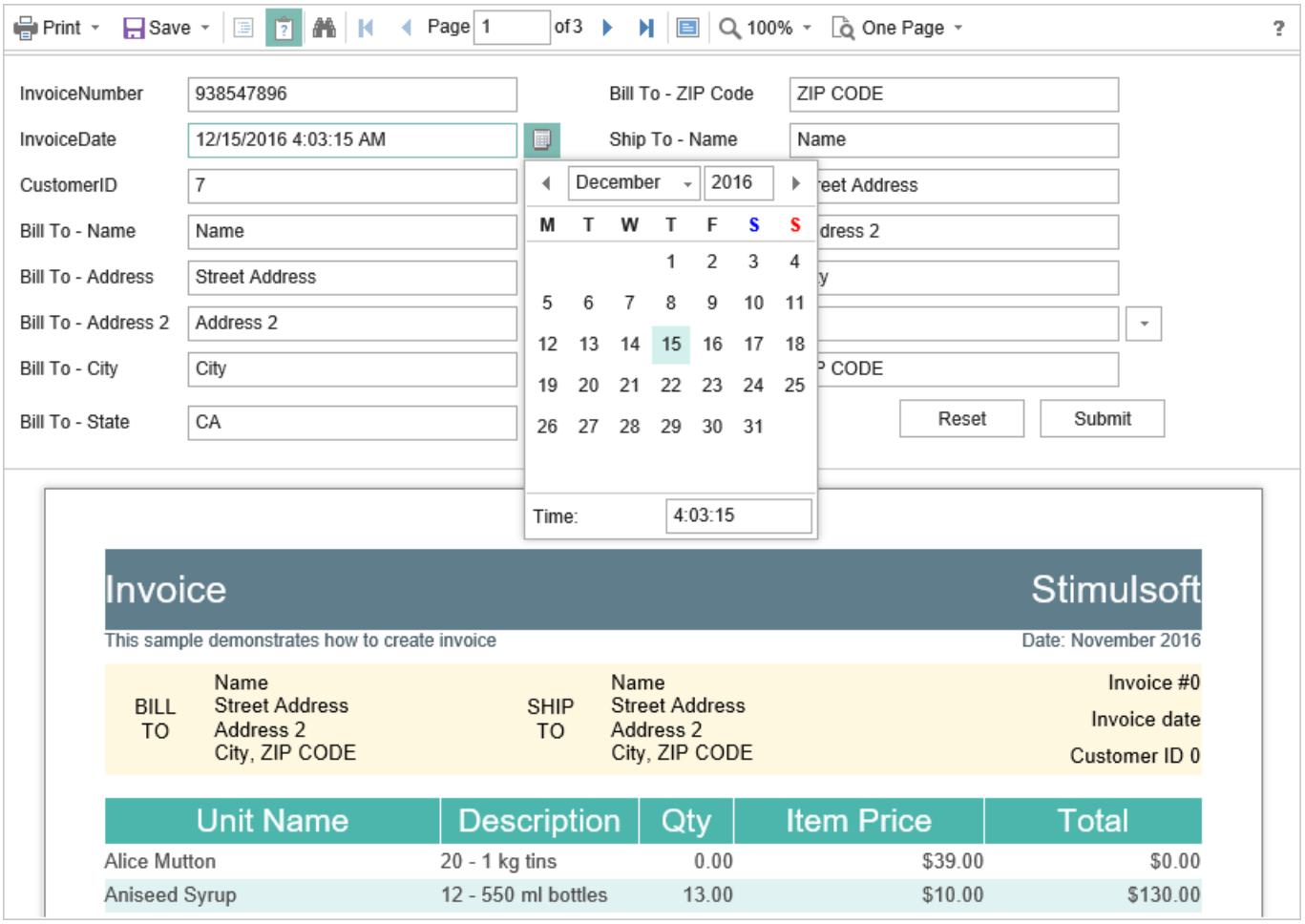
Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To work with report parameters in the **HTML5 Viewer**, there is a special settings panel. To add a parameter to the panel, you need to define a variable in a report requested by the user. When viewing a report in the viewer, such a variable will be

© 2003-2024 Stimulsoft

automatically added to the settings panel. It supports all report variables (normal variables, date and time, borders, lists, etc.).



To work with reports with parameters, no additional viewer settings are required. If you need to perform some actions before applying the parameters, you can define a special **Interaction** action.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         Interaction = "ViewerInteraction"
      }
})
...
```

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   // Some code before any interaction
   // ...

   return StiMvcViewer.InteractionResult();
}
...
```

This action is called during any interactive actions of the viewer. If you need to perform any actions only when applying report parameters, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side on any request and contain all necessary information and states of the client part of the viewer. To determine the type of action of the viewer, it is enough to check the Action property of the viewer parameters.

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   StiRequestParams requestParams = StiMvcViewer.GetRequestParams();
   if (requestParams.Action == StiAction.Variables)
   {
      // Some code before apply parameters
   }

   return StiMvcViewer.InteractionResult();
}
...
```

If you do not need to work with parameters, you can completely disable this feature. To do this, use the **ShowParametersButton** property in the **Toolbar** section of properties, which should be set to **false**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Toolbar =
      {
```

© 2003-2024 Stimulsoft

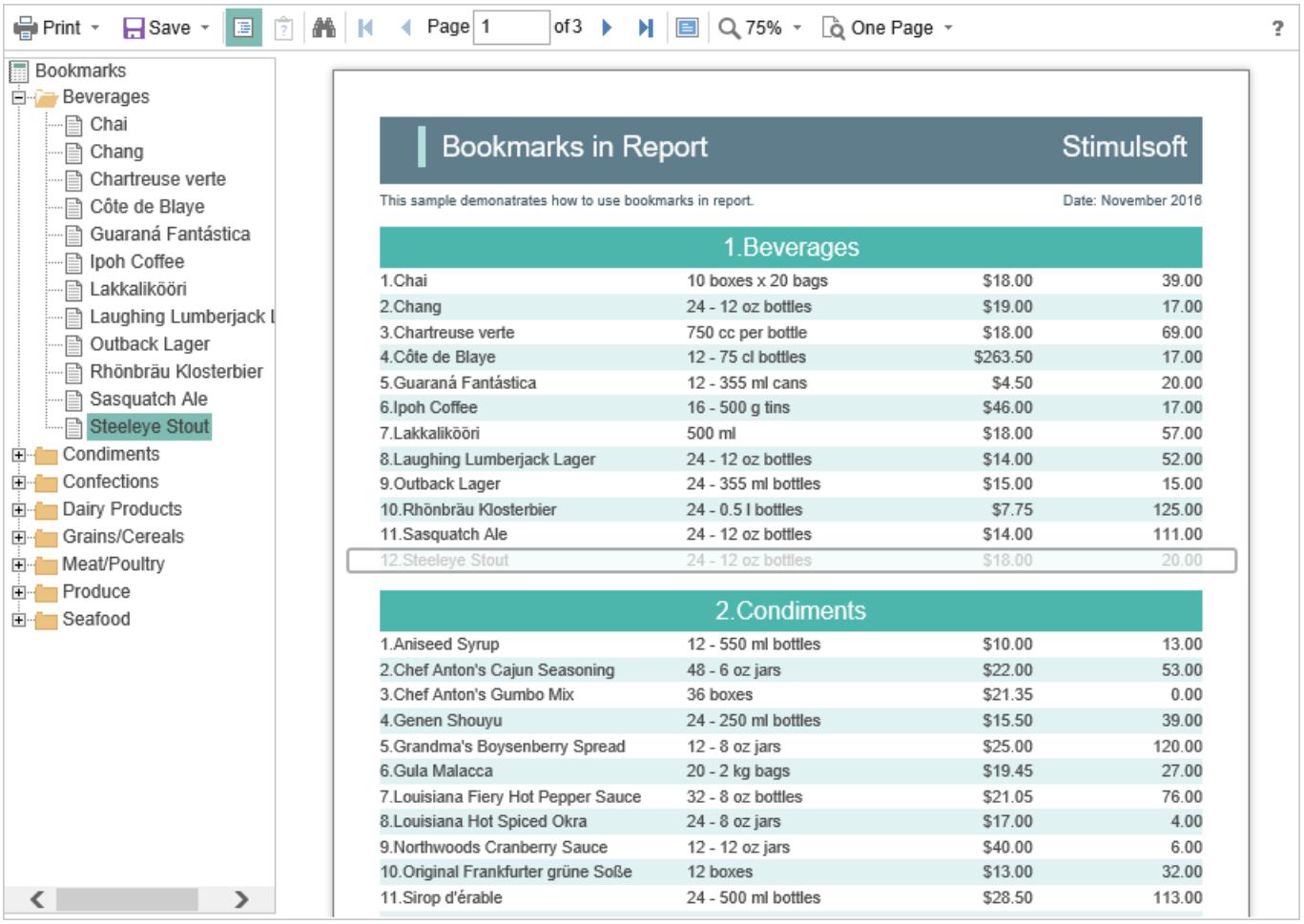
```
        ShowParametersButton = false
    }
})
...
```

Information

The options panel will not be displayed with such a viewer configuration, even if the parameters are present in the displayed report.

4.1.12 Work with Bookmarks

The **HTML5 Viewer** component supports report bookmarks. A panel with bookmarks will be displayed when displaying such a report on the left side of the page. When you select a bookmark of the report, the viewer will carry out an automatic transition to the specified page, and the report item with a bookmark is highlighted.



© 2003-2024 Stimulsoft

By default, the bookmarks bar width is 180 pixels. The **HTML5 Viewer** component allows you to change this value. For this, the **BookmarksTreeWidth** property, which value is specified in pixels, is used.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Appearance =
      {
         BookmarksTreeWidth = 200
      }
})
...
```

If work with report bookmarks is not required, you can disable this feature. For this, set the **ShowBookmarksButton** property to **false**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Toolbar =
      {
         ShowBookmarksButton = false
      }
})
...
```

Information

In this case, report bookmarks will not be displayed, even if they are present in the displayed report. This property has no effect on printing and exporting reports.

When printing a report with bookmarks, the bookmark tree will be hidden. If you want to print bookmarks with the report, it is necessary to set the **BookmarksPrint** property to **true**.

© 2003-2024 Stimulsoft

Index.cshtml

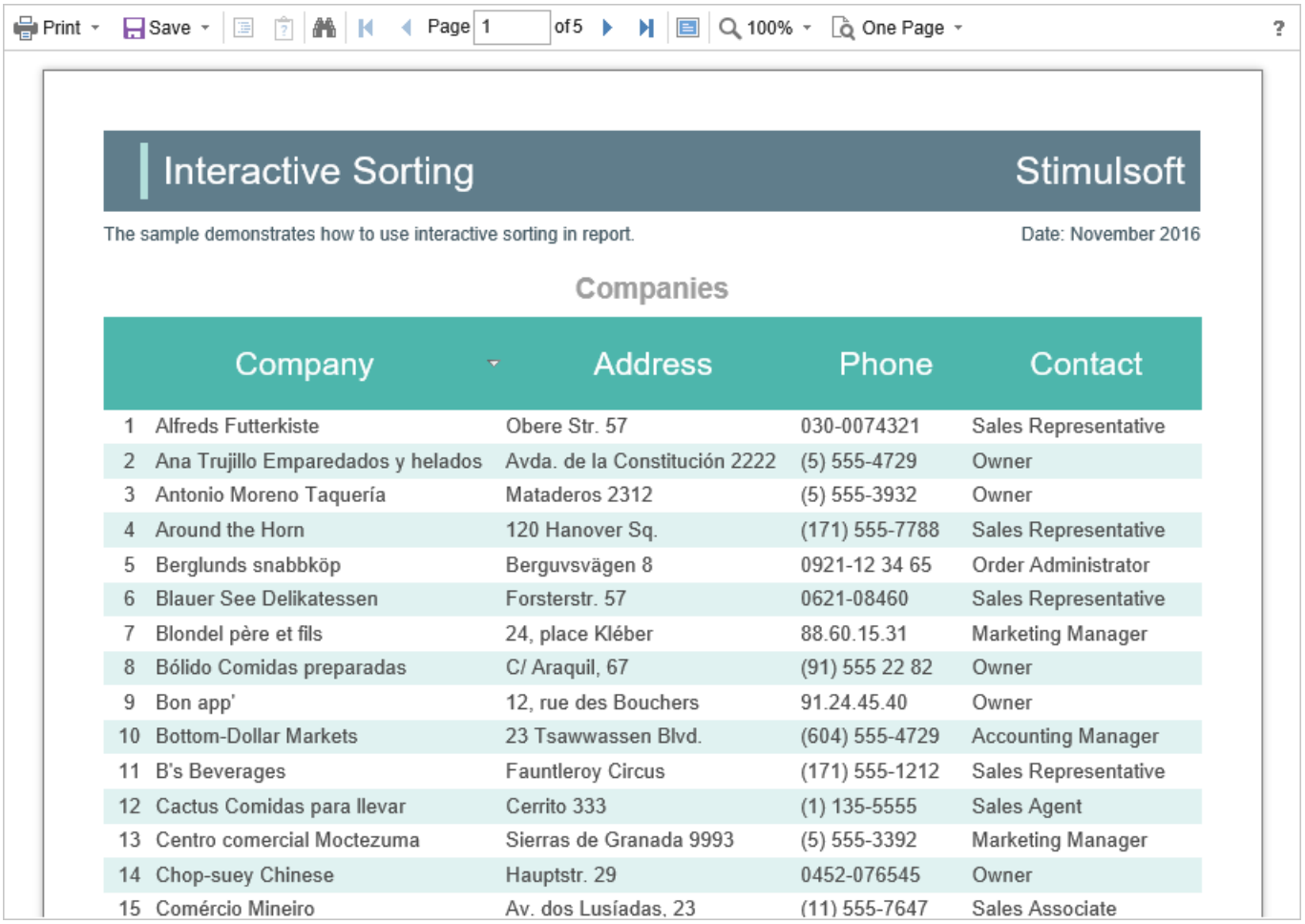
```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Appearance =
      {
         BookmarksPrint = true
      }
})
...
```

4.1.13 Dynamic Sorting, Collapsing, and Drill-Down

The **HTML5 Viewer** component supports dynamic sorting, collapsing, and drill-down of reports. Dynamic sorting provides the ability to change the direction of sorting in a rendered report. To do this, click on the component that has the dynamic sorting enabled. Dynamic sorting is carried out in the following directions - **Ascending** and **Descending**. Each time the component is clicked, the sorting direction is reversed.

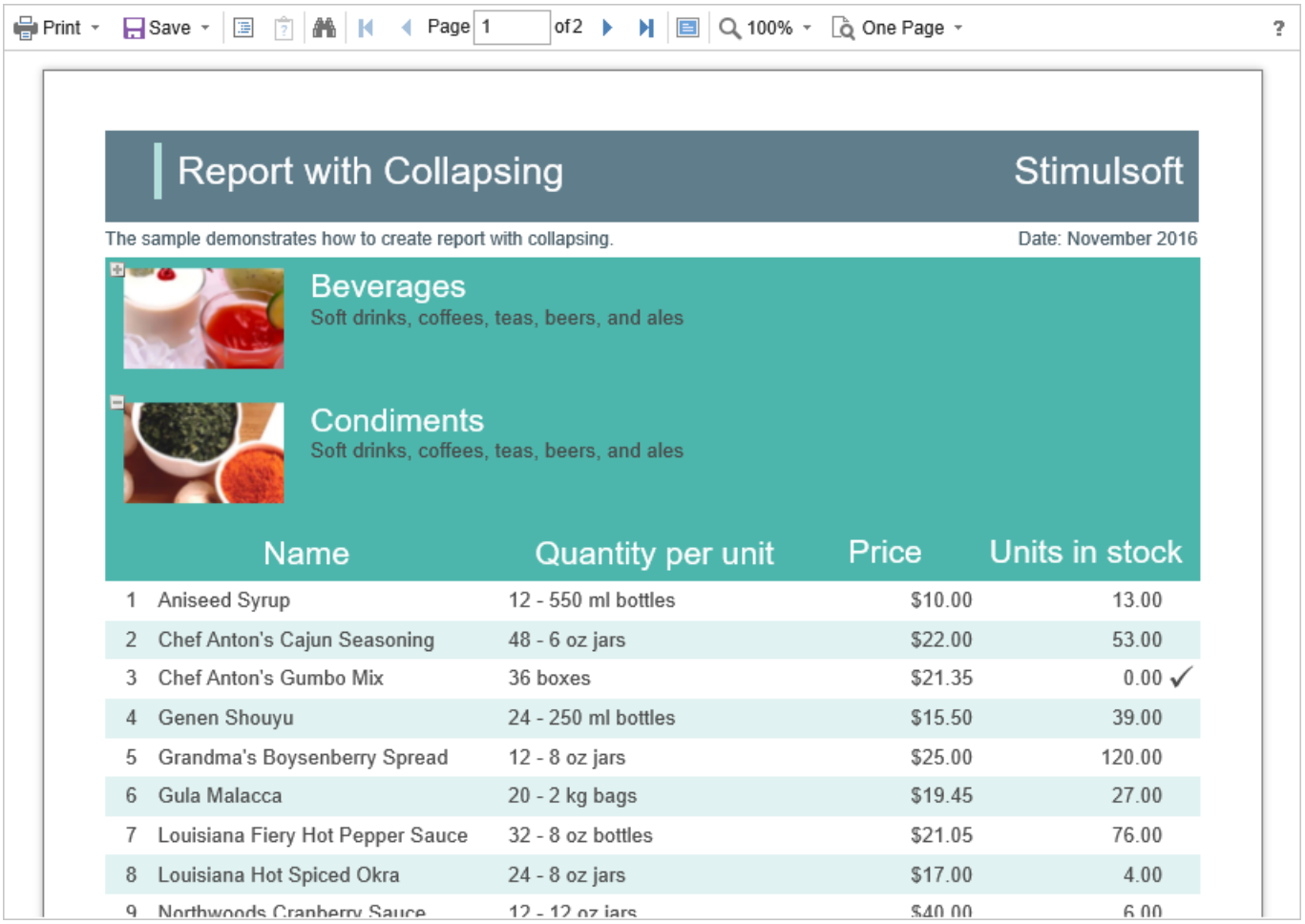
Multi-level sorting is allowed in the report. To do this, hold down the **Ctrl** key and sequentially click on the sorted components in the report. To reset sorting, you can click on any sorted component without holding down the **Ctrl** key.

© 2003-2024 Stimulsoft



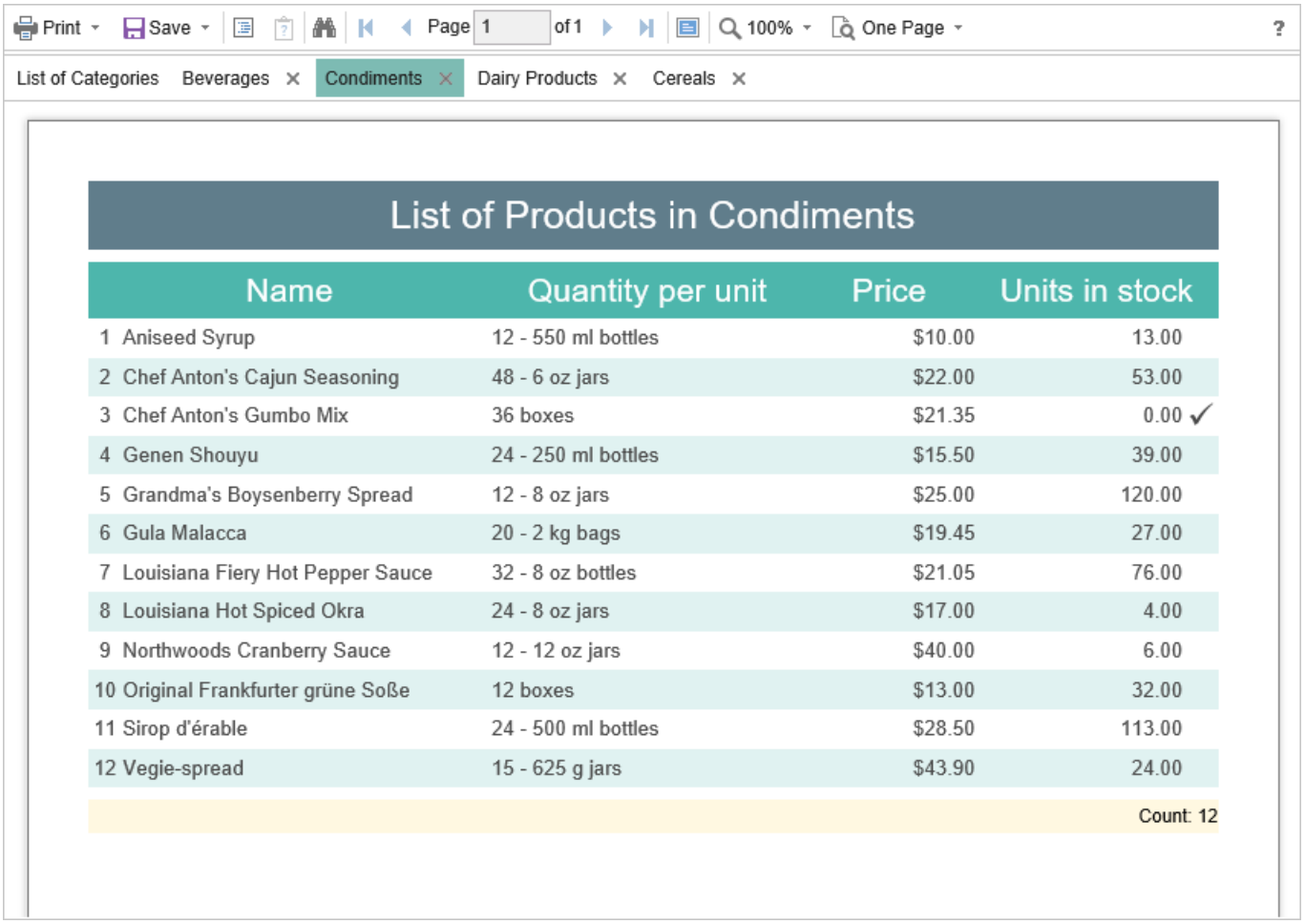
A report with dynamic collapsing is an interactive report in which blocks can collapse/expand their content when you click on the block title. Report elements, which can be collapsed/expanded, are indicated by special icons - **[-]** or **[+]**.

© 2003-2024 Stimulsoft



When using drill-down, under the main panel of the viewer, the drill-down panel with tabs for drill-down reports will be displayed. The currently displayed report will be highlighted.

© 2003-2024 Stimulsoft



To work with dynamic sorting, collapsing, and drill-down reports, no additional viewer settings are required. A special interaction action is used to perform any actions before sorting, collapsing, or drill-down. It will be called when interactive action of the viewer.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         Interaction = "ViewerInteraction"
      }
})
...
```

HomeController.cs

© 2003-2024 Stimulsoft

```
...
public ActionResult ViewerInteraction()
{
   // Some code before any interaction
   // ...

   return StiMvcViewer.InteractionResult();
}
...
```

To get the type of action, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side by any request and contain all necessary information and states of the client part of the viewer. For each type of interactivity, the viewer has a certain type of action:

- **Sorting** – when using column sorting;
- **DrillDown** – when using drill-down in reports;
- **Collapsing** – when using collapsing report blocks.

HomeController.cs

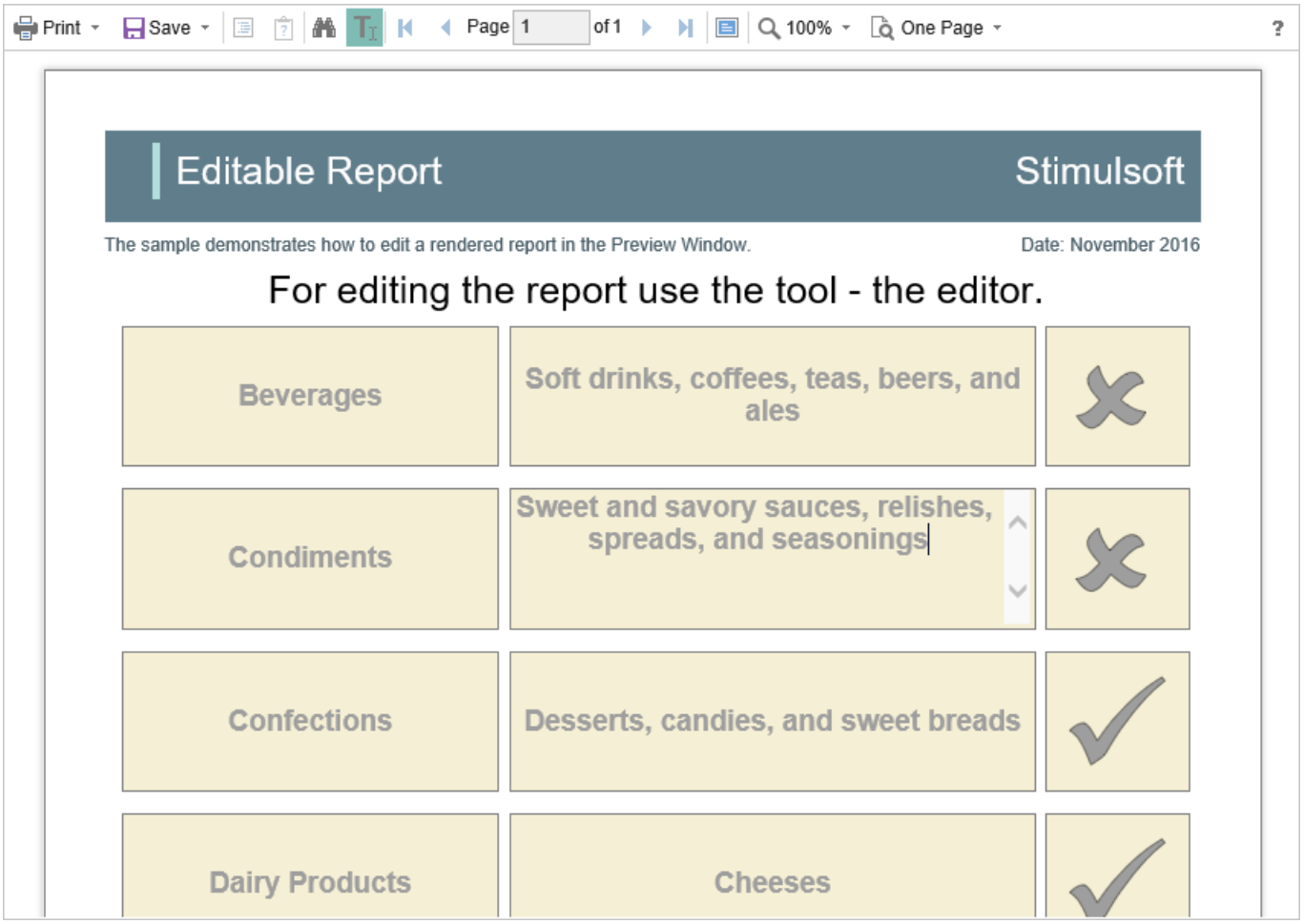
```
...
public ActionResult ViewerInteraction()
{
   StiRequestParams requestParams = StiMvcViewer.GetRequestParams();
   switch (requestParams.Action)
   {
      case StiAction.Sorting:
         break;

      case StiAction.DrillDown:
         break;

      case StiAction.Collapsing:
         break;
   }

   return StiMvcViewer.InteractionResult();
}
...
```

4.1.14 Editing Report

The **HTML5 Viewer** component has the ability to edit report items, such as text boxes and checkboxes. You should mark the required components as editable in the report template for the editing to be possible. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be

© 2003-2024 Stimulsoft

applied to the report.



For the report edit mode, no special settings of the viewer required.

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

4.1.15 Sending Report by Email

Information

Please note that the Send Report by Email option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides the ability to send reports by email. To activate this feature, you should set the **ShowSendEmailButton** property of the viewer to **true** and define the **EmailReport** action.

Index.cshtml

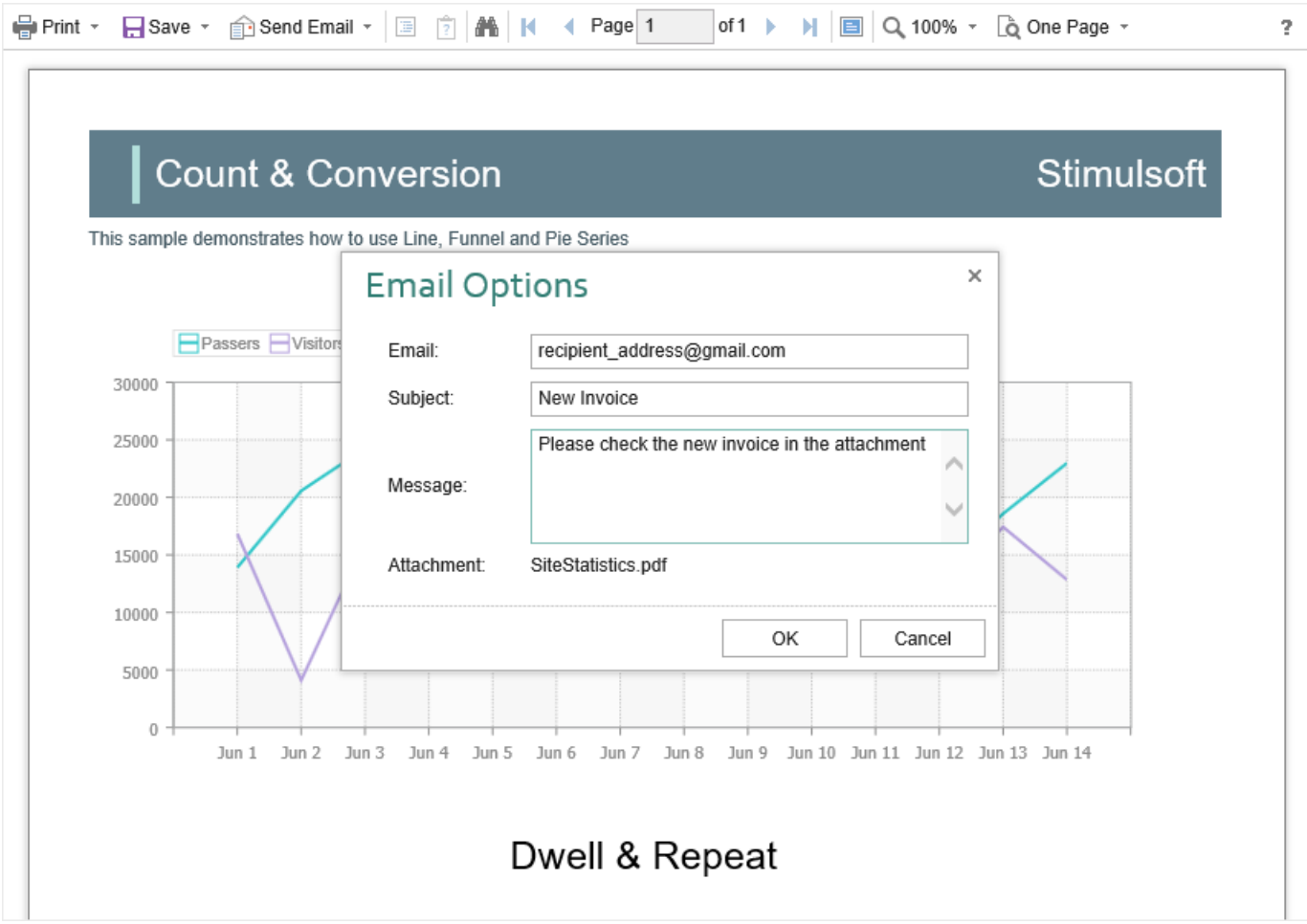
```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         EmailReport = "EmailReport"
      },
      Toolbar =
      {
         ShowSendEmailButton = true
      }
})
...
```

HomeController.cs

```
...
public ActionResult EmailReport()
{
   StiEmailOptions options = StiMvcViewer.GetEmailOptions();

   // Passed from the viewer, can be checked and changed
   // options.AddressTo = "";
   // options.Subject = "";
   // options.Body = "";

   // Should be filled here
   options.AddressFrom = "admin_address@test.com";
   options.Host = "smtp.test.com";
   options.Port = 465;
   options.UserName = "admin_address@test.com";
   options.Password = "admin_password";

   // options.CC.Add("email@test.com");
   // options.BCC.Add("email@test.com");
   // options.EnableSsl = true;

   return StiMvcViewer.EmailReportResult(options);
}
...
```

© 2003-2024 Stimulsoft

When sending a report by email, the menu to select the attachment format is displayed. It corresponds to the menu for selecting the format for exporting the report. After selecting the format, the dialog to send email parameters, such as the recipient's email, subject, and text of the message, is displayed.



After confirmation of sending the email, the above described **EmailReport** event will be called. You can check and correct the data entered in this form. The exported report file will be attached to the email automatically.

The **HTML5 Viewer** component allows you to set default values for the send email form. The **DefaultEmailAddress**, **DefaultEmailSubject**, and **DefaultEmailMessage** properties can be used for this. By default, these properties are empty.

© 2003-2024 Stimulsoft

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Email =
      {
         DefaultEmailAddress = "recipient_address@gmail.com",
         DefaultEmailSubject = "New Invoice",
         DefaultEmailMessage = "Please check the new invoice in the
         attachment"
      }
})
...
```

4.1.16 Calling Designer from Viewer

The **HTML5 Viewer** component has the ability to call the report designer. The special **Design** button in the toolbar of the viewer (the button is disabled by default) should be used. To use this feature, you should set the **ShowDesignButton** property to true and define the **DesignReport** event handler.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         DesignReport = "DesignReport"
      },
      Toolbar =
      {
         ShowDesignButton = true
      }
})
...
```

HomeController.cs

```
...
public ActionResult DesignReport()
{
   StiReport report = StiMvcViewer.GetReportObject();
   ViewBag.ReportName = report.ReportName;

   return View("Designer");
}
...
```

Information

© 2003-2024 Stimulsoft

The viewer does not run the designer. It only calls the specified action, in which you can get all the necessary parameters. Then, in action, you can implement a redirection to another View, which contains the report designer.

4.1.17 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to use the server cache to store rendered reports. If you do not use caching, you should load the report, connect data, and render it again every time you request a page. If you use caching, the previously rendered report will be loaded from the cache every time you refresh the page.

When using caching, it should be taken into account that every report saved in the cache takes up server memory and, with a large number of requests to reports, this can become a critical issue. Therefore, you need to choose between two options: low memory requirements but high in performance or low-performance requirements but high in memory.

You can manage caching with the following properties.

The CacheMode property

This property of the viewer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled. Each action of the viewer requires loading the report from the file and, if it is a report template, then render it;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this cache (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a

© 2003-2024 Stimulsoft

packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** – for caching, the current session is used. The report is saved as a packed string in this cache.

The CacheItemPriority property

This property sets the priority of the report stored in the server's cache. It affects the automatic clearing of the server memory in case of a lack of memory. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes for which you want to save the report in the server cache. If you use caching and the requested report is not found in the cache (the objects storage time has expired), it will be requested again using a special **GetReport** event, then connect the report data and render it.

StiCacheHelper

The **HTML5 Viewer** component provides the ability to define your methods of working with report caching. For this purpose, a special class **StiCacheHelper** is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetReport** and **SaveReport**.

HomeController.cs

```
...
public class ViewerController : Controller
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override StiReport GetReport(string guid)
      {
```

© 2003-2024 Stimulsoft

```
        string path =
        System.IO.Path.Combine(System.Web.HttpContext.Current.Server.MapPat
        h("~/"), "CacheFiles", guid);
        if (System.IO.File.Exists(path))
        {
          StiReport report = new StiReport();
          string packedReport = System.IO.File.ReadAllText(path);
          if (guid.EndsWith("template"))
          report.LoadPackedReportFromString(packedReport);
          else report.LoadPackedDocumentFromString(packedReport);

          return report;
        }
        return null;

        //return base.GetReport(guid);
      }

      public override void SaveReport(StiReport report, string guid)
      {
        string packedReport = guid.EndsWith("template") ?
        report.SavePackedReportToString() :
        report.SavePackedDocumentToString();
        string path =
        System.IO.Path.Combine(System.Web.HttpContext.Current.Server.MapPat
        h("~/"), "CacheFiles", guid);
        System.IO.File.WriteAllText(path, packedReport);

        //base.SaveReport(report, guid);
      }
    }

    static ViewerController()
    {
      StiMvcViewer.CacheHelper = new StiMyCacheHelper();
    }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as a value of the static **StiMvcViewer.CacheHelper** property in the controller constructor.

Information

If report caching is disabled (the **CacheMode** property of the viewer is set to **None**), the specified class will not be used.

© 2003-2024 Stimulsoft

4.1.18 Additional Methods

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

For **HTML5 Viewer**, several additional methods are used to get the object of the currently viewed report, parameters of the viewer's current state, and other useful data. These methods can be used in any actions of the viewer.

The GetReportObject() method

Returns the report object with which the viewer is currently working. It is possible to perform the necessary actions with it - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the viewer, specifying it as a parameter in the resulting action method.

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   StiReport report = StiMvcViewer.GetReportObject();
   report.ReportName = "MyReportName";

   return StiMvcViewer.InteractionResult(report);
}
...
```

The GetRouteValues() method

Returns values for URLs with which the viewer page was opened. Thus, it is possible to get the initial collection of run page parameters in any viewer action and use these values for any checks and conditions.

HomeController.cs

© 2003-2024 Stimulsoft

```
...
public ActionResult ViewerInteraction()
{
   RouteValueDictionary routeValues = StiMvcViewer.GetRouteValues();

   return StiMvcViewer.InteractionResult();
}
...
```

You can also get values of URL parameters by parameter name, specifying it as the parameter of the called action of the viewer.

HomeController.cs

```
...
public ActionResult ViewerInteraction(string id)
{
   return StiMvcViewer.InteractionResult();
}
...
```

The GetFormValues() method

Returns the values of the form that initiated (opened by the POST request) a page of the viewer. Thus, it is possible to get a collection of form parameters in any action of the viewer.

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   NameValueCollection formValues = StiMvcViewer.GetFormValues();

   return StiMvcViewer.InteractionResult();
}
...
```

By default, this feature is disabled to optimize requests of the client-side of the viewer to the server. To enable it, set the **PassFormValues** property to **true**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
```

© 2003-2024 Stimulsoft

```
   new StiMvcViewerOptions() {
      Server =
      {
         PassFormValues = true
      }
})
...
```

The GetRequestParams() method

Returns all parameters of the current state of the viewer passed to the server-side. They can be useful for determining the type of action that the viewer is currently executing - for example, to determine the type of export, and all action parameters.

HomeController.cs

```
...
public ActionResult ExportReport()
{
   StiRequestParams requestParams = StiMvcViewer.GetRequestParams();
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiMvcViewer.GetReportObject();

      // Some action with report for the PDF export
      // ...

      return StiMvcViewer.ExportReportResult(report);
   }

   return StiMvcViewer.ExportReportResult();
}
...
```

You can change the values of some parameters. After making changes, for the correct operation of the viewer, you should transfer the modified parameter object to the input of the resulting method.

HomeController.cs

```
...
public ActionResult ViewerInteraction()
{
   StiRequestParams requestParams = StiMvcViewer.GetRequestParams();
   if (requestParams.Action == StiAction.Variables)
   {
      requestParams.Interaction.Variables["Variable1"] = "MyValue";
      return StiMvcViewer.InteractionResult(requestParams);
   }
```

© 2003-2024 Stimulsoft

```
   return StiMvcViewer.InteractionResult();
}
...
```

The GetExportSettings() method

Returns all the parameters of the current report export. The type of the parameter object will correspond to the type of export selected in the viewer menu. Any export parameters can be changed and passed to the input of the resulting method. In this case, the report will be exported with the parameters transferred.

HomeController.cs

```
...
public ActionResult ExportReport()
{
   StiExportSettings settings = StiMvcViewer.GetExportSettings();
   if (settings.GetExportFormat() == StiExportFormat.Pdf)
   {
      StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
      pdfSettings.EmbeddedFonts = true;
      pdfSettings.AllowEditable = StiPdfAllowEditable.No;
      return StiMvcViewer.ExportReportResult(settings);
   }

   return StiMvcViewer.ExportReportResult();
}
...
```

4.1.19 Export and Printing from Code

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** provides the ability to print reports in various ways and export reports to various formats. These actions are performed using the viewer menu. If you want to print or export a report by using the code, for example, in the controller action, you can use the special **StiMvcReportResponse** class. This class contains a set of static methods that allow you to print or export a report from the code, and

© 2003-2024 Stimulsoft

the report viewer is not required.

Index.cshtml

```
...
@Html.ActionLink("Print Report from Code", "PrintReport")
<br />
@Html.ActionLink("Export Report from Code", "ExportReport")
...
```

HomeController.cs

```
...
private StiReport LoadSimpleList()
{
   DataSet dataSet = new DataSet();
   dataSet.ReadXml(Server.MapPath("Reports/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));
   report.RegData(dataSet);

   return report;
}

public ActionResult PrintReport()
{
   StiReport report = LoadSimpleList();

   return StiMvcReportResponse.PrintAsPdf(report);
   //return StiMvcReportResponse.PrintAsHtml(report);
}

public ActionResult ExportReport()
{
   StiReport report = LoadSimpleList();

   return StiMvcReportResponse.ResponseAsPdf(report);
   //return StiMvcReportResponse.ResponseAsExcel2007(report);
   //return StiMvcReportResponse.ResponseAsText(report);
   //StiMvcReportResponse.ResponseAsJson(report);
}
...
```

The **StiMvcReportResponse** class contains methods for printing in PDF and HTML formats and methods to export the report in any of the supported formats. As arguments, methods can take various export settings, displaying modes and options for saving received files.

© 2003-2024 Stimulsoft

4.1.20 Timeout

When working with the **StiMvcViewer** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property
Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Server =
      {
         CacheTimeout = 10
      }
})
...
```

Using the cache will increase the speed of the report viewer. See the chapter Caching for more information

RequestTimeout Property
Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Server =
      {
         RequestTimeout = 30
      }
})
...
```

© 2003-2024 Stimulsoft

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection and data sources in the report.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Actions =
      {
         GetReport = "GetReport",
         ViewerEvent = "ViewerEvent"
      }
})
...
```

HomeController.cs

```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiMvcViewer.GetReportResult(report);
}

public ActionResult ViewerEvent()
{
   return StiMvcViewer.ViewerEventResult();
}
...
```

4.1.21 Viewer Settings

The **HTML5 Viewer** is configured using properties that are located in the **StiMvcViewerOptions** class. All properties are divided into groups. Some of the groups contain subgroups for ease of use. The following is an example of setting the properties of the viewer.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcViewer("MvcViewer1",
   new StiMvcViewerOptions() {
      Theme = StiViewerTheme.Office2022WhiteTeal,
      Localization = "~/Content/Localization/en.xml",
      Actions =
      {
         GetReport = "GetReport",
         ViewerEvent = "ViewerEvent"
      },
      Appearance =
      {
         InterfaceType = StiInterfaceType.Auto,
         ScrollbarsMode = true,
         ShowTooltips = false
      },
      Exports =
      {
         DefaultSettings =
         {
            ExportToPdf =
            {
               CreatorString = "Company Name",
               ImageQuality = 0.75f
            }
         },
         ShowExportToDbf = false,
         ShowExportToDif = false
      }
})
...
```

Please note that all dashboard elements have their own save options and full-screen buttons for preview. There are no special options to control displaying them, but they can be disabled through the properties of the element. The code below should be added after loading the report before passing it to the viewer.

Default.aspx.cs

```
...
var dbsElementInteraction = (report.GetComponentByName("RegionMap1") as
Stimulsoft.Report.Dashboard.IStiElementInteraction).DashboardInteraction;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowFullScreenButton =
false;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowSaveButton = false;
...
```

Main settings (without groups)

© 2003-2024 Stimulsoft

Name	Description
Theme	Sets the viewer theme. The list of available themes can be found in the **StiViewerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Sets the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used. It is built into the viewer and does not require additional XML files.
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default width is 100%.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the automatic height is set depending on the size of the report page, or 650 pixels in the view mode of the viewer with scrollbars.

Actions

Name	Description
GetReport	Specifies the name of the action method for preparing the rendered report. Specifies the name of the action method for preparing the constructed report. If report caching is enabled, this action will be called only once when the report is requested or if the requested report is not found in the server cache.

© 2003-2024 Stimulsoft

PrintReport	Specifies the name of the action method of report printing. This is not relevant when viewing dashboards.
ExportReport	Specifies the name of the action method of the export the report to the specified format.
EmailReport	Specifies the name of the action method of sending the report by email. This is not relevant when viewing dashboards.
Interaction	Specifies the name of the action method for the viewer to work with interactive operations, such as using parameters, dynamic sorting, collapsing, and drill-down.
DesignReport	Specifies the name of the action method to go to the specified view by clicking the Design button on the viewer panel.
ViewerEvent	Specifies the name of the action method of basic viewer events and the processing actions of the viewer, such as printing and exporting a report, working with parameters, and interactivity, if these actions are not specified separately. In addition, this action is used to load scripts and styles of the viewer. This action is mandatory.

Server

Name	Description
Controller	Specifies the name of the report controller for the report viewer. If this property is not specified, then the current controller will be used to process requests.
RouteTemplate	Sets the route template that is returned when the report viewer actions are executed. If the property is not set, then the MVC project template will be used instead. The default value

© 2003-2024 Stimulsoft

	of the property is null.
RequestTimeout	Sets the response timeout from the server in seconds, after which an error will be generated. The default value is 20 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the report since the last action of the viewer. The default value is 20 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled. The report will be reloaded each time using the **GetReport** event; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as storage, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of a lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or the server session will be automatically re-saved after a certain period of time when the viewer is idle (every 3 minutes). By default, the property is set to **true**.

© 2003-2024 Stimulsoft

UseRelativeUrls	Sets the viewer mode in which relative URLs are used for AJAX requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value that specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the viewer. If **false**, only the necessary parameters are used to request the resources of the viewer. This corresponds to the correct work of the browser cache. By default, the property is set to **true**.
PassQueryParametersToReport	Enables using all the URL parameters of the request as the variable values. The variable names must match the parameters. The default value of the property is **false**.
PassFormValues	Enables passing the values of the POST form to the client-side if these values are required to be used in the actions of the viewer. If you enable this property, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
ShowServerErrorPage	Enables displaying an HTML page with the details of the error that occurred on the server-side. When the property is enabled, the details of the error will be displayed in the viewer window. If the property is disabled, only the numeric error code and a short error text in the dialog box will be displayed. By default, the property is set to **true**.
UseCompression	Enables compression of the viewer requests into the GZip stream. That allows to decrease the amount of internet traffic but slows down the viewer slightly. The default value of the property is **false**.

© 2003-2024 Stimulsoft

UseCacheForResources	Enables caching of the component resources on the server-side. The following resources are supported: scripts, styles, and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
UseLocalizedCache	Sets a value that enables the use of a different cache depending on the selected localization. The default value of the property is **false**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Sets the path to the CSS file of the viewer's styles. The standard styles of the chosen theme will not be loaded if this property has got a value. The default value of the property is an empty string.
BackgroundColor	Sets the background color of the viewer. By default, it is set to **White**.
PageBorderColor	Sets the border color of the viewer. By default, it is set to **Gray**.
RightToLeft	Sets the **Right to Left** mode for viewer controls. By default, the property is set to **false**.
FullScreenMode	Sets the full-screen display mode of the viewer. By default, the property is set to **false**.
ScrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to **false**.
OpenLinksWindow	Sets the target window for opening links

© 2003-2024 Stimulsoft

	contained in the report. By default, the property is set to **Blank** (new window).
OpenExportedReportWindow	Sets the target window for opening the export file from the viewer. By default, the property is set to **Blank** (new window).
DesignWindow	Sets the destination window for launching the report designer. The default value of the property is **Self** (which is the current window).
ShowTooltips	Enables showing tips for the viewer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enables showing links to online documentation for the viewer controls. By default, the property is set to **true**.
ShowDialogsHelp	Sets a value that indicates that showing or hiding the help button in dialogs. By default, the property is set to **true**.
PageAlignment	Sets the position of the report page in the viewer window. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – the page will be aligned left; › **Center** – the page will be centered (default value); › **Right** – the page will be aligned right.
ShowPageShadow	Enables displaying shadow for report pages. By default the property is set to **true**.
BookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default, the property is set to **false**.
BookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
ParametersPanelPosition	Specifies the position of the report parameters panel. It can take one of the following

© 2003-2024 Stimulsoft

	StiParametersPanelPosition enumeration values: › **Top** - the panel will be docked to the top margin (default value); › **Left** - the panel will be docked to the left margin.
ParametersPanelMaxHeight	Sets the maximum height of the parameters bar in pixels. By default, the maximum height is 300 pixels.
ParametersPanelColumnsCount	Sets the number of columns to display report parameters. By default, there are two columns.
ParametersPanelSortDataItems	Gets or sets a value that indicates that variable items will be sorted. By default, the property is set to **true**.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
InterfaceType	Sets the type of interface used for the viewer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the viewer's interface is determined automatically depending on the device the report is displayed on. That is the default value. › **Mouse** – the standard interface with a mouse control will be used for all the screen types. › **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability. › **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the

© 2003-2024 Stimulsoft

	viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.
AllowMobileMode	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to **false**, then the mobile view will not be used. If the option is set to **true**, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option is set to **true**.
ChartRenderType	Sets the displaying mode of charts on the report page. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages. It can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
DatePickerFirstDayOfWeek	Sets the first day of the week for the date picker. It can take one of the following values of the **StiFirstDayOfWeek** enumeration:

© 2003-2024 Stimulsoft

	› **Monday** – the first day of the week is Monday (default value); › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
AllowTouchZoom	Sets ability to change the scale of the report page by using the two-fingers gesture (Pinch to Zoom) for the touch-screens. The default value of the property is **true**.
ShowReportIsNotSpecifiedMessage	Sets a value that indicates that 'The report is not specified' message will be shown. The default value of the property is **true**.
PrintToPdfMode	Sets the Print to PDF mode. It has the following values: › **StiPrintToPdfMode.Hidden** - hidden print mode (default value); › **StiPrintToPdfMode.Popup** - the PDF document will be displayed before printing in a pop-up window.
ImagesQuality	Gets or sets the image quality that will be used on the viewer page. It has the following values: › **StiImagesQuality.Low** - low quality, used to speed up loading reports and saves memory; › **StiImagesQuality.Normal** - normal quality, suitable for most cases (default value); › **StiImagesQuality.High** - high quality, used for ultra-high-definition displays, but may slow down the loading of pages.
CombineReportPages	Sets a value that indicates that if a report contains several pages, then they will be combined in preview. By default, the property is set to **false**.

Toolbar

© 2003-2024 Stimulsoft

Name	Description
Visible	Enables displaying the viewer toolbar. By default, the property is set to **true**.
DisplayMode	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the **StiToolbarDisplayMode** enumeration: › **Simple** - all controls are located on the same control panel (default value); › **Separated** - the control panel is split into top and bottom panels.
BackgroundColor	Specifies the background color of the viewer toolbar. The default color of the selected theme is used.
BorderColor	Specifies the border color of the viewer toolbar. The default color of the selected theme is used.
FontColor	Specifies the text color for the toolbar and the viewer menu. The default color of the selected theme is used.
FontFamily	Specifies the font for the toolbar and the viewer menu. The default font of the selected theme is used.
Alignment	Sets the alignment mode for the controls on the viewer toolbar. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – elements will be aligned left; › **Center** – elements will be centered; › **Right** – elements will be aligned right; › **Default** – the alignment depends on the RightToLeft property (default value).
ShowButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default, the property is set to **true**.
ShowPrintButton	Enables showing the button - **Print** - on the viewer toolbar. By default, the property is set to

© 2003-2024 Stimulsoft

	true.
ShowOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to true.
ShowSendEmailButton	Enables showing the button - **Send Email** - on the viewer toolbar. By default, the property is set to **false**. Also, you should add the EmailReport action.
ShowFindButton	Enables showing the button - **Find** - on the viewer toolbar. By default, the property is set to **true**.
ShowBookmarksButton	Enables showing the button - **Bookmarks** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
ShowParametersButton	Enables showing the button - **Parameters** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.
ShowResourcesButton	Enables showing the button - **Resources** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.
ShowEditorButton	Enables showing the button - **Editor** - on the viewer toolbar. By default, the property is set to **true**.
ShowFullScreenButton	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports

© 2003-2024 Stimulsoft

	or dashboards. By default, the property is set to **true**.
ShowFirstPageButton	Enables showing the button - **First Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowPreviousPageButton	Enables showing the button - **Previous Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowCurrentPageControl	Enables showing the current report page indicator. By default, the property is set to **true**.
ShowNextPageButton	Enables showing the button - **Next Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowLastPageButton	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowZoomButton	Enables showing the button to select the report zoom. By default, the property is set to **true**.
ShowViewModeButton	Enables showing the button to select the view mode of the report page. By default, the property is set to **true**.
ShowDesignButton	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **false**.
ShowAboutButton	Enables showing the button - **About** - on the viewer toolbar. By default, the property is set to **true**.
ShowRefreshButton	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to **true**.
ShowPinToolbarButton	Enables displaying of the **Pin Toolbar** button on the viewer's toolbar. The button is available only in the Mobile mode of the viewer's interface. The default value of the property is

© 2003-2024 Stimulsoft

	true.
PrintDestination	Sets the report printing mode. It can take one of the following values of the **StiPrintDestination** enumeration: › **Default** – a menu with a choice of printing modes will be displayed (default value); › **Pdf** – printing will be done in the PDF format; › **Direct** – printing will be done to the HTML format directly to the printer, the system print dialog will be displayed; › **PopupWindow** – printing will be done in the HTML format via the preview window of the report.
ViewMode	Sets the mode for displaying report pages. It can take one of the following **StiWebViewMode** enumeration values: › **SinglePage** - displays one page of the report selected in the toolbar of the viewer (default value); › **Continuous** - displays all pages of the report; › **MultiplePages** - displays all report pages as a table.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of the following values: › **StiZoomMode.PageWidth** – when the viewer runs, the zoom necessary to display the report by the page width will be set; › **StiZoomMode.PageHeight** – when the viewer runs, the zoom necessary to display the report by the page height will be set.
MenuAnimation	Enables animation when the viewer menu shows/hides. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowMenuMode	Sets the display mode of the viewer menu. It can take one of the following values of the **StiShowMenuMode** enumeration: › **Click** – shows menu by mouse click (default value); › **Hover** – shows menu by hovering the mouse cursor.
AutoHide	Enables auto-hiding of the viewer's toolbar. The property will work only for the Mobile mode of the viewer's interface. The default value of the property is **false**.

Export

Name	Description
DefaultSettings	This group of properties provides the ability to specify the default export settings for each export type. These settings will be applied to the export dialogs when the viewer runs or to the report if export dialogs are disabled.
StoreExportSettings	Enables saving selected settings in the export dialogs. Settings will be stored in browser cookies. By default, the property is set to **true**.
ShowExportDialog	Enables showing the export options dialog box. If the property is set to **false**, the export will be done with the default settings. By default, the property is set to **true**.
ShowExportToDocument	Enables the export menu item - **Document File**. By default, the property is set to **true**.
ShowExportToPdf	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToXps	Enables the export menu item - **Microsoft XPS**

© 2003-2024 Stimulsoft

	File. By default, the property is set to **false**.
ShowExportToPowerPoint	Enables the export menu item - **Microsoft PowerPoint 2007/2010 File**. By default, the property is set to **true**.
ShowExportToHtml	Enables the export menu item - **HTML File**. By default, the property is set to **true**.
ShowExportToHtml5	Enables the export menu item - **HTML5 File**. By default, the property is set to **true**.
ShowExportToMht	Enables the export menu item - **MHT Web Archive**. By default, the property is set to **true**.
ShowExportToText	Enables the export menu item - **Text File**. By default, the property is set to **true**.
ShowExportToRtf	Enables the export menu item - **Rich Text File**. By default, the property is set to **true**.
ShowExportToWord2007	Enables the export menu item - **Microsoft Word 2007/2010 File**. By default, the property is set to **true**.
ShowExportToOpenDocumentWriter	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to **true**.
ShowExportToExcel	Enables the export menu item - **Microsoft Excel File**. By default, the property is set to **true**.
ShowExportToExcelXml	Enables the export menu item - **Microsoft Excel Xml File**. By default, the property is set to **true**.
ShowExportToExcel2007	Enables displaying the **Microsoft Excel 2007/2010 File**. export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToOpenDocumentCalc	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to **true**.
ShowExportToCsv	Enables the export menu item - **CSV File**. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
ShowExportToDbf	Enables the export menu item - **DBF File**. By default, the property is set to **true**.
ShowExportToXml	Enables the export menu item - **XML File**. By default, the property is set to **true**.
ShowExportToDif	Enables the export menu item - **Data Interchange Format (DIF) File**. By default, the property is set to **true**.
ShowExportToSylk	Enables the export menu item - **Symbolic Link (SYLK) File**. By default, the property is set to **true**.
ShowExportToJson	Enables the export menu item - **JSON File**. By default, the property is set to **true**.
ShowExportToImageBmp	Enables displaying the **BMP Image** export menu item when viewing reports and the **BMP Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageGif	Enables displaying the **GIF Image** export menu item when viewing reports and the **GIF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageJpeg	Enables displaying the **JPEG Image** export menu item when viewing reports and the **JPEG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePcx	Enables displaying the **PCX Image** export menu item when viewing reports and the **PCX Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePng	Enables displaying the **PNG Image** export menu item when viewing reports and the **PNG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageTiff	Enables displaying the **TIFF Image** export menu item when viewing reports and the **TIFF Image**

© 2003-2024 Stimulsoft

	item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageMetafile	Enables displaying the **Windows Metafile** export menu item when viewing reports and the **Windows Metafile** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvg	Enables displaying the **Scalable Vector Graphics (SVG) File** export menu item when viewing reports and the **Scalable Vector Graphics (SVG) File** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvgz	Enables displaying the **Compressed SVG (SVGZ) File** export menu item when viewing reports and the **Compressed SVG (SVGZ) File** item when viewing dashboards. By default, the property is set to **true**.
ShowOpenAfterExport	Enables displaying the **Open After Export** parameter in export settings menu. By default the property is set to **true**.

Email

Name	Description
ShowEmailDialog	Enables displaying settings for sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server-side in the **EmailReport** action. By default, the property is set to **true**.
ShowExportDialog	Enables displaying export options dialog box when sending an email. If the property is set to **false**, the export will be done with the default settings. By default, the property is set to **true**.
DefaultEmailAddress	Sets the default recipient email, i.e., the address

© 2003-2024 Stimulsoft

	to which the email with the attached report will be sent.
DefaultEmailSubject	Sets the default email subject (header).
DefaultEmailMessage	Sets the default email message (text).

4.2 HTML5 Designer

YouTube

Watch videos for working with ASP.NET MVC HTML5 Designer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

Samples

See on GitHub examples of working with the ASP.NET MVC HTML5 Designer component. All examples are separate projects grouped into one solution for Visual Studio.

The **HTML5 Designer** (**StiMvcDesigner**) component is designed to create reports in the web browser. You do not need to install the .NET Framework, ActiveX components, or any special plug-ins on the client-side. All that is needed is any modern Web browser.

With the help of **HTML5 Designer**, you can create, edit, save and preview reports on any computer with any operating system installed. Since the designer only uses HTML and JavaScript technologies, it can be run on devices without Flash or Silverlight support - tablets, smartphones. Also, the designer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Designer** component uses the AJAX technology to perform all actions on reports, which allows you to get rid of reloading the entire page, save Web traffic, and speed up work.

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use **HTML5 Designer** in a Web-project, the installation of the Stimulsoft.Reports.Web NuGet package is required:
- Select 'Manage NuGet Packages...' menu item in the project's pop-up menu;
- In the 'Browse' tab, type 'Stimulsoft.Reports.Web' in the search textbox;
- Click the Stimulsoft.Report.Web package, select the version of the package, and click **Install**. If the package should be updated, use the **Update** button.

If for some reason, that is not possible, the following assemblies should be added to the project:
- Stimulsoft.Base.dll
- Stimulsoft.Report.dll
- Stimulsoft.Report.Check.dll
- Stimulsoft.Report.Helper.dll
- Stimulsoft.Report.Mvc.dll
- Stimulsoft.Report.Web.dll
- Stimulsoft.Report.WebDesign.dll

To add the ability to create and edit dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web (this package is associated with the package Stimulsoft.Reports.Web. If it is missed it will be installed automatically):
- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Dashboards.Web in the search bar on the Browse tab;
- Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

If, for any reason this is not possible, you should additionally add the following assemblies to the project:
- Stimulsoft.Dashboard.dll
- Stimulsoft.Dashboard.Drawing.dll
- Stimulsoft.Dashboard.Export.dll

© 2003-2024 Stimulsoft

- How this Works?
- Activation
- Editing Reports and Dashboards
- Creating New Reports and New Dashboards
- Saving Reports and Dashboards
- Preview
- Settings
- Additional Features of Preview
- Timeout
- Localization
- Using Themes
- Caching
- Additional Methods

4.2.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the designer, you need to place the **StiMvcDesigner** component on the page, set the necessary settings to it, and set the necessary actions in the view controller. When the report designer runs, the following actions occur:

› The .NET component generates HTML and JavaScript code that is necessary for displaying and running the designer;

› When the component is output, the JavaScript method is launched. It requests the report template on the server side displays it in the designer window;

› Various actions in the designer (for example, report preview, saving the report template, export reports, sorting, drill-down, etc.) call a certain action on the server-side. You can perform the necessary manipulations with the report.

4.2.2 Activation

YouTube

Watch videos that show how to activate the ASP.NET MVC HTML5 Designer.

© 2003-2024 Stimulsoft

Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiMvcDesigner** component.

HomeController.cs

```
...
public class HomeController : Controller
{
   static HomeController()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
      var path = System.Web.HttpContext.Current.Server.MapPath("~/Content/
      license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

4.2.3 Editing Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To edit a report template, you need to add the **StiMvcDesigner** component to the page, specify the minimum necessary settings, and define the actions required in

© 2003-2024 Stimulsoft

the view controller.

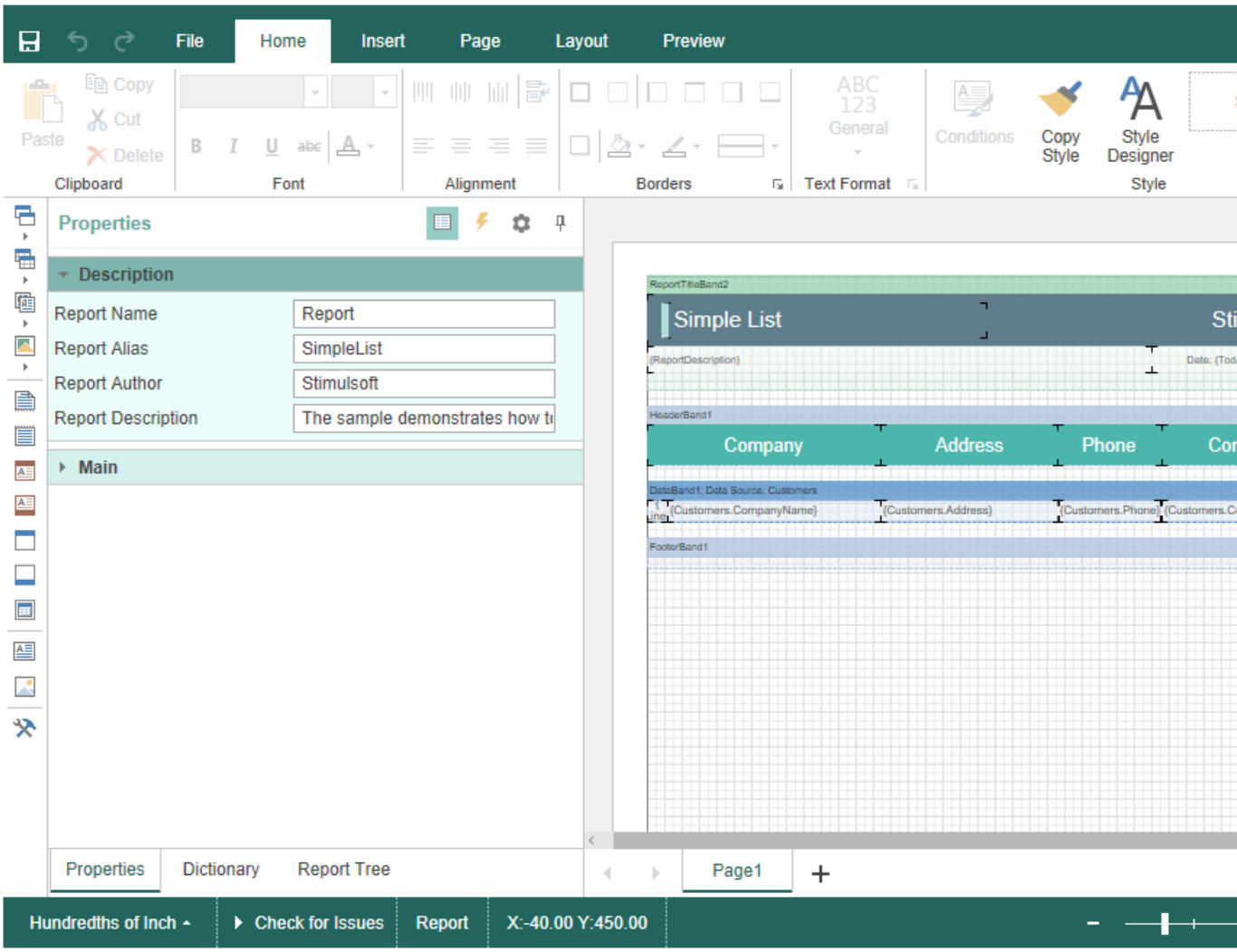
Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         GetReport = "GetReport",
         DesignerEvent = "DesignerEvent"
      }
})
...
```

HomeController.cs

```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("~/Content/SimpleList.mrt"));
   //report.Load(Server.MapPath("~/Content/Dashboard.mrt"));

   return StiMvcDesigner.GetReportResult(report);
}

public ActionResult DesignerEvent()
{
   return StiMvcDesigner.DesignerEventResult();
}
...
```

© 2003-2024 Stimulsoft



The **GetReport** action is used to load an editable report template. It is called automatically after the report designer is loaded. The **DesignerEvent** action is designed to process various additional designer actions, such as working with data and components, previewing reports, and others.

Information

The **DesignerEvent** action is mandatory. Without it, the correct work of the designer is impossible.

© 2003-2024 Stimulsoft

4.2.4 Creating New Reports and New Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the report designer with a new (empty) report, it is enough to create a new report in the **GetReport** action and return it to the designer. If necessary, you can load data for the report or perform any other required actions.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         GetReport = "GetReport"
      }
})
...
```

HomeController.cs

```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   //var newDashboard = StiReport.CreateNewDashboard();

   return StiMvcDesigner.GetReportResult(report);
   //return StiMvcDesigner.GetReportResult(newDashboard);
}
...
```

You can also create a new report using the main menu of the designer. The **CreateReport** action is used to load data for a new report or perform any other necessary actions. This action will be called when creating a new empty report or when creating a report using the wizard.

Index.cshtml

© 2003-2024 Stimulsoft

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         CreateReport = "CreateReport"
      }
})
...
```

HomeController.cs

```
...
public ActionResult CreateReport()
{
   StiReport report = new StiReport();
   //var newDashboard = StiReport.CreateNewDashboard();

   // Register data for the new report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));
   report.RegData(data);
   //newDashboard.RegData(data);
   report.Dictionary.Synchronize();
   //newDashboard.Dictionary.Synchronize();

   return StiMvcDesigner.GetReportResult(report);
   //return StiMvcDesigner.GetReportResult(newDashboard);
}
...
```

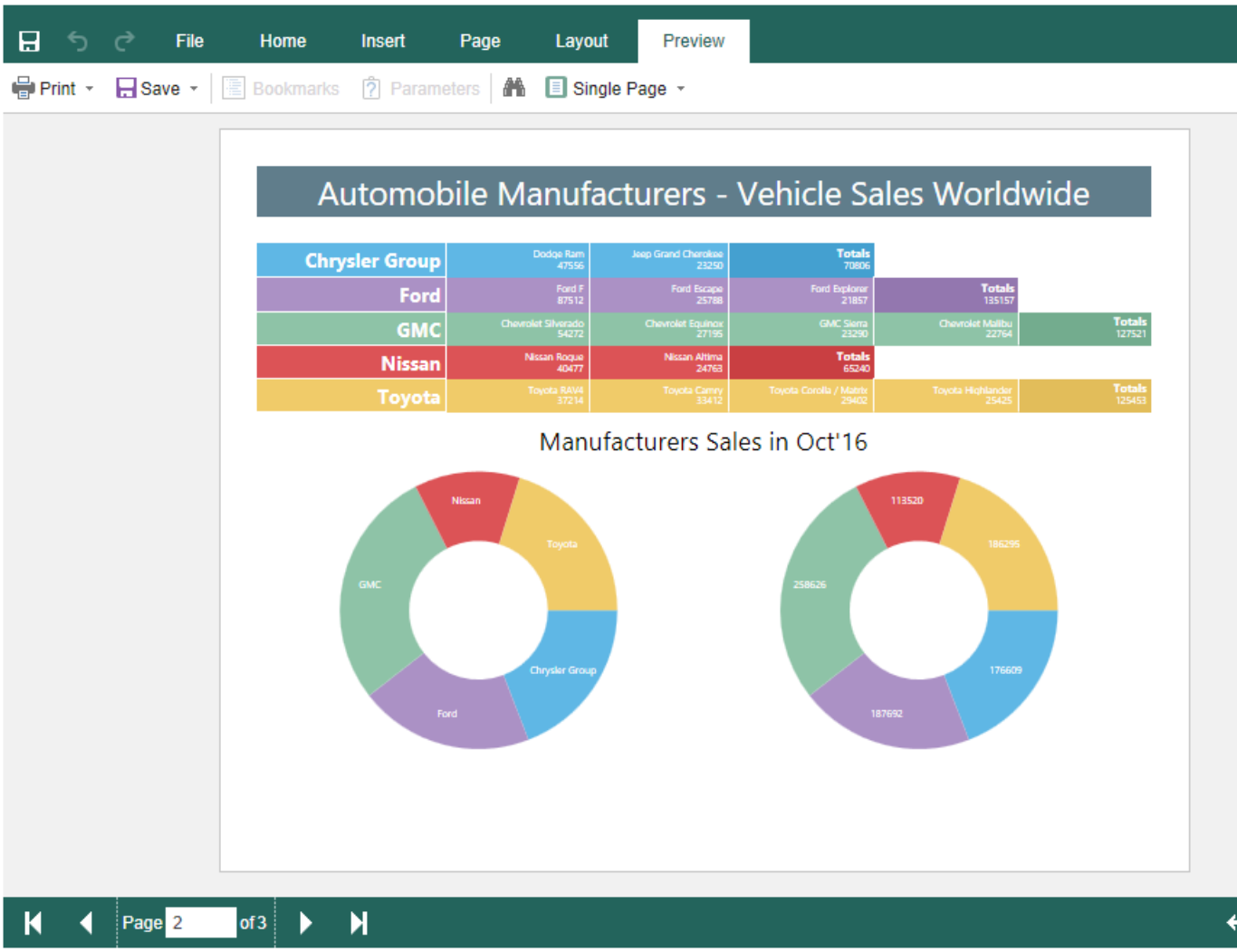
4.2.5 Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides the ability to preview reports. To preview the report, just go to the appropriate tab in the designer window. The report template will be transferred to the server-side, rendered, and displayed in the embedded viewer.

© 2003-2024 Stimulsoft



Before previewing the report, it is possible to perform any necessary actions, for example, connect data for the report. To do this, you can use the particular PreviewReport action that will be called before previewing the report. The **PreviewReport** action is called before preparing and rendering a report for viewing until it saves the cash.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         PreviewReport = "PreviewReport"
      }
})
```

© 2003-2024 Stimulsoft

```
...
```

HomeController.cs

```
...
public ActionResult PreviewReport()
{
   StiReport report = StiMvcDesigner.GetActionReportObject();
   //var Dashboard = StiMvcDesigner.GetActionReportObject();

   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));
   report.RegData(data);
   //Dashboard.RegData(data);

   return StiMvcDesigner.PreviewReportResult(report);
   //return StiMvcDesigner.PreviewReportResult(Dashboard);
}
...
```

If you need to make actions on your report immediately before displaying the report, you can use the **GetPreviewReport** action, which is called after the request of the prepared report from the cash.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         GetPreviewReport = "GetPreviewReport"
      }
})
...
```

HomeController.cs

```
...
public ActionResult GetPreviewReport()
{
   StiReport report = StiMvcDesigner.GetActionReportObject();
   //var Dashboard = StiMvcDesigner.GetActionReportObject();

   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));
   report.RegData(data);
   //Dashboard.RegData(data);
   //report.IsRendered = false;

   return StiMvcDesigner.PreviewReportResult(report);
```

© 2003-2024 Stimulsoft

```
   //return StiMvcDesigner.PreviewReportResult(Dashboard);
}
...
```

Information

So as in this event, a prepared report for viewing is transferred. If you need to render again, you should set the **report.IsRendered = false** flag.

4.2.6 Additional Features of Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The preview window of the **HTML5 Designer** component has a fully functional interactive HTML5 Viewer that can print and export reports, supports working with report parameters, dynamic sorting, interactive reports, collapsing, etc. To use these features, you do not need any additional settings for the report designer.

In any of the above actions, you can work with the report template, such as changing its properties and parameters and connecting new data for rendering.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         ExportReport = "ExportReport"
      }
})
...
```

HomeController.cs

```
...
public ActionResult ExportReport()
```

© 2003-2024 Stimulsoft

```
{
   StiReport report = StiMvcDesigner.GetActionReportObject();
   // ...

   return StiMvcDesigner.ExportReportResult(report);
}
...
```

Information

If you do not need any of these additional options to preview the report (for example, exporting or printing a report), you can disable them using the appropriate properties of the **HTML5 Designer** component.

4.2.7 Saving Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides two ways of saving the report which are available in the main menu and in the main panel of the designer - **Save Report** and **Save As**. In turn, each of these ways has its own modes and settings.

Saving a report and dashboard on the server side

To save the editable report on the server-side, you need to set the **SaveReport** action, which will be called when you select Save in the main menu or click the Save button on the main panel of the designer.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         SaveReport = "SaveReport"
      }
```

© 2003-2024 Stimulsoft

```
})
...
```

HomeController.cs

```
...
public ActionResult SaveReport()
{
   StiReport report = StiMvcDesigner.GetReportObject();

   // Save the report template
   // ...

   return StiMvcDesigner.SaveReportResult();
}
...
```

This action returns a response to the client-side of the designer about the result of saving the report. After saving the report, it is possible to display a dialog box with an error or a text message.

HomeController.cs

```
...
public ActionResult SaveReport()
{
   StiReport report = StiMvcDesigner.GetReportObject();

   // Save the report template
   // ...

   // Completion of the report saving with message dialog box
   return StiMvcDesigner.SaveReportResult("Some message after saving");
}
...
```

You can get a report name from the designer save dialog or an original report name.

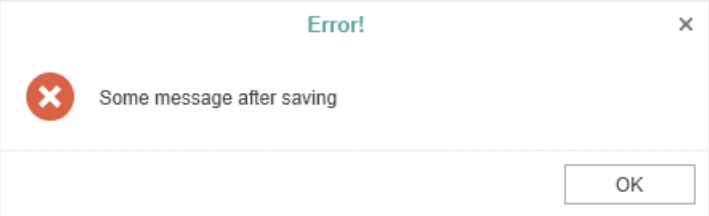
HomeController.cs

```
...
public ActionResult SaveReport()
{
   var requestParams = StiMvcDesigner.GetRequestParams();
   var report = StiMvcDesigner.GetReportObject();

   //Report name from designer save dialog
   var savingReportName = requestParams.Designer.FileName;
```

© 2003-2024 Stimulsoft

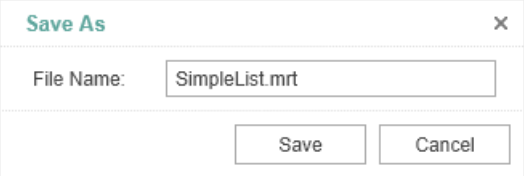
```
    //Original report name from properties
    var originalReportName = report.ReportName;

    return StiMvcDesigner.SaveReportResult();
}
...
```

In this case, the dialog with the specified text will be displayed. The text can contain both an error message of saving or a warning or any other message.



Saving reports and dashboards on the client side

To save the edited report on the client-side as a file, no additional designer settings are required. It is enough to click the **Save As** main menu item. The dialog box will be displayed. In this dialog, you can change the name of the report file. The file will be saved to the local disk of the computer.



The **HTML5 Designer** component provides the ability to change the behavior of the specified save option. For this purpose, the special **SaveReportAs** action is used in the designer. If you use this event, the report will be saved on the server-side. The work of this event will be similar to the **SaveReport** action.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
    new StiMvcDesignerOptions() {
        Actions =
        {
```

© 2003-2024 Stimulsoft

```
        SaveReportAs = "SaveReportAs"
      }
})
...
```

HomeController.cs

```
...
public ActionResult SaveReportAs()
{
   StiReport report = StiMvcDesigner.GetReportObject();

   // Save the report template
   // ...

   return StiMvcDesigner.SaveReportResult();
}
...
```

Saving settings

The report is saved in the background mode without reloading the page in the web browser window. Suppose you need to control the process of saving the report visually. In that case, you should change the value of the **SaveReportMode** (or **SaveReportAsMode**) property of the designer to one of the three specified values - **Hidden** (default value), **Visible**, or **NewWindow**.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         SaveReportAs = "SaveReportAs"
      },
      Behavior =
      {
         SaveReportAsMode = StiSaveMode.Visible
      }
})
...
```

If the **SaveReportMode** property is set to **Visible**, the report save action will be called in the current browser window in the normal (visible) mode using the POST request. If the **SaveReportMode** property is set to **NewWindow**, the report save event will be called in a new window of the web browser. By default, this property is set to **Hidden** - the report save event is called in the background using the AJAX

© 2003-2024 Stimulsoft

request and is not displayed in the browser window. The same values and behavior are applicable to the **SaveReportAsMode** property.

4.2.8 Localization

The **HTML5 Designer** component supports the complete localization of its interface. Use the special **Localization** property to localize the report designer interface. As a value of this property, you should specify the path to the localization XML file (relative or absolute).

Index.cshtml

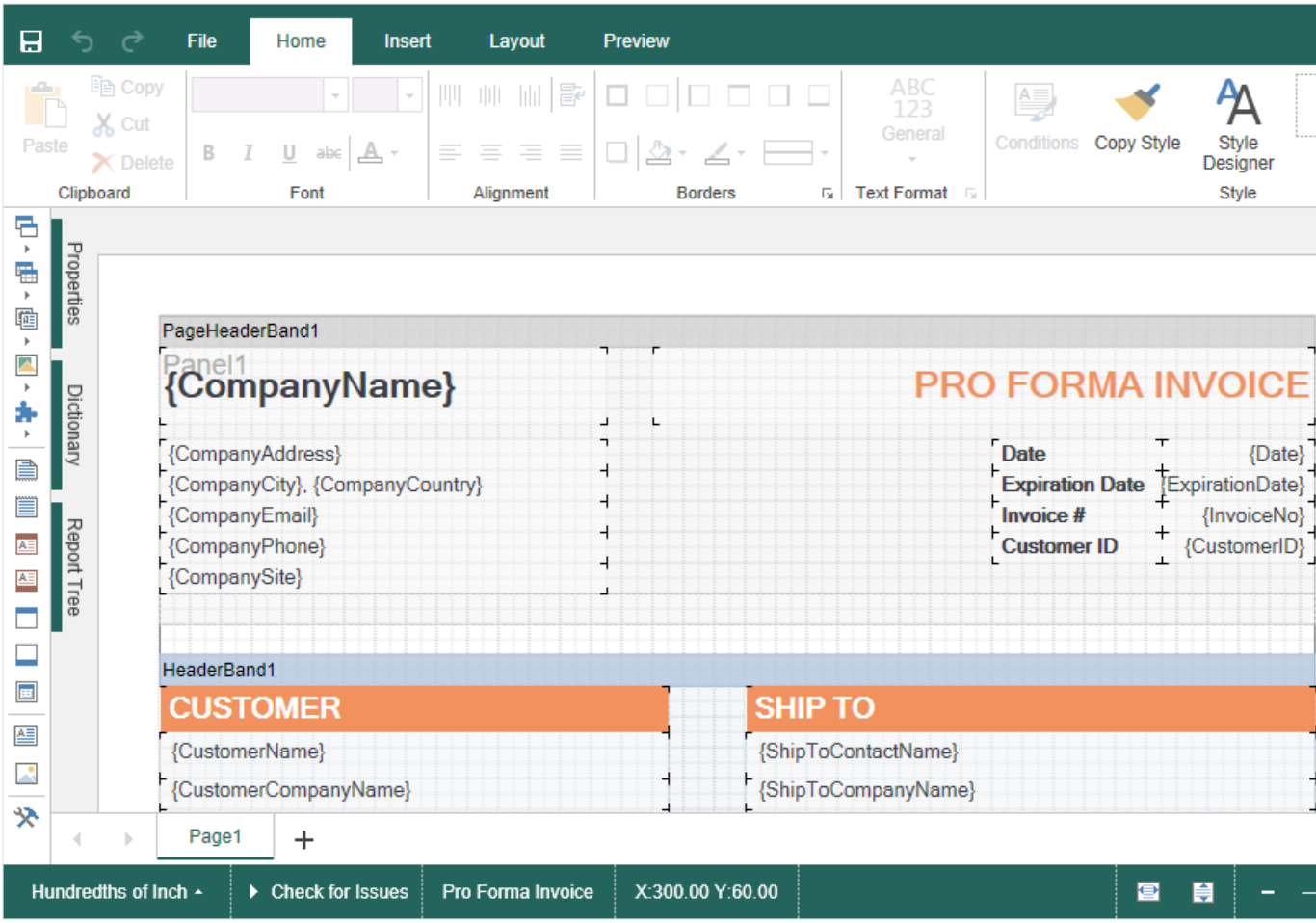
```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Localization = "~/Content/Localization/en.xml"
})
...
```

The interface of the report designer allows you to select the necessary localization from an access list. To do this, set value for the **LocalizationDirectory** property as the folder in which the localization XML files are stored.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Localization = "~/Content/Localization/en.xml",
      LocalizationDirectory = "~/Content/Localization"
})
...
```

© 2003-2024 Stimulsoft



Information

If the value for the **Localization** property is set, then when you run the report designer, the localization specified in this property will always be applied. If the property value is not set, the localization selected from the list of available localizations in the report designer panel will be automatically loaded.

4.2.9 Using Themes

In the **HTML5 Designer** component, you can change the appearance of visual controls. To change the theme, you should use the **Theme** property.

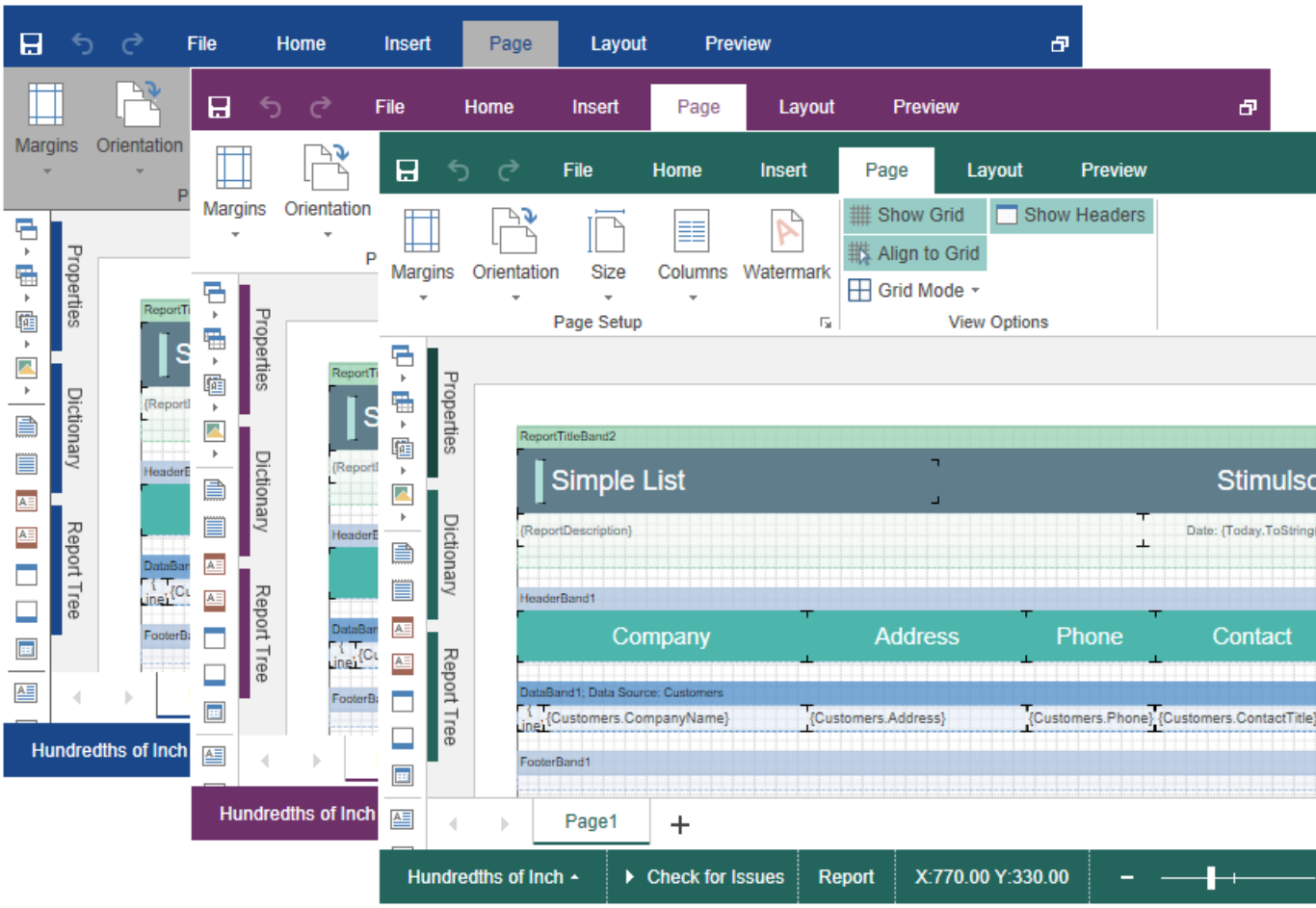
Default.aspx

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
```

© 2003-2024 Stimulsoft

```
    Theme = StiDesignerTheme.Office2022WhiteTeal
})
...
```

There are currently **2 themes** available with different color accents. As a result, **more than 50** variants of the appearance are available. This allows you to customize the appearance of the designer for almost any design of the Web project.



4.2.10 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

HTM5 Designer uses the server cache to store the editable report template. It is necessary because the client part of the designer contains only a visual representation of components of a report template. The report object itself with all the parameters and properties is stored on the server-side.

You can manage caching with the following properties.

The CacheMode property

This property of the designer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** - for caching, the current session is used, in which the report is saved as a packed string.

The CacheItemPriority property

This property sets the priority of the report stored in the cache of the server. It affects the automatic clearing of the server memory in case of memory shortage. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes you want to store the report in the server cache. If, when using caching, the requested report is not found in the cache (time of storing this report expired), then it will be requested again using the special **GetReport** action. In this case, the unsaved changes may be lost.

© 2003-2024 Stimulsoft

The **HTML5 Designer** component provides the ability to specify its methods for working with report caching. For this purpose, a special **StiCacheHelper** class is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetObject** and **SaveObject**.

HomeController.cs

```
...
public class DesignerController : Controller
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override object GetObject(string guid)
      {
         string path =
         System.IO.Path.Combine(System.Web.HttpContext.Current.Server.MapPat
         h("~/"), "CacheFiles", guid);
         if (System.IO.File.Exists(path))
         {
            byte[] cacheData = System.IO.File.ReadAllBytes(path);
            return StiCacheHelper.GetObjectFromCacheData(cacheData);
         }
         return null;

         //return base.GetObject(guid);
      }

      public override void SaveObject(object obj, string guid)
      {
         byte[] cacheData = StiCacheHelper.GetCacheDataFromObject(obj);
         string path =
         System.IO.Path.Combine(System.Web.HttpContext.Current.Server.MapPat
         h("~/"), "CacheFiles", guid);
         System.IO.File.WriteAllBytes(path, cacheData);

         //base.SaveObject(obj, guid);
      }

      public override void RemoveReport(string guid)
      {
         string path =
         System.IO.Path.Combine(System.Web.HttpContext.Current.Server.MapPat
         h("~/"), "CacheFiles", guid);
         if (File.Exists(path))
         {
            File.Delete(path);
         }
      }
   }

   static DesignerController()
```

© 2003-2024 Stimulsoft

```
    {
        StiMvcDesigner.CacheHelper = new StiMyCacheHelper();
    }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as the value of the **StiMvcDesigner.CacheHelper** static property in the constructor of the controller.

4.2.11 Additional Methods

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

For **HTML5 Designer**, several additional methods are used to get the object of the currently edited report, parameters of the current state of the designer, and other useful data. These methods can be used in any actions of the designer.

The GetReportObject() Method

Returns the report object with which the designer is currently working. It is possible to perform the necessary actions - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the designer, specifying it as a parameter in the resulting action method.

HomeController.cs

```
...
public ActionResult ExportReport()
{
    StiReport report = StiMvcDesigner.GetReportObject();
    report.ReportName = "MyReportName";

    return StiMvcDesigner.ExportReportResult(report);
}
...
```

© 2003-2024 Stimulsoft

The GetActionReportObject() method

Returns the report object that will be used for the particular action. For example, for the **OpenReport** action, this method returns a report loaded from the local disk of the computer. For the **PreviewReport** action, the method returns a prepared copy of the report for preview.

HomeController.cs

```
...
public ActionResult OpenReport()
{
   StiReport report = StiMvcDesigner.GetActionReportObject();

   // Register data for the opened report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(Server.MapPath("~/Content/Data/Demo.xml"));
   report.RegData(data);
   report.Dictionary.Synchronize();

   return StiMvcDesigner.GetReportResult(report);
}
...
```

The GetRouteValues() method

Returns values for URLs with which the designer page was opened. Thus, it is possible to get the initial collection of page parameters to run the designer and use these values for any checks and conditions.

HomeController.cs

```
...
public ActionResult ExportReport()
{
   RouteValueDictionary routeValues = StiMvcDesigner.GetRouteValues();

   return StiMvcDesigner.ExportReportResult();
}
...
```

You can also get values of URL parameters by parameter name, specifying it as the parameter of the called action of the designer.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public ActionResult ExportReport(string id)
{
   return StiMvcDesigner.ExportReportResult();
}
...
```

The GetRequestParams() method

Returns all parameters of the current state of the designer passed to the server-side. They can be useful for determining the type of action that the designer is currently executing - for example, to determine the type of export, and all action parameters.

HomeController.cs

```
...
public ActionResult ExportReport()
{
   StiRequestParams requestParams = StiMvcDesigner.GetRequestParams();
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiMvcDesigner.GetReportObject();

      // Some action with report for the PDF export
      // ...

      return StiMvcDesigner.ExportReportResult(report);
   }

   return StiMvcDesigner.ExportReportResult();
}
...
```

The GetExportSettings() method

Returns all parameters of the current report export. The parameter object type will correspond to the type of export selected in the report preview menu. Any export parameters can be changed and passed to the input of the resulting method. In this case, the report will be exported with the parameters transferred.

HomeController.cs

```
...
public ActionResult ExportReport()
{
```

© 2003-2024 Stimulsoft

```
    StiExportSettings settings = StiMvcDesigner.GetExportSettings();
    if (settings.GetExportFormat() == StiExportFormat.Pdf)
    {
        StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
        pdfSettings.EmbeddedFonts = true;
        pdfSettings.AllowEditable = StiPdfAllowEditable.No;
        return StiMvcDesigner.ExportReportResult(settings);
    }

    return StiMvcDesigner.ExportReportResult();
}
...
```

4.2.12 Timeout

When working with the **StiMvcDesigner** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
    new StiMvcDesignerOptions() {
        Server =
        {
            CacheTimeout = 10
        }
})
...
```

Using cache will increase the speed of the report designer. See the chapter Caching for more information

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

© 2003-2024 Stimulsoft

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Server =
      {
         RequestTimeout = 30
      }
})
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection and data sources in the report.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Actions =
      {
         GetReport = "GetReport",
         DesignerEvent = "DesignerEvent"
      }
})
...
```

HomeController.cs

```
...
public ActionResult GetReport()
{

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiMvcDesigner.GetReportResult(report);
}

public ActionResult DesignerEvent()
{
   return StiMvcDesigner.DesignerEventResult();
```

© 2003-2024 Stimulsoft

```
}
...
```

4.2.13 Add custom functions

Information

See on GitHub example of adding a custom function in the ASP.NET MVC HTML5 Designer component.

You can add a custom function to the Dictionary in the report designer when integrating it into your application. After adding the custom function, you can use this in creating reports and dashboards. Below is the example of adding a function for converting text to uppercase.

DesignerController.cs

```
...
public static string MyFunc(string value)
{
   return value.ToUpper();
}
...
static DesignerController()
{
   var ParamNames = new string[1];
   var ParamTypes = new Type[1];
   var ParamDescriptions = new string[1];

   ParamNames[0] = "value";
   ParamDescriptions[0] = "Descriptions";
   ParamTypes[0] = typeof(string);

   // How to add my function
   StiFunctions.AddFunction(
   "MyCategory",
   "MyFunc",
   "MyFunc",
   "Description",
   typeof(DesignerController),
   typeof(string),
   "Return Description",
   ParamTypes,
   ParamNames,
   ParamDescriptions);
}
...
```

© 2003-2024 Stimulsoft

4.2.14 Settings

The **HTML5 Designer** configuration is done using properties that are located in the **StiMvcDesignerOptions** class. All properties are divided into groups. Some of the groups contain subgroups of properties for ease of use. Below is an example of setting some properties of the designer.

Index.cshtml

```
...
@Html.Stimulsoft().StiMvcDesigner("MvcDesigner1",
   new StiMvcDesignerOptions() {
      Theme = Stimulsoft.Report.Web.StiDesignerTheme.Office2022WhiteTeal,
      Localization = "~/Content/Localization/en.xml",
      Actions =
      {
         GetReport = "GetReport",
         PreviewReport = "PreviewReport",
         SaveReport = "SaveReport",
         DesignerEvent = "DesignerEvent"
      },
      Appearance =
      {
         InterfaceType = StiInterfaceType.Auto,
         ShowTooltipsHelp = false,
         ShowDialogsHelp = false,
         DefaultUnit = Stimulsoft.Report.StiReportUnitType.Centimeters
      },
      Dictionary =
      {
         PermissionBusinessObjects =
         Stimulsoft.Report.Web.StiDesignerPermissions.None,
         PermissionDataConnections =
         Stimulsoft.Report.Web.StiDesignerPermissions.View
      },
      Bands =
      {
         ShowChildBand = false,
         ShowEmptyBand = false,
         ShowOverlayBand = false
      }
})
...
```

Basic settings (without groups)

Name	Description
Theme	Specifies the theme of the report designer. The

© 2003-2024 Stimulsoft

	list of available themes is located in the **StiDesignerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Specifies the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used. It is built into the designer and does not require additional XML files.
LocalizationDirectory	Specifies the path to the directory with XML localization files. The localization files located in the specified folder will be loaded to the localization list in the designer panel.
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the component is expanded to the entire area of the browser window.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the component is expanded to the entire area of the browser window.

Actions

Name	Description
GetReport	Specifies the name of the action method to prepare the report template when loading the designer.
OpenReport	Specifies the name of the action method to open the report template from the designer menu.

© 2003-2024 Stimulsoft

CreateReport	Specifies the name of the action method to prepare the report template when creating the new report in the designer.
SaveReport	Specifies the name of the action method to save the report template on the server-side.
SaveReportAs	Specifies the name of the action method to store the report template on the server-side when using the **Save As** menu item. If no action is specified, the built-in method of saving the report template to the local disk will be used.
PreviewReport	Specifies the name of the action method to prepare the rendered report in the preview window.
ExportReport	Specifies the name of the action method to export reports to the specified format.
Exit	Specifies the name of the action method to go to the desired view by clicking the Exit button in the main menu of the report designer.
DesignerEvent	Specifies the name of the action method of the report designer to handle additional designer actions, such as working with data, report components, and others. Also, this action is used to load scripts and designer styles.

Server

Name	Description
Controller	Sets the name of the controller to process requests. If this property is not set, then the current controller will be used to process the requests.
RouteTemplate	Sets the route template that is returned when the report designer actions are executed. If the property is not set, then the MVC project template will be used instead. The default value

© 2003-2024 Stimulsoft

	of the property is null.
ShowServerErrorPage	Enables the display of an HTML page with error details that occurred on the server side. The error details will be displayed in the preview window if the property is enabled. If the property is disabled, the numeric error code and a short error description will be displayed in the dialog window. By default, the property is set to **true**.
AllowAutoUpdateCookies	Allows the designer to update the cookies automatically on every request to the server. By default, cookies are set when creating the designer, if they are not specified in the report. By default, the property is set to **false**.
AllowAntiforgeryToken	Allow the designer to automatically request and send the antiforgery token. By default, the property is set to **true**.
RequestTimeout	Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled in **HTML5 Designer**; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the

© 2003-2024 Stimulsoft

	storage, the report is serialized to a packed string; › **StringSession** – the session is used as a repository, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or server session will be automatically re-saved after a certain period of time if the designer is idle (about every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the designer mode in which relative URLs are used for requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value which specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the designer. If **false**, only the necessary parameters are used to request the resources of the designer. This corresponds to the more correct operation of the browser cache. By default, the property is set to **true**.
PassFormValues	Enables transferring the POST form values to the client side, if these values are to be used in the actions of the designer. If you enable this feature, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
UseCompression	Enables compression of designer requests in the

© 2003-2024 Stimulsoft

	GZip stream. This allows you to reduce the amount of Internet traffic, but slows down the designer. By default, the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server side. The following resources are supported: scripts, styles and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Specifies the path to the CSS file of styles for the report designer. If this property is set, the standard styles of the selected theme will not be loaded. The default value is an empty value.
DefaultUnit	Sets the units for the size of the report and all its components. By default, centimeters are used.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the **StiZoomMode** enumeration: › **PageWidth** – when the designer runs, the zoom, necessary to display the report by the page width, will be set; › **PageHeight** – when the designer runs, the zoom, required to display the page height of the report, will be set.

© 2003-2024 Stimulsoft

ShowAnimation	Enables animation for various elements of the designer interface. By default, the property is set to **true**.
ShowOpenDialog	Allows to display the open dialog, or to open with the open event. By default, the property is set to **true**.
ShowTooltips	Enables displaying tooltips for designer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enable displaying links to online documentation in tooltips for designer controls. By default, the property is set to **true**.
ShowDialogsHelp	Enables displaying links to online documentation on the titles of dialog forms of the designer. By default, the property is set to **true**.
InterfaceType	Sets the type of interface used for the designer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the interface type of the designer will be selected automatically depending on the device used (default value); › **Mouse** – forced use of the interface to control the designer with the mouse; › **Touch** – forced use of the Touch interface to control the designer via the touch screen (mobile devices), also in this mode, the interface elements are increased.
DatePickerFirstDayOfWeek	Sets the first day of the week for the select date item. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Auto** – automatic detection of the first day of the week from the browser settings (default value); › **Monday** – the first day of the week is

© 2003-2024 Stimulsoft

	Monday; › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
ShowReportTree	Enables displaying the tree of report components. By default, the property is set to **true**.
ChartRenderType	Gets or sets the type of the chart in the preview. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages in the preview tab. Can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.

© 2003-2024 Stimulsoft

CloseDesignerWithoutAsking	Sets a value which indicates that the designer will be closed without asking. By default, the property is set to **true**.
ShowSystemFonts	Sets a visibility of the system fonts in the fonts list. By default, the property is set to **true**.
WizardTypeRunningAfterLoad	Calls the Report wizard after starting the report designer. It may have one of the following **StiWizardType** enumeration values: › **None** - runs the report designer without running the report wizard; › **StandardReport** - runs the Standard wizard; › **MasterDetailReport** - runs the Master-Detail wizard; › **LabelReport** - runs the Label wizard; › **InvoicesReport** - runs the Invoice wizard; › **OrdersReport** - runs the Order wizard; › **QuotationReport** - runs the Quote wizard.
AllowWordWrapTextEditors	Allows word wrap in the text editors. By default, the property is set to **true**.

Behavior

Name	Description
ShowSaveDialog	Enables displaying the dialog to insert a report name when it is saved. The name of the report will be transferred in the parameters of the report designer. By default, the property is set to **true**.
UndoMaxLevel	Sets the maximum number to cancel actions with the report (the Undo/Redo function). A big value of this property will consume memory on the server-side to store the undo parameters. The default value is **6**.
AllowChangeWindowTitle	Allows using the title of the browser window to display the file name of the edited report. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
SaveReportMode	Sets the mode to save the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
SaveReportAsMode	Sets the mode for saving the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - the saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - the saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - the saving of the report is called in a new window (tab) of the web browser.
CheckReportBeforePreview	Sets the value that allows running the report checker before preview.

FileMenu

Name	Description
Visible	Enables displaying the main menu of the report designer. By default, the property is set to **true**.
ShowNew	Enables showing the main menu item - **New**. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowFileMenuNewReport	Sets a visibility of the new report button in the designer. By default, the property is set to **true**.
ShowFileMenuNewDashboard	Sets a visibility of the new dashboard button in the designer. By default, the property is set to **true**.
ShowOpen	Enables showing the main menu item - **Open**. By default, the property is set to **true**.
ShowSave	Enables showing the main menu item - **Save**. By default, the property is set to **true**.
ShowSaveAs	Enables showing the main menu item - **Save As**. By default, the property is set to **true**.
ShowClose	Enables showing the main menu item - **Close**. By default, the property is set to **true**.
ShowExit	Enables showing the main menu item - **Exit**. By default, the property is set to **true**.
ShowReportSetup	Enables showing the main menu item - **Report Setup**. By default, the property is set to **true**.
ShowOptions	Enables showing the main menu item - **Options**. By default, the property is set to **true**.
ShowInfo	Enables showing the main menu item - **Info**. By default, the property is set to **true**.
ShowAbout	Enables showing the main menu item - **About**. By default, the property is set to **true**.
ShowHelp	Enables showing the main menu item - **Help**. By default, the property is set to **true**.

Dictionary

Name	Description
Visible	Enables showing the data dictionary of the report. By default, the property is set to **true**.
UseAliases	Allows you to use aliases in the data dictionary. It has the three values of the **StiUseAliases**

© 2003-2024 Stimulsoft

	enumeration: › **Auto** - defines the mode of using aliases from a saved value in cookies (default value); › **True** - sets the mode of using aliases in the data dictionary; › **False** - disables the mode of using aliases in the data dictionary.
NewReportDictionary	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It has the three values of the **StiNewReportDictionary** enumeration: › **Auto** - defines the mode to create or join the data dictionary from a saved value in cookies (default value); › **DictionaryNew** - sets the mode to create a new data dictionary when creating a new report; › **DictionaryMerge** - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
ShowDictionaryContextMenuProperties	Sets a visibility of the **Properties** item in the dictionary context menu. By default, the property is set to **true**.
ShowDictionaryActions	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.
PermissionDataConnections	Sets the available actions to connect data to the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataSources	Sets available actions on report data sources. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataColumns	Sets the available actions on data columns in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionBusinessObjects	Sets the available actions on the business

© 2003-2024 Stimulsoft

	objects in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataRelations	Sets the available actions to linking data in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionVariables	Sets available actions on report variables. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionResources	Sets the available actions for the resources in the Report Dictionary. Takes one or several values from the **StiDesignerPermissions** enumeration.
PermissionSqlParameters	Sets the available actions for the parameters of the SQL queries for the Report DataSources. Takes one or several values from **StiDesignerPermissions** enumeration.
DataTransformationsPermissions	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

The table below shows all available values for the **StiDesignerPermissions** enumeration, which can be set for the dictionary elements of the report.

Value	Description
None	Disables any action on the item of the data dictionary.
All	Allows any action on the item of the data dictionary.
Create	Allows creating a specific data dictionary item.
Delete	Allows deleting a specific data dictionary item.
Modify	Allows modifying a specific data dictionary item.
View	Allows viewing a specific data dictionary item.

© 2003-2024 Stimulsoft

ModifyView	Allows modifying and viewing a specific data dictionary item.

Toolbar

Name	Description
ShowToolbar	Enables displaying the toolbar of the designer. By default, the property is set to **true**.
ShowSetupToolboxButton	Enables displaying the button to call the dialog box with settings for the side toolbar. By default, the property is set to **true**.
ShowInsertButton	Enables displaying the **Insert** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowLayoutButton	Enables displaying the tab **Layout** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPageButton	Enables displaying the tab **Page** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPreviewButton	Enables displaying the tab **Preview** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the designer. By default, the property is set to **true**.
ShowAboutButton	Enables displaying the **About** on the toolbar of the designer. By default, the property is set to **false**.

PropertiesGrid

Name	Description

© 2003-2024 Stimulsoft

Visible	Enables displaying the property panel. By default, the property is set to **true**.
Width	Sets the width of the property panel. By default, the width is set to **370 px**.
LabelWidth	Specifies the width of the labels on the properties panel. By default, the width is set to **160 px**.
PropertiesGridPosition	Sets **Left** or **Right** position of the properties grid in the designer. It has the three values of the **StiPropertiesGridPosition** enumeration: › **Left**; › **Right**.
ShowPropertiesWhichUsedFromStyles	Sets a visibility of the properties which used from styles in the designer. By default, the property is set to **false**.

Components

Name	Description
ShowText	Enables displaying the **Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowTextInCells	Enables displaying the **Text in Cells** component in the insert menu for report components. By default, the property is set to **true**.
ShowRichText	Enables displaying the **Rich Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowImage	Enables displaying the **Image** component in the insert menu for report components. By default, the property is set to **true**.
ShowBarCode	Enables displaying the **Bar Code** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowShape	Enables displaying the **Shape** component in the insert menu for report components. By default, the property is set to **true**.
ShowHorizontalLinePrimitive	Enables displaying the **Horizontal Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowVerticalLinePrimitive	Enables displaying the **Vertical Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowRectanglePrimitive	Enables displaying the **Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowRoundedRectanglePrimitive	Enables displaying the **Rounded Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowPanel	Enables displaying the **Panel** component in the insert menu for report components. By default, the property is set to **true**.
ShowClone	Enables displaying the **Clone** component in the insert menu for report components. By default, the property is set to **true**.
ShowCheckBox	Enables displaying the **Check Box** component in the insert menu for report components. By default, the property is set to **true**.
ShowSubReport	Enables displaying the **Sub Report** component in the insert menu for report components. By default, the property is set to **true**.
ShowZipCode	Enables displaying the **Zip Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowTable	Enables displaying the **Table** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowCrossTab	Enables displaying the **Cross-Tab** component in the insert menu for report components. By default, the property is set to **true**.
ShowChart	Enables displaying the **Chart** component in the insert menu for report components. It affects on all chart types. By default, the property is set to **true**.
ShowMap	Enables displaying the **Map** component in the insert menu for report components. By default, the property is set to **false**.
ShowGauge	Enables displaying the **Gauge** component in the insert menu for report components. By default, the property is set to **false**.
ShowSparkline	Enables displaying the **Sparkline** component in the insert menu for report components. By default, the property is set to **true**.
ShowMathFormula	Enables displaying the **Math Formula** component in the insert menu for report components. By default, the property is set to **true**.
ShowElectronicSignature	Enables displaying the **Electronic Signature** component in the insert menu for report components. By default, the property is set to **true**.
ShowPdfDigitalSignature	Enables displaying the **PDF Digital Signature** component in the insert menu for report components. By default, the property is set to **true**.

Bands

Name	Description
ShowReportTitleBands	Enables displaying the **Report Title** item in the **Bands** menu of the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowReportSummaryBand	Enables displaying the **Report Summary** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageHeaderBand	Enables displaying the **Page Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageFooterBand	Enables displaying the **Page Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupHeaderBand	Enables displaying the **Group Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupFooterBand	Enables displaying the **Group Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHeaderBand	Enables displaying the **Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowFooterBand	Enables displaying the **Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnHeaderBand	Enables displaying the **Column Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnFooterBand	Enables displaying the **Column Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowDataBand	Enables displaying the **Data** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHierarchicalBand	Enables displaying the **Hierarchical** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowChildBand	Enables displaying the **Child** item in the **Bands** menu of the designer. By default, the property is

© 2003-2024 Stimulsoft

	set to **true**.
ShowEmptyBand	Enables displaying the **Empty** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowOverlayBand	Enables displaying the **Overlay** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowTableOfContents	Enables displaying the **Table of Contents** item in the **Bands** menu of the designer. By default, the property is set to **true**.

DashboardElements

Name	Description
ShowTableElement	Enables displaying the **Table** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowCardsElement	Enables displaying the **Cards** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowChartElement	Enables displaying the **Chart** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowGaugeElement	Enables displaying the **Gauge** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPivotTableElement	Enables displaying the **Pivot** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowIndicatorElement	Enables displaying the **Indicator** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowProgressElement	Enables displaying the **Progress** element in the Dashboard Elements menu of the designer. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
ShowRegionMapElement	Enables displaying the **Region Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowOnlineMapElement	Enables displaying the **Online Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowImageElement	Enables displaying the **Image** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTextElement	Enables displaying the **Text** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPanelElement	Enables displaying the **Panel** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowShapeElement	Enables displaying the **Shape** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowButtonElement	Enables displaying the **Button** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowListBoxElement	Enables displaying the **List Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowComboBoxElement	Enables displaying the **Combo Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewElement	Enables displaying the **Tree View** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewBoxElement	Enables displaying the **Tree View Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowDatePickerElement	Enables displaying the **Date Picker** element in

© 2003-2024 Stimulsoft

	the Dashboard Elements menu of the designer. By default, the property is set to **true**.

CrossBands

Name	Description
ShowCrossGroupHeaderBand	Enables displaying the **Cross Group Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossGroupFooterBand	Enables displaying the **Cross Group Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossHeaderBand	Enables displaying the **Cross Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossFooterBand	Enables displaying the **Cross Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossDataBand	Enables displaying the **Cross Data** section in the insert menu for report components. By default, the property is set to **true**.

Dashboards

Name	Description
ShowNewDashboardButton	Sets visibility of the **New Dashboard** button in the designer. By default, the property is set to **true**.

Pages

© 2003-2024 Stimulsoft

Name	Description
ShowNewPageButton	Sets visibility of the **New Page** button in the designer. By default, the property is set to **true**.

When designing a report or dashboard in the report designer, you can also define **ExportOptions**, **EmailOptions**, and **PreviewToolbarOptions** on the **Preview** tab. These options are similar to the report viewer options.

5 Reports and Dashboards for ASP.NET Core MVC

.NET Core is a cross-platform technology for creating Web applications for Windows, Linux, and macOS. **ASP.NET Core MVC** is a multifunctional platform for creating Web applications and APIs using the Model-View-Controller design structure. Stimulsoft company provides tools for creating, displaying, and transforming reports and dashboards using this technology.

Tools for creating and editing reports:	Tools for viewing and converting reports:	Tools for creating and editing dashboards:	Tools for viewing and converting dashboards:
HTML5 designer	HTML5 viewer	HTML5 designer	HTML5 viewer

5.1 HTML5 Viewer

YouTube

Watch videos .NET Core HTML5 Viewer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

Samples

© 2003-2024 Stimulsoft

See on GitHub examples for working with the .NET Core HTML5 Viewer component. All examples are separate projects, grouped into one solution for Visual Studio.

The **HTML5 Viewer** (**StiNetCoreViewer**) component is designed for viewing reports in the web browser. You do not need to install the .NET Framework, ActiveX components, or any special plug-ins on the client-side. All that is needed is any modern Web browser.

With the help of **HTML5 Viewer**, you can view, print, and export reports on any computer with any operating system installed. Since the viewer only uses HTML and JavaScript technologies, it can be run on devices with no Flash or Silverlight support - tablets, smartphones. Also, the viewer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Viewer** component uses the AJAX technology to perform all actions (uploading a report, paging, scaling, interactivity in reports, etc.), which allows you to get rid of reloading the entire page save Web traffic, and speed up work. The report engine built using the .NET Core technology is used to render reports. This is a cross-platform technology. It allows you to deploy the application on servers that use operating systems like Windows, macOS, and Linux.

The **HTML5 Viewer** supports many themes, animated interface, bookmarks, interactive reports, editing of report elements on the page, full-screen mode, search panel, and other necessary features for viewing reports.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Viewer** in a Web project, you need to install the NuGet package of Stimulsoft.Reports.Web.NetCore:

› Select "Manage NuGet Packages ..." in the context menu of the project;

© 2003-2024 Stimulsoft

› Specify Stimulsoft.Reports.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

To add the ability to view and export dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web.NetCore (this package is associated with the package Stimulsoft.Reports.Web.NetCore. If it is missed, it will be installed automatically):
› Select "Manage NuGet Packages ..." in the context menu of the project;
› Specify Stimulsoft.Dashboards.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

- How this works?
- Activation
- Showing Reports and Dashboards
- Connecting data
- Localization
- Printing Reports
- Exporting Reports and Dashboards
- Viewing Modes
- Work with Parameters
- Work with Bookmarks
- Viewer Settings
- Interactive Reports
- Timeout
- Editing Rendered Reports
- Sending Reports by Email
- Calling Designer from Viewer
- Caching
- Using Themes
- Basic Features
- Additional Methods
- Export and Printing from Code

5.1.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

To run the viewer, you need to place the **StiNetCoreViewer** component on the page, set the necessary settings to it, and set the necessary actions in the view controller. When the report viewer runs, the following actions occur:

- The .NET Core component generates HTML and JavaScript code that is necessary for displaying and running the viewer;
- When the component is output, the JavaScript method is launched. It requests the first page of the report on the server-side or the entire report (depending on the selected mode) and the required report parameters;
- Each action in the viewer (for example, paging, printing or exporting a report, etc.) calls a certain action on the server-side, in which you can perform the necessary manipulations with the report;
- The viewer saves the report in cache or server session to speed up the work, which makes it impossible to re-build the report.

5.1.2 Activation

YouTube

Watch videos that show how to activate the ASP.NET Core HTML5 Viewer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiNetCoreViewer** component.

HomeController.cs

```
...
//Activation with using license code
public class HomeController : Controller
{
   static HomeController()
   {
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";
   }
}
```

© 2003-2024 Stimulsoft

```
//Activation with using license file
public class HomeController : Controller
{
   public HomeController(IHostingEnvironment hostEnvironment)
   {
      var path = Path.Combine(hostEnvironment.ContentRootPath, "Content\
      \license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

5.1.3 Showing Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

To show the report, you need to add the **StiNetCoreViewer** component to the page and set it to the minimum required properties, and, in the view controller, specify the necessary actions.

Index.cshtml

© 2003-2024 Stimulsoft

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent"
   }
})
...
```

HomeController.cs

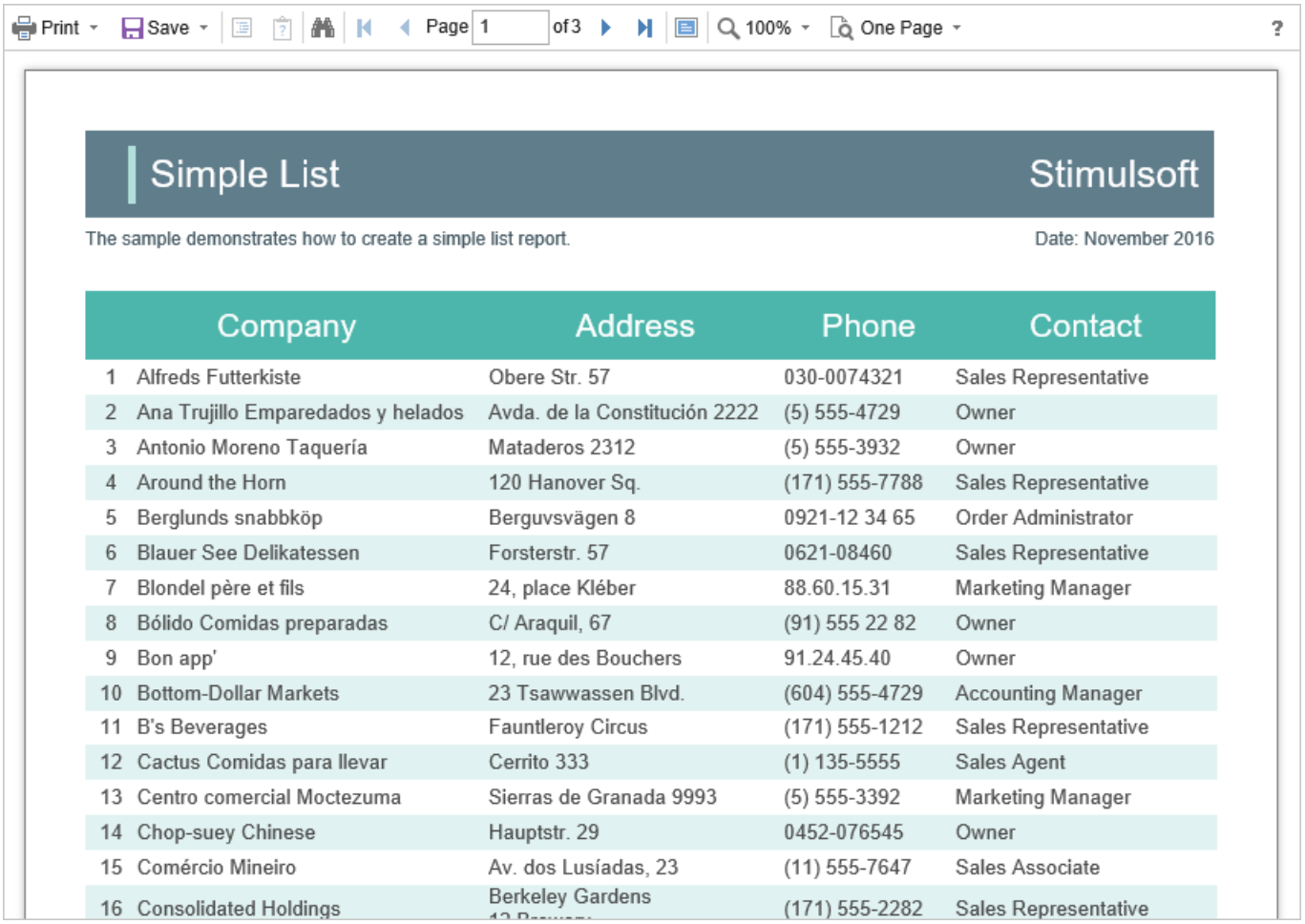
```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
   //report.Load(StiNetCoreHelper.MapPath(this, "Reports/Dashboard.mrt"));

   return StiNetCoreViewer.GetReportResult(this, report);
}

public IActionResult ViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}
...
```

In the above example, the processing of two actions of the viewer is added. The **GetReport** action returns the report prepared for preview. The **ViewerEvent** action handles the viewer events.

Information

The **ViewerEvent** action is mandatory. Without it, the correct operation of the viewer is not possible.

© 2003-2024 Stimulsoft



If the report was not rendered before showing, the **HTML5 Viewer** component will automatically render it. So you can use report templates and rendered reports to display reports.

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.LoadDocument(StiNetCoreHelper.MapPath(this, "Reports/
   SimpleList.mdc"));

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

© 2003-2024 Stimulsoft

Since the dashboard is not a static document and requires data to work, the format of the rendered MDC document is not available for it. Instead, it is possible to use a snapshot of the report in the MRT format, which contains all the data necessary for the dashboard to work and be correctly displayed in the viewer.

HomeController.cs

```
...
public ActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath("~/Reports/Snapshot.mrt"));

   return StiNetCoreViewer.GetReportResult(report);
}
...
```

Loading custom fonts

You can load custom fonts using the **StiFontCollection** class, having specified the file that contains a font. To do this, you should call the static method in the controller constructor to load a font.

ViewerController.cs

```
...
public class ViewerController : Controller
{
   static ViewerController()
   {
      Stimulsoft.Base.StiFontCollection.AddFontFile(StiNetCoreHelper.MapPat
      h(this, "/fonts/my-font/font-name.ttf"));
   }
}
...
```

5.1.4 Connecting Data

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Data to a report can be connected in various ways. The easiest way is to store

© 2003-2024 Stimulsoft

connection settings in the report template. You can also connect the data from the code. This can be done when the report is loaded in the **GetReport** action.

HomeController.cs

```
...
public IActionResult GetReport()
{
   DataSet ds = new DataSet();
   ds.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/
   TwoSimpleLists.mrt"));
   report.Dictionary.Databases.Clear();
   report.RegData("Demo", ds);

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Data for the report can be connected not only when the report is loaded. For example, you can connect new data at the moment of interactive actions in the viewer (applying report parameters, sorting, drill-down, collapsing). To do this, you should set the **Interaction** action for the **HTML5 Viewer** component and, in the action handler, connect the data for the current report. In the same way, you can connect data in other actions of the viewer.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent",
      Interaction = "ViewerInteraction"
   }
})
...
```

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   DataSet data = new DataSet();
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
```

© 2003-2024 Stimulsoft

```
   StiReport report = StiNetCoreViewer.GetReportObject(this);
   report.RegData("Demo", data);

   return StiNetCoreViewer.InteractionResult(this, report);
}
...
```

If you want to connect new data only for a certain interactive action of the viewer, for example, only when you apply report parameters you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side on any request and contain all necessary information and states of the client part of the viewer. To determine the type of action of the viewer, it is enough to check the Action property of the viewer parameters.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      DataSet data = new DataSet();
      data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

      StiReport report = StiNetCoreViewer.GetReportObject(this);
      report.RegData("Demo", data);

      return StiNetCoreViewer.InteractionResult(this, report);
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

SQL data sources

The connection parameters to the SQL data source and any other ones can be stored in the report template. Suppose you want to set the connection parameters from the code before rendering the report (for example, for security reasons or depending on the authorized user). In that case, you can use the example below.

HomeController.cs

© 2003-2024 Stimulsoft

```
...
public IActionResult GetReport()
{
   OracleConnection connection = new OracleConnection("Data
   Source=Oracle8i;Integrated Security=yes");
   connection.Open();
   OracleDataAdapter adapter = new OracleDataAdapter();
   adapter.SelectCommand = new OracleCommand("SELECT * FROM Products",
   connection);

   DataSet dataSet = new DataSet("productsDataSet");
   adapter.Fill(dataSet, "Products");

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/
   SqlSampleReport.mrt"));
   report.RegData("Products", dataSet);

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Also, for SQL data sources used in the report, you can specify the Query Timeout in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection and data sources in the report.

HomeController.cs

```
...
public IActionResult GetReport()
{

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath("Report.mrt"));
   ((StiSqlDatabase)
   report.Dictionary.Databases["Connection"]).ConnectionString = @"Data
   Source=server;Integrated Security=True;Initial Catalog=DataBase";
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).SqlCommand = "select *
   from Table where Column = 100";
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

© 2003-2024 Stimulsoft

Information

For SQL data sources of other types, the connection is created similarly, and an adapter corresponding to the data source type is connected. For example, for the MS SQL data source, you should connect SqlDataAdapter. For OracleDataAdapter is required for Oracle. Also, you should specify a connection string that matches the connection type.

You can also use data for designing reports and dashboards obtained from OData storage. In this case, you can do the authorization using a username, user password, or using a token. Authorization parameters are specified in the connection string to the OData storage using the ";" separator.

HomeController.cs

```
...
public IActionResult GetReport()
{

   var report = new StiReport();

   //Authorization using a user account
   var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
   services.odata.org/V4/Northwind/
   Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Password;C
   lient_Id=Your Client ID", false, null);

   //Authorization using a user token
   var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
   services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your token",
   false, null);

   report.Dictionary.Databases.Add(oDataDatabase);
   oDataDatabase.Synchronize(report);

   //Query with data filter
   ((StiSqlSource)report.Dictionary.DataSources["Products"]).SqlCommand =
   "Products?$filter=ProductID eq 2";

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

The table below shows the connection string templates for different types of data

© 2003-2024 Stimulsoft

sources.

Data Source	Connection String Template
MS SQL	Integrated Security=False; Data Source=myServerAddress;Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
MySQL	Server=myServerAddress; Database=myDataBase;UserId=myUsername; Pwd=myPassword;
ODBC	Driver={SQL Server}; Server=myServerAddress;Database=myDataBase; Uid=myUsername; Pwd=myPassword;
OLE DB	Provider=SQLOLEDB.1; Integrated Security=SSPI;Persist Security Info=False; Initial Catalog=myDataBase;Data Source=myServerAddress
Oracle	Data Source=TORCL;User Id=myUsername;Password=myPassword;
MS Access	Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Password=pass;Data Source=C:\ \myAccessFile.accdb;
PostgreSQL	Server=myServerAddress; Port=5432; Database=myDataBase;User Id=myUsername; Password=myPassword;
Firebird	User=SYSDBA; Password=masterkey; Database=SampleDatabase.fdb;DataSource=myServerAddress; Port=3050; Dialect=3; Charset=NONE;Role=; Connection lifetime=15; Pooling=true; MinPoolSize=0;MaxPoolSize=50; Packet Size=8192; ServerType=0;
SQL CE	Data Source=c:\MyData.sdf; Persist Security Info=False;
SQLite	Data Source=c:\mydb.db; Version=3;
DB2	Server=myAddress:myPortNumber;Database=m

© 2003-2024 Stimulsoft

	yDataBase;UID=myUsername;PWD=myPassword;Max Pool Size=100;Min Pool Size=10;
Infomix	Database=myDataBase;Host=192.168.10.10;Server=db_engine_tcp;Service=1492;Protocol=onsoctcp;UID=myUsername;Password=myPassword;
Sybase	Data Source=myASEserver;Port=5000;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
Teradata	Data Source=myServerAddress;User ID=myUsername;Password=myPassword;
VistaDB	Data Source=D:\folder\myVistaDatabaseFile.vdb4;Open Mode=ExclusiveReadWrite;
Universal(dotConnect)	Provider=Oracle;direct=true;data source=192.168.0.1;port=1521;sid=sid;user=user;password=pass
MongoDB	mongodb://<user>:<password>@localhost/test
OData	http://services.odata.org/v3/odata/OData.svc/
Other...	The table shows the most commonly used templates for the connection string. You can view various connection string options at the special website.

Data from XML, JSON, Excel files

Connecting to XML and JSON data sources can be stored in the report template. If you want to specify data files from the code, you can use the example below.

HomeController.cs

```
...
public IActionResult GetReport()
{
   DataSet data = new DataSet();
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
```

© 2003-2024 Stimulsoft

```
    report.RegData(data);

    return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

HomeController.cs

```
...
public IActionResult GetReport()
{
    DataSet data
    = StiJsonToDataSetConverterV2.GetDataSetFromFile(StiNetCoreHelper.MapPat
    h(this, "Data/Demo.json"));

    StiReport report = new StiReport();
    report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
    report.RegData(data);

    return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Information

The viewer has the possibility of obtaining data from an Excel file. To do this, you can use the following method.

```
DataSet dataSet = StiExcelConnector.Get().GetDataSet(new StiExcelOptions(ar
```

5.1.5 Localization

The **HTML5 Viewer** component supports the complete localization of its interface. To localize the report viewer interface, use the special **Localization** property. The value of this property should specify the path to the localization XML file (relative or absolute).

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
    Localization = "Localization/en.xml"
})
...
```

When you load the report viewer, the localization file will be loaded automatically.

© 2003-2024 Stimulsoft

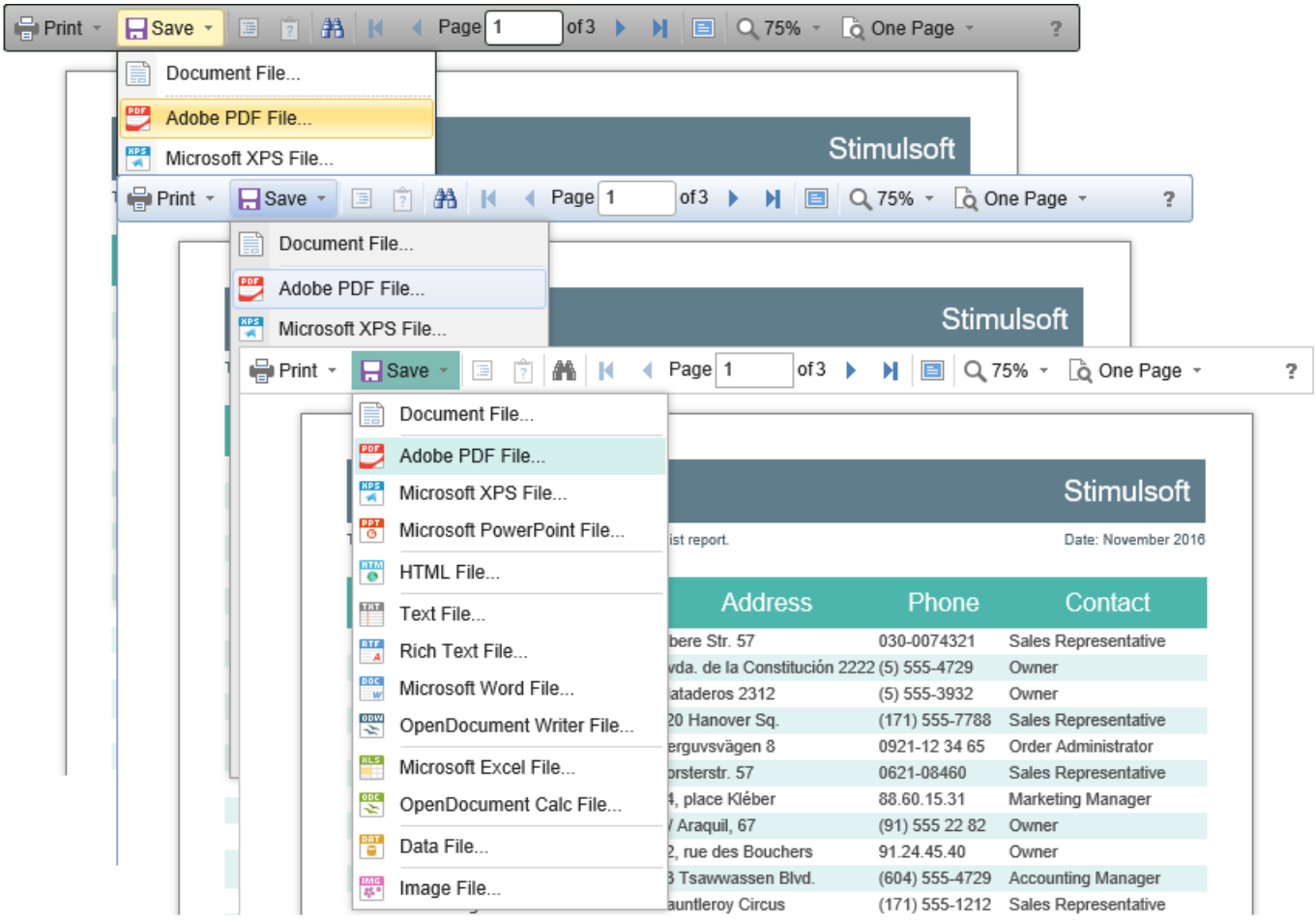
5.1.6 Using Themes

The **HTML5 Viewer** component can change the appearance of visual controls. To change the theme, use the **Theme** property, which can take one of the values of the **StiViewerTheme** enumeration.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Theme = StiViewerTheme.Office2022WhiteTeal
})
...
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.



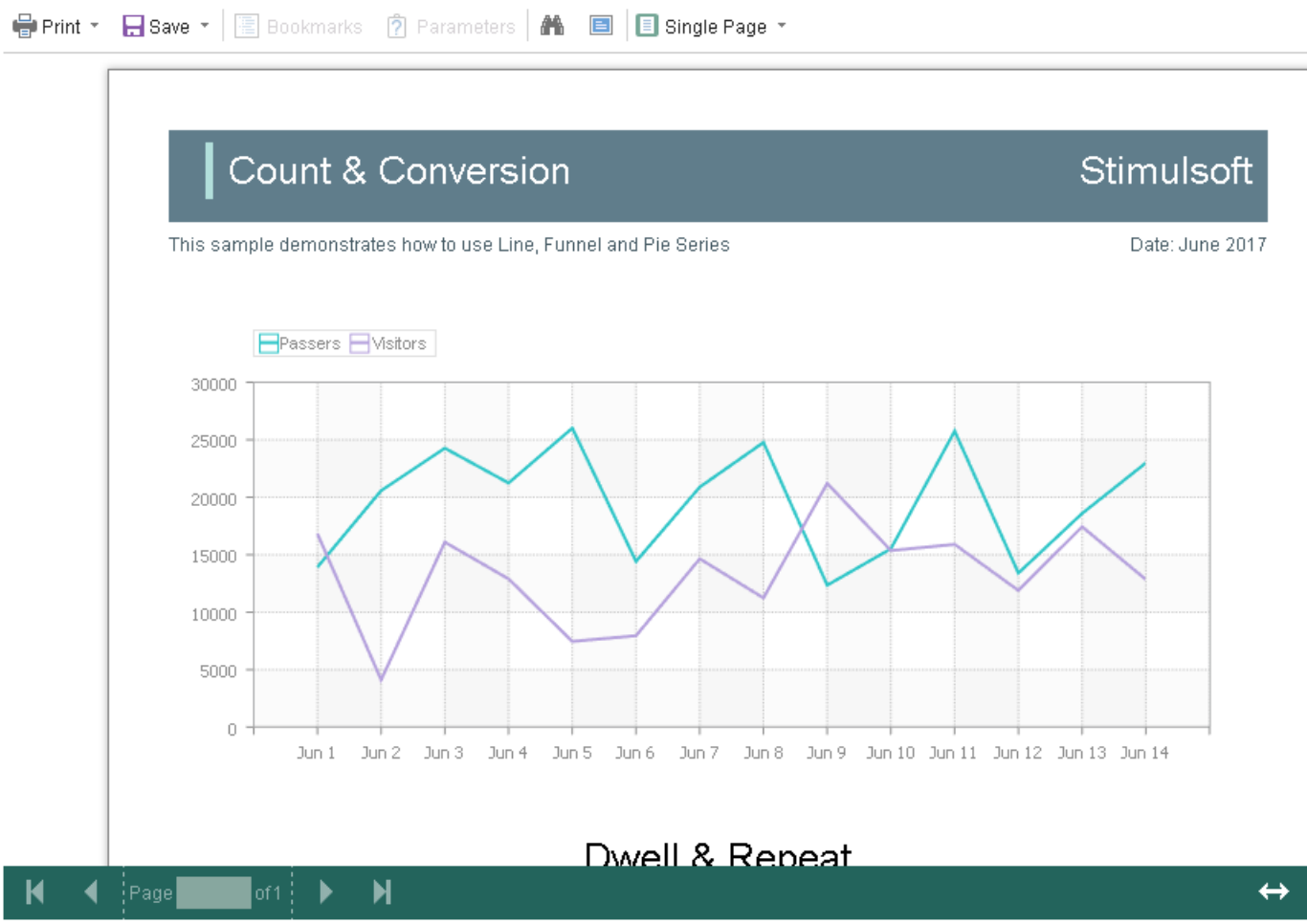
© 2003-2024 Stimulsoft

By default, the viewer has only the top toolbar on which all the report controls are located. If necessary, the toolbar can be split into top and bottom parts. The top panel will contain the menu for printing and exporting the report and the buttons for working with parameters and bookmarks. The bottom panel will contain controls to switch between the report pages and setting the zoom of pages. To enable this mode, enable the **ToolbarDisplayMode** property. It has values **Simple** and **Separated**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      ScrollbarsMode = true
   },
   Toolbar =
   {
      DisplayMode = StiToolbarDisplayMode.Separated
   }
})
...
```

© 2003-2024 Stimulsoft



In addition, it is possible to set the appearance parameters for the main elements of the viewer. For example, you can change the font and color of the control panel inscriptions of the viewer, set the background of the viewer, set the color of page borders, etc. Below is a list of available properties that change the appearance of the viewer, and their default values.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      BackgroundColor = Color.White,
      PageBorderColor = Color.Blue,
      ShowPageShadow = true
   },
   Toolbar =
   {
      BackgroundColor = Color.White,
```

© 2003-2024 Stimulsoft

```
        BorderColor = Color.Gray,
        FontColor = Color.Black,
        FontFamily = "Arial"
    }
})
...
```

5.1.7 Basic Features

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The main features of the viewer include the following operations:

- Displaying the report.
- Switching between the report pages.
- Changing the scale.
- Displaying the preview mode.

All specified operations are performed in the AJAX mode without restarting the browser page. For the correct work of these operations, you should define a special ViewerEvent action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
    Actions =
    {
        ViewerEvent = "ViewerEvent"
    }
})
...
```

HomeController.cs

```
...
public IActionResult ViewerEvent()
{
    // Some code before viewer event
    // ...

    return StiNetCoreViewer.ViewerEventResult(this);
}
```

© 2003-2024 Stimulsoft

```
. . .
```

Information

This action is mandatory. Without it, the correct operation of the viewer is not possible.

The **ViewerEvent** action returns a prepared HTML page of the report (or set of pages), built taking into account the current state of the viewer. If necessary, you can change the parameters of the current report in the specified action and update the report data in case of interactive actions of the viewer.

5.1.8 Printing Reports

Information

Please note that the print option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides several options for printing a report. Each has its advantages and disadvantages.

Print to PDF
Printing will be done by exporting the report to PDF. The advantages are greater accuracy of positioning and printing of the report elements compared to other printing options. Among the drawbacks, one can mention the mandatory presence of a plug-in installed in a web browser for viewing PDF files (modern browsers have embedded PDF viewer and printer).

Print with Preview
The report will be printed in a separate pop-up browser window in HTML. The report can be previewed and then sent to the printer or copied to another location as text or HTML code. Advantages - cross-browser compatibility when printing, no need to install special plug-ins. The disadvantage is the relatively low accuracy of the position of the report elements due to the peculiarities of the implementation of

© 2003-2024 Stimulsoft

HTML formatting.

Print without Preview
The report will be printed directly to the printer without preview. After selecting this menu item, the system print dialog is displayed. Since printing in this mode is carried out in the HTML format, the print quality is similar to printing a report with a preview.

Information

When printing to the **HTML format**, you should check the compliance of report page settings and printer parameters (paper size, orientation, margins, indents), as well as check your browser print settings, such as margins, headers, footers, watermarks printing, color printing.

The print function does not require additional settings for the viewer. If you need to perform any actions before printing a report, you can define a special **PrintReport** action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      PrintReport = "PrintReport"
   }
})
...
```

HomeController.cs

```
...
public IActionResult PrintReport()
{
   // Some code before print
   // ...

   return StiNetCoreViewer.PrintReportResult(this);
}
...
```

© 2003-2024 Stimulsoft

Print setup

If you choose printing a report in the viewer panel, a menu with printing options is displayed. The **HTML5 Viewer** component is able to force the required printing mode. To do this, set the **PrintDestination** property to one of the following values of the **StiPrintDestination** enumeration.

- **Default** – the menu will be displayed (the default property value);
- **Pdf** – print to the PDF format;
- **Direct** – printing to the HTML format directly to the printer, the system print dialog will be displayed;
- **WithPreview** – print to the HTML format with preview in a pop-up window.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      PrintDestination = StiPrintDestination.Default
   }
})
...
```

The **HTML5 Viewer** component is able to completely disable report printing. To do this, set the value of the **ShowPrintButton** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowPrintButton = false
   }
})
...
```

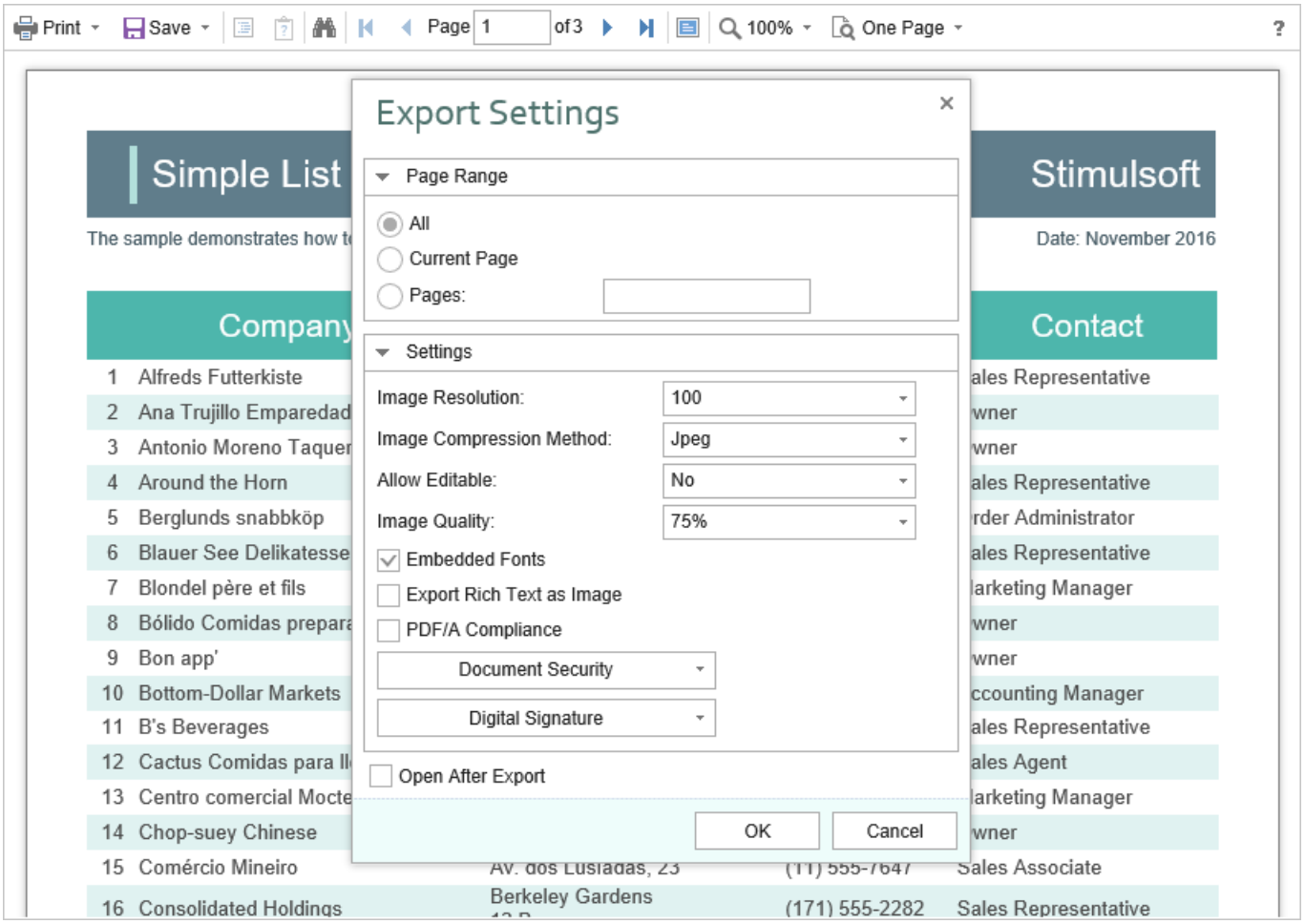
5.1.9 Exporting Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

The **HTML5 Viewer** component allows you to export the displayed report to three dozen various formats, such as **PDF**, **HTML**, **Word**, **Excel**, **XPS**, **RTF**, **images**, **text**, and others. You may export the dashboard to PDF, Excel, image files.



The export function does not require additional settings for the viewer. If you need to perform any actions before exporting the report, you can define a special **ExportReport** action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      ExportReport = "ExportReport"
```

© 2003-2024 Stimulsoft

```
   }
})
...
```

HomeController.cs

```
...
public IActionResult ExportReport()
{
   // Some code before export
   // ...

   return StiNetCoreViewer.ExportReportResult(this);
}
...
```

Export settings

Each report export format of the **HTML5 Viewer** component has a lot of settings, and each setting has its default values. Sometimes you need to set other default values. For this purpose, a special **DefaultSettings** property of the viewer is used. It is a container for all the default export settings.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Exports =
   {
      DefaultSettings =
      {
         ExportToPdf =
         {
            ImageQuality = 0.75f,
            ImageFormat = Stimulsoft.Report.Export.StiImageFormat.Color
         },
         ExportToHtml =
         {
            ExportMode = Stimulsoft.Report.Export.StiHtmlExportMode.Div,
            UseEmbeddedImages = true
         }
      }
   }
})
...
```

If it is required, you can completely hide export dialogs. Exporting will always be done with default settings. For this, it is enough to set the value of the

© 2003-2024 Stimulsoft

ShowExportDialog property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Exports =
   {
      ShowExportDialog = false
   }
})
...
```

The **HTML5 Viewer** component contains 30+ export formats, and sometimes you need to disable unwanted formats. This allows you to simplify UI and the use of the viewer. To disable unused export formats, it is enough to set the values for the corresponding properties of the viewer listed in the list below to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Exports =
   {
      ShowExportToDocument = true,
      ShowExportToPdf = true,
      ShowExportToXps = true,
      ShowExportToPowerPoint = true,
      ShowExportToHtml = true,
      ShowExportToHtml5 = true,
      ShowExportToMht = true,
      ShowExportToText = true,
      ShowExportToRtf = true,
      ShowExportToWord2007 = true,
      ShowExportToOpenDocumentWriter = true,
      ShowExportToExcel = true,
      ShowExportToExcelXml = true,
      ShowExportToExcel2007 = true,
      ShowExportToOpenDocumentCalc = true,
      ShowExportToCsv = true,
      ShowExportToDbf = true,
      ShowExportToXml = true,
      ShowExportToDif = true,
      ShowExportToSylk = true,
      ShowExportToImageBmp = true,
      ShowExportToImageGif = true,
      ShowExportToImageJpeg = true,
      ShowExportToImagePcx = true,
      ShowExportToImagePng = true,
      ShowExportToImageTiff = true,
      ShowExportToImageMetafile = true,
      ShowExportToImageSvg = true,
```

© 2003-2024 Stimulsoft

```
      ShowExportToImageSvgz = true
   }
})
...
```

The **HTML5 Viewer** component can completely disable the export menu. To do this, set the value of the **ShowSaveButton** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowSaveButton = false
   }
})
...
```

5.1.10 Viewing Modes

The **HTML5 Viewer** component has two modes for displaying reports - with and without scrollbars. By default, the view mode without scrollbars is set. To enable the scrollbar view mode, set the value of the **ScrollbarsMode** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      ScrollbarsMode = true
   }
})
...
```

In the first mode (without scrollbars), the viewer displays a page or report as a whole, automatically stretching the preview space. If the width and height are specified, the viewer will truncate the page that is out of bounds. In the second mode, unlike the first one, when the page is out of bounds of the viewer's size, no truncation will be performed. Scrollbars will appear, using which you can view the entire page or report.

Information

© 2003-2024 Stimulsoft

In the report mode with scrollbars, you should set the height of the viewer, otherwise the default height will be set to **650 pixels**.

The **HTML5 Viewer** component provides the full-screen report and dashboard mode. By default, the standard view mode is enabled, the viewer has the specified dimensions in the settings. To enable the full-screen mode, set the **FullScreenMode** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      FullScreenMode = true
   }
})
...
```

Also, to enable or disable the full-screen mode, you can use the corresponding button on the control panel of the viewer.

The **HTML5 Viewer** component has three modes to display reports - page-by-page, entire report, and tabular display of report pages. To control the modes, the **ViewMode** property is used. It can have one of the specified values - **SinglePage**, **Continuous**, **MultiplePages**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ViewMode = StiWebViewMode.SinglePage
   }
})
...
```

The **HTML5 Viewer** component supports interaction on a regular PC display and works with a touchscreen of screens and mobile devices. The **InterfaceType**

© 2003-2024 Stimulsoft

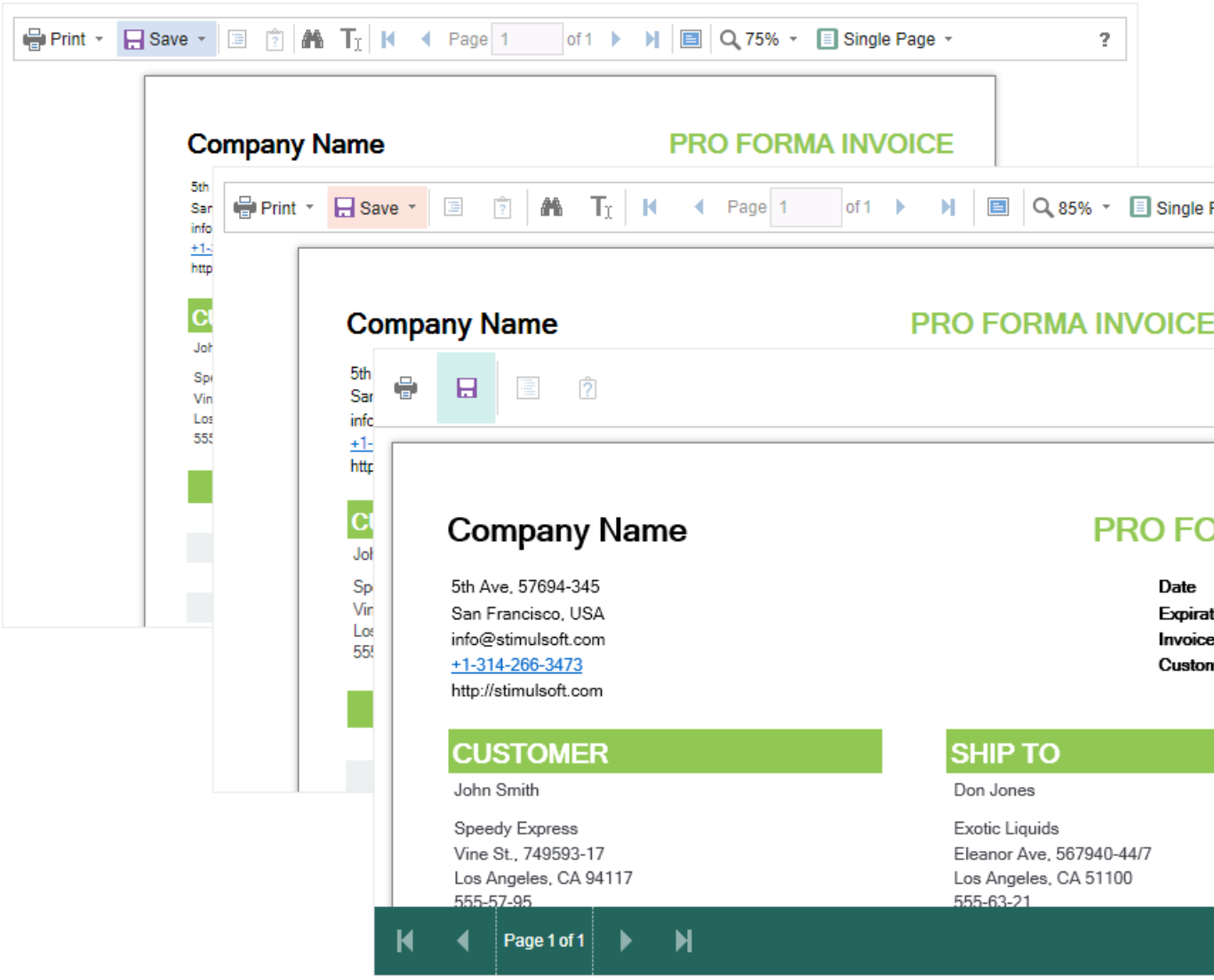
property allows controlling the interface modes. The property can have one of the following values:

- **Auto** – the viewer's interface is determined automatically depending on the device the report is displayed on. That is the default value.
- **Mouse** – the standard interface with a mouse control will be used for all the screen types.
- **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability.
- **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with smartphones.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      InterfaceType = StiInterfaceType.Auto
   }
})
...
```

© 2003-2024 Stimulsoft



5.1.11 Work with Parameters

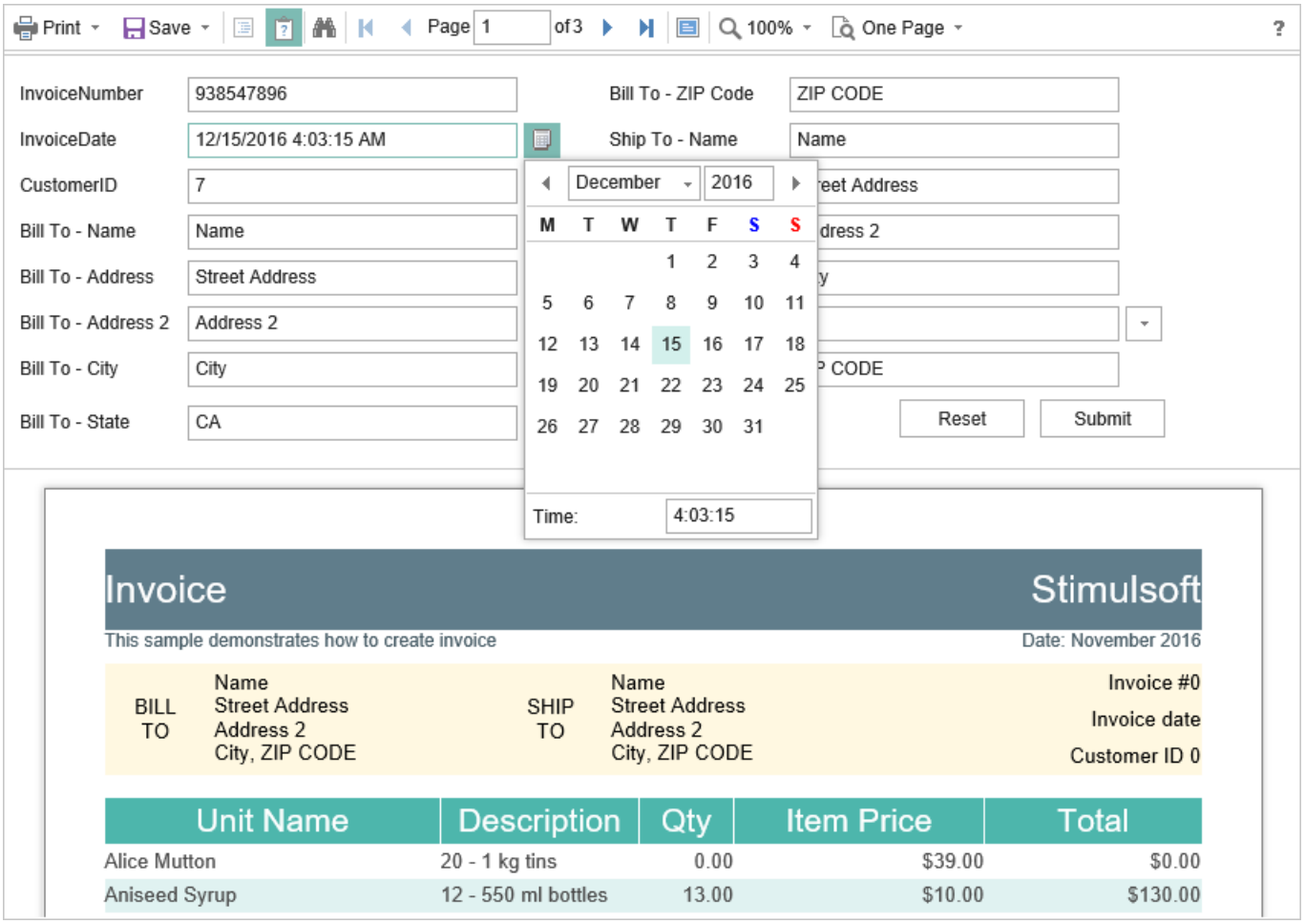
Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To work with report parameters in the **HTML5 Viewer**, there is a special settings panel. To add a parameter to the panel, you need to define a variable in a report requested by the user. When viewing a report in the viewer, such a variable will be

© 2003-2024 Stimulsoft

automatically added to the settings panel. It supports all report variables (normal variables, date and time, borders, lists, etc.).



To work with reports with parameters, no additional viewer settings are required. If you need to perform some actions before applying the parameters, you can define a special **Interaction** action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      Interaction = "ViewerInteraction"
   }
})
...
```

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   // Some code before any interaction
   // ...

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

This action is called during any interactive actions of the viewer. If you need to perform any actions only when applying report parameters, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side on any request and contain all necessary information and states of the client part of the viewer. To determine the type of action of the viewer, it is enough to check the **Action** property of the viewer parameters.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      // Some code before apply parameters
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

If you do not need to work with parameters, you can completely disable this feature. To do this, use the **ShowParametersButton** property in the **Toolbar** section of properties, which should be set to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowParametersButton = false
```

© 2003-2024 Stimulsoft

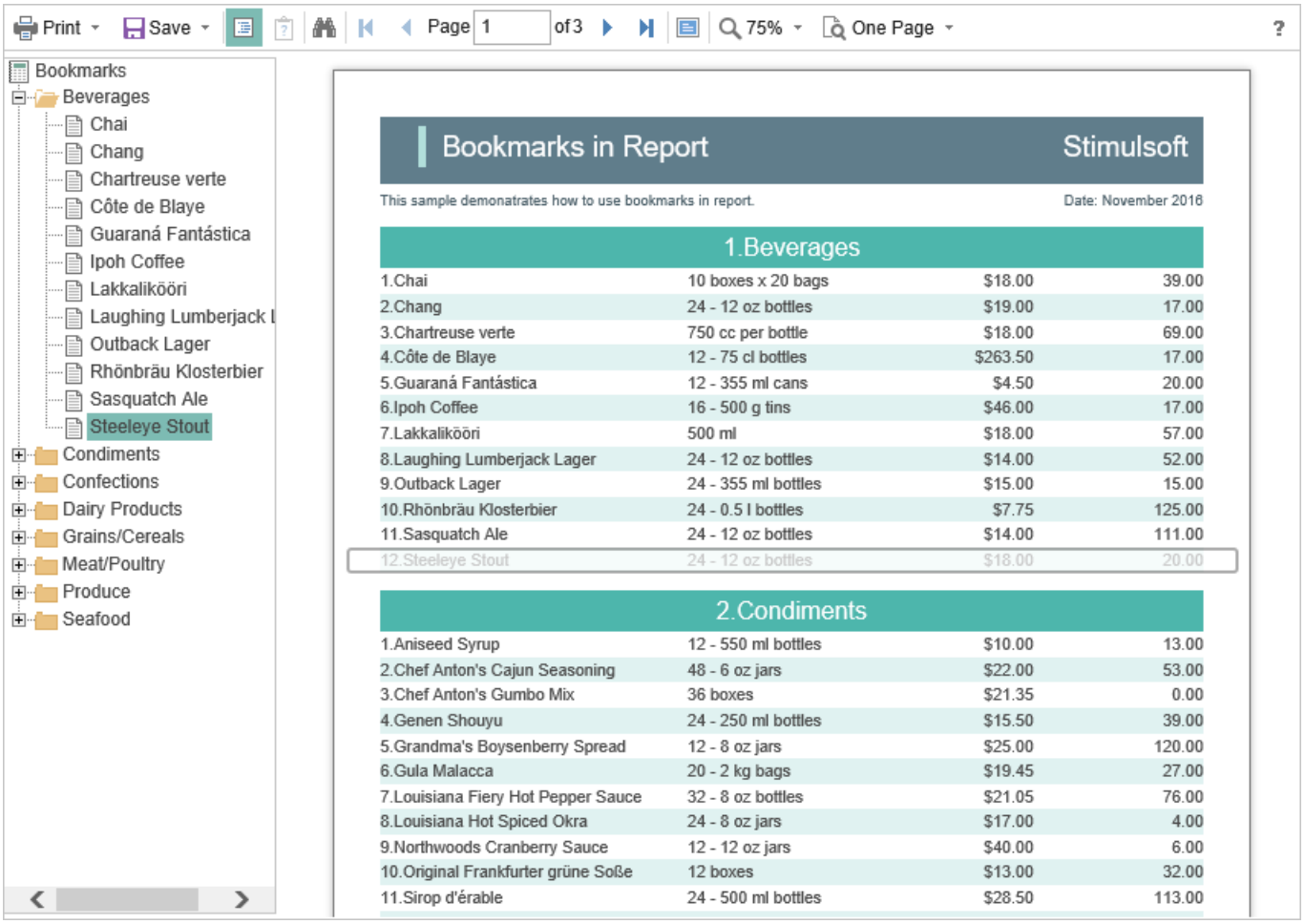
```
    }
})
...
```

Information

The options panel will not be displayed with such a viewer configuration, even if the parameters are present in the displayed report.

5.1.12 Work with Bookmarks

The **HTML5 Viewer** component supports report bookmarks. A panel with bookmarks will be displayed when displaying such a report on the left side of the page. When you select a bookmark of the report, the viewer will carry out an automatic transition to the specified page, and the report item with a bookmark is highlighted.



© 2003-2024 Stimulsoft

By default, the bookmarks bar width is 180 pixels. The **HTML5 Viewer** component allows you to change this value. For this, the **BookmarksTreeWidth** property, which value is specified in pixels, is used.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      BookmarksTreeWidth = 200
   }
})
...
```

If work with report bookmarks is not required, you can disable this feature. For this, set the **ShowBookmarksButton** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowBookmarksButton = false
   }
})
...
```

Information

In this case, report bookmarks will not be displayed, even if they are present in the displayed report. This property has no effect on printing and exporting reports.

When printing a report with bookmarks, the bookmark tree will be hidden. If you want to print bookmarks with the report, it is necessary to set the **BookmarksPrint** property to **true**.

Index.cshtml

```
...
```

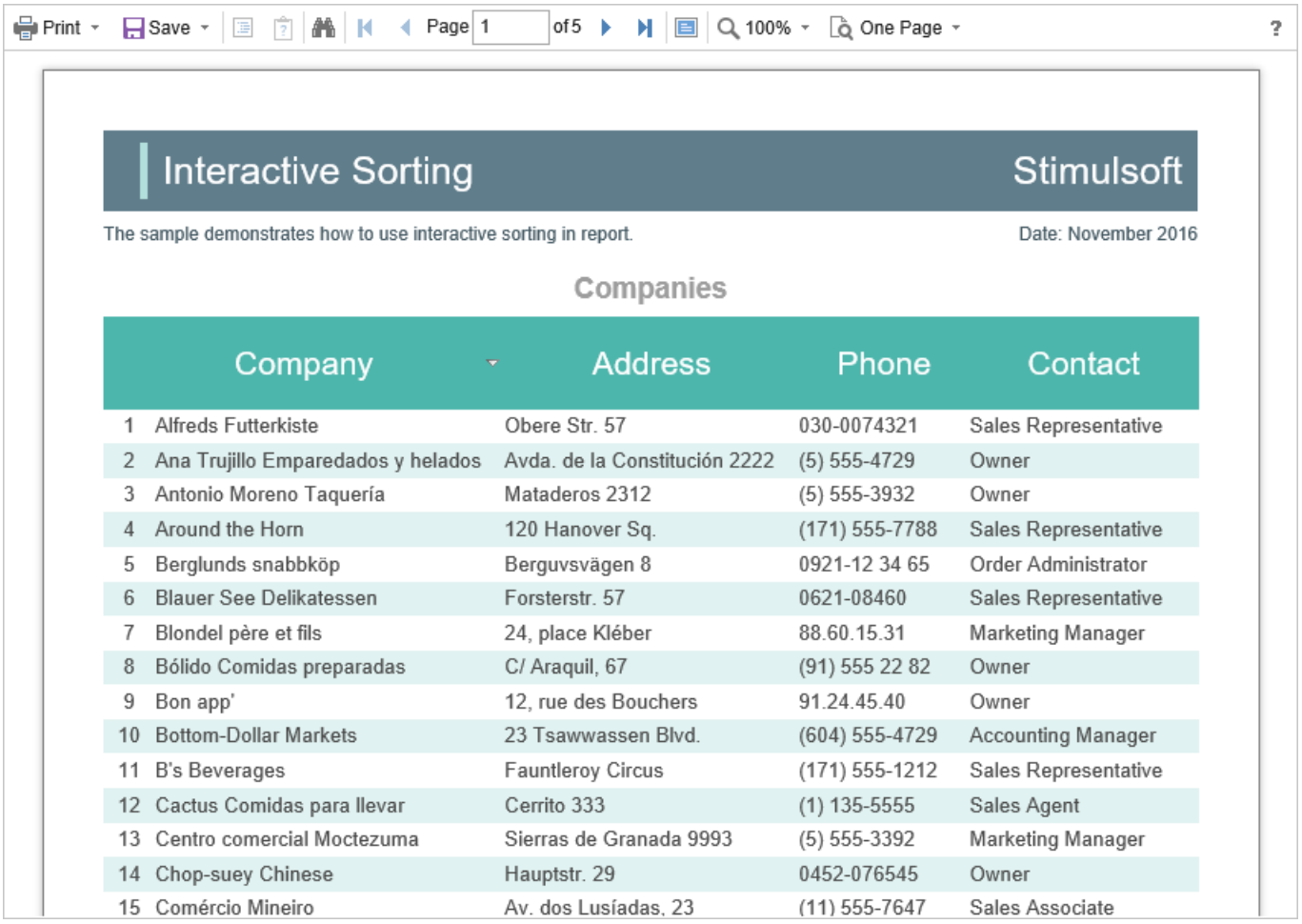
© 2003-2024 Stimulsoft

```
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions () {
   Appearance =
   {
      BookmarksPrint = true
   }
})
...
```

5.1.13 Dynamic Sorting, Collapsing, and Drill-Down

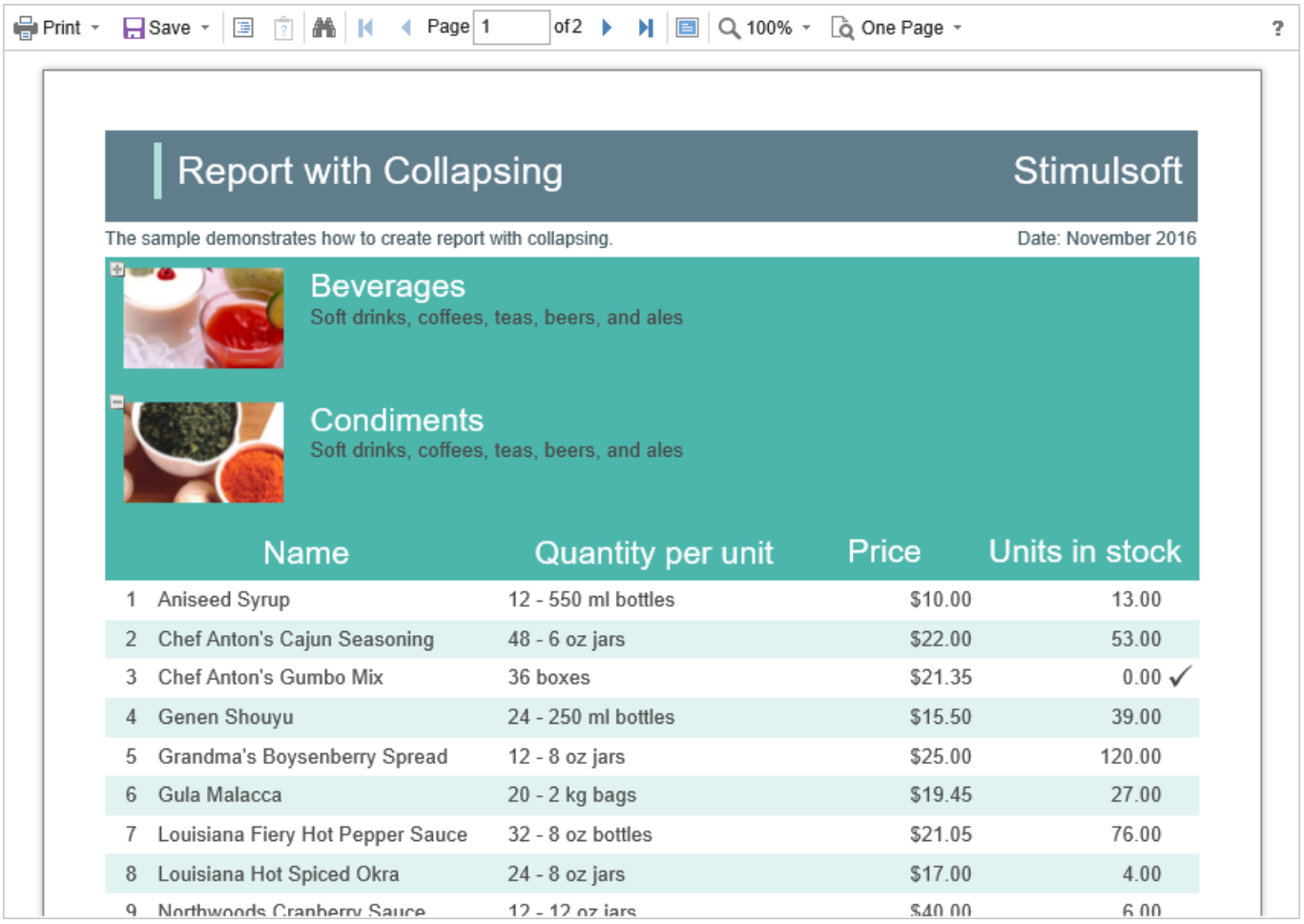
The **HTML5 Viewer** component supports dynamic sorting, collapsing, and drill-down of reports. Dynamic sorting provides the ability to change the direction of sorting in a rendered report. To do this, click on the component that has dynamic sorting enabled. Dynamic sorting is carried out in the following directions - **Ascending** and **Descending**. Each time the component is clicked, the sorting direction is reversed.

Multi-level sorting is allowed in the report. To do this, hold down the **Ctrl** key and sequentially click on the sorted components in the report. To reset sorting, you can click on any sorted component without holding down the **Ctrl** key.



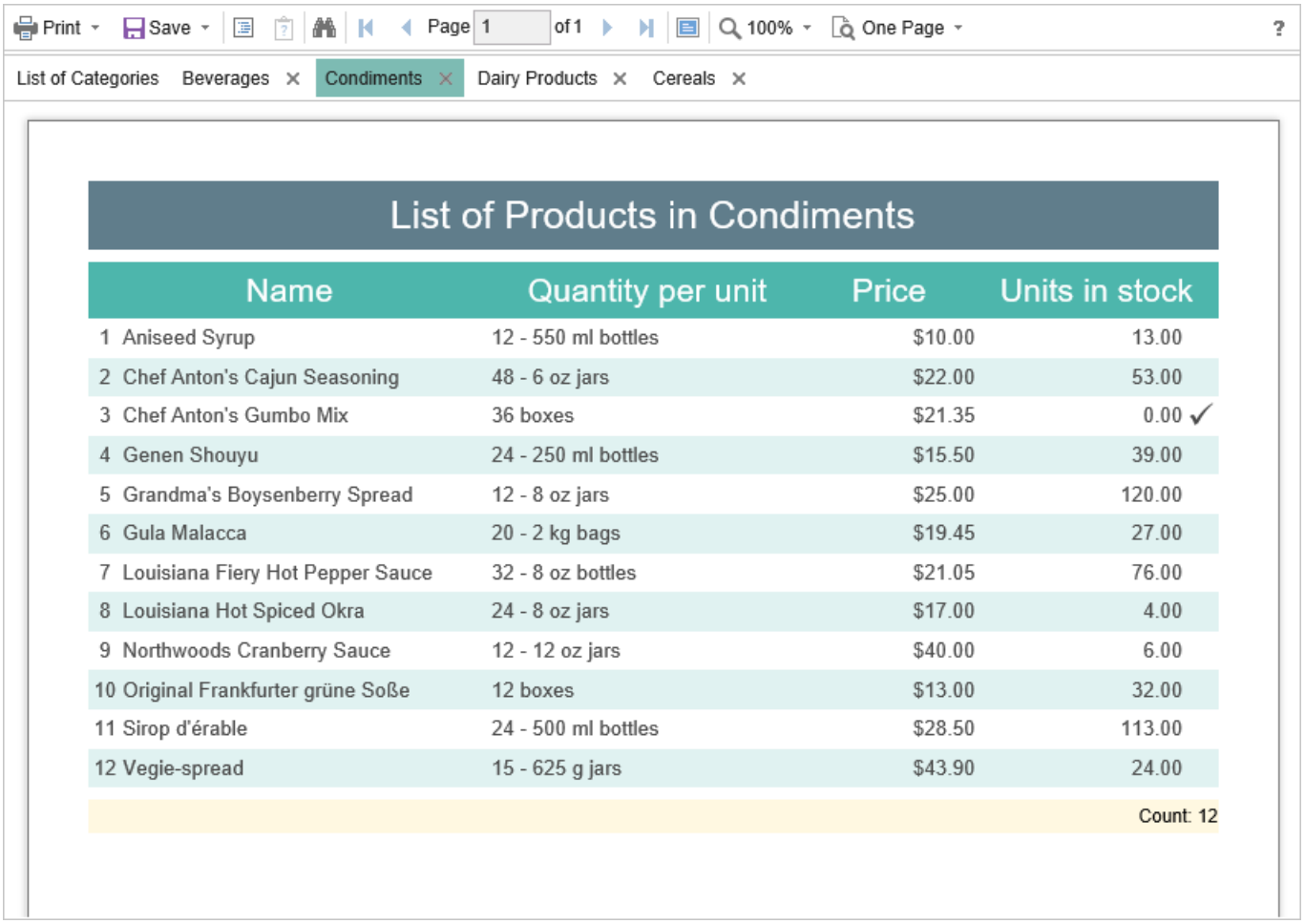
© 2003-2024 Stimulsoft

A report with dynamic collapsing is an interactive report in which blocks can collapse/expand their content when you click on the block title. Report elements, which can be collapsed/expanded, are indicated by special icons - **[-]** or **[+]**.



When using drill-down, under the main panel of the viewer, the drill-down panel with tabs for drill-down reports will be displayed. The currently displayed report will be highlighted.

© 2003-2024 Stimulsoft



To work with dynamic sorting, collapsing, and drill-down reports, no additional viewer settings are required. A special **Interaction** action is used to perform any actions before sorting, collapsing, or drill-down of the report. It will be called when interactive action of the viewer.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      Interaction = "ViewerInteraction"
   }
})
...
```

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   // Some code before any interaction
   // ...

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

To get the type of action, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side by any request and contain all necessary information and states of the client part of the viewer. For each type of interactivity, the viewer has a certain type of action:

- **Sorting** – when using column sorting;
- **DrillDown** – when using drill-down in reports;
- **Collapsing** – when using collapsing report blocks.

HomeController.cs

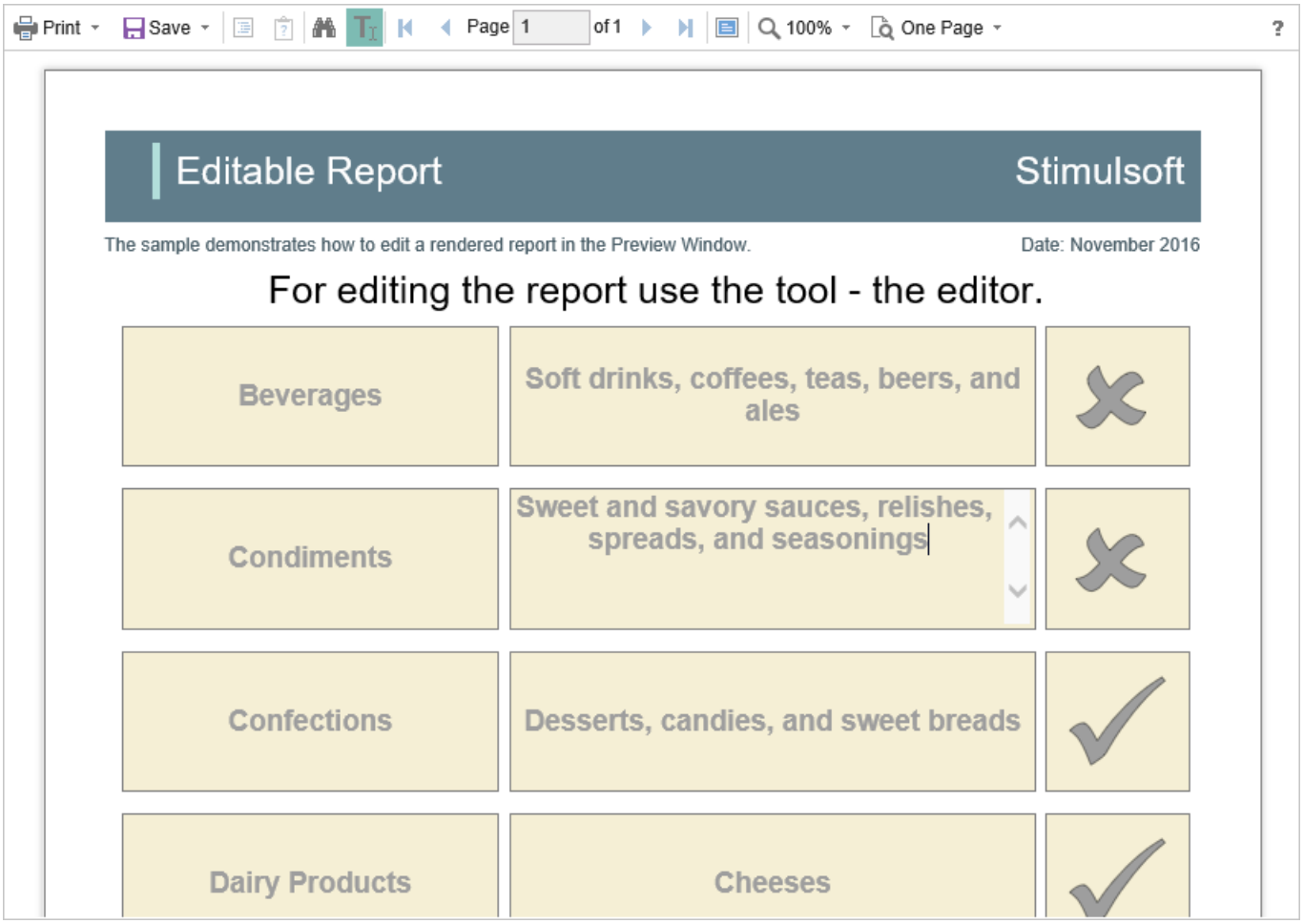
```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   switch (requestParams.Action)
   {
      case StiAction.Sorting:
         break;

      case StiAction.DrillDown:
         break;

      case StiAction.Collapsing:
         break;
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

5.1.14 Editing Report

The **HTML5 Viewer** component has the ability to edit report items, such as text boxes and checkboxes. You should mark the required components as editable in the report template for the editing to be possible. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be

© 2003-2024 Stimulsoft

applied to the report.



For the report edit mode, no special settings of the viewer required.

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

5.1.15 Sending Report by Email

Information

Please note that the Send Report by Email option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides the ability to send reports by email. To activate this feature, you should set the **ShowSendEmailButton** property of the viewer to **true** and define the **EmailReport** action.

Index.cshtml

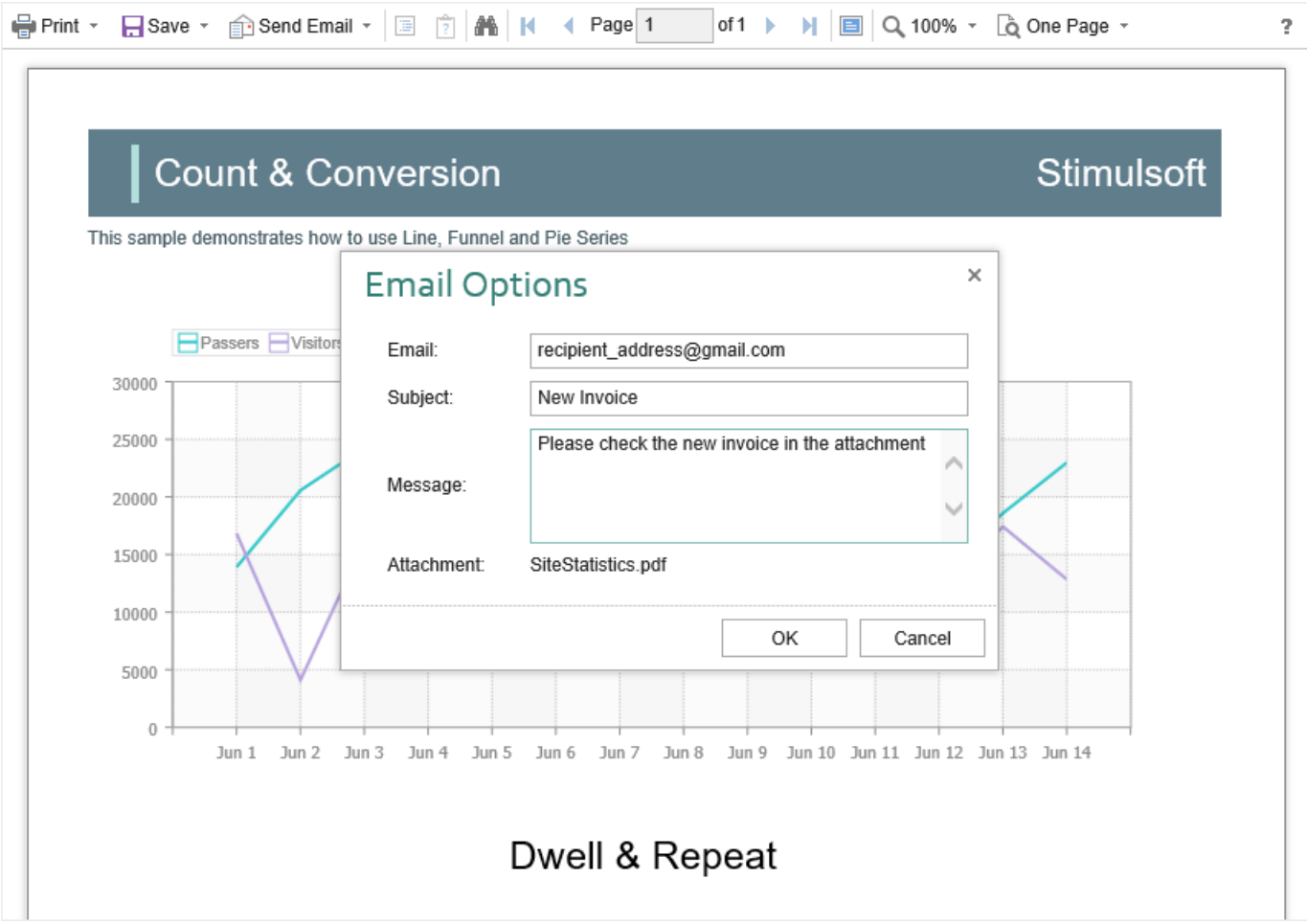
```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      EmailReport = "EmailReport"
   },
   Toolbar =
   {
      ShowSendEmailButton = true
   }
})
...
```

HomeController.cs

```
...
public IActionResult EmailReport()
{
   StiEmailOptions options = StiNetCoreViewer.GetEmailOptions(this);

   // Passed from the viewer, can be checked and changed
   // options.AddressTo = "";
   // options.Subject = "";
   // options.Body = "";

   // Should be filled here
   options.AddressFrom = "admin_address@test.com";
   options.Host = "smtp.test.com";
   options.Port = 465;
   options.UserName = "admin_address@test.com";
   options.Password = "admin_password";

   // options.CC.Add("email@test.com");
   // options.BCC.Add("email@test.com");
   // options.EnableSsl = true;

   return StiNetCoreViewer.EmailReportResult(this, options);
}
...
```

© 2003-2024 Stimulsoft

When sending a report by email, the menu to select the attachment format is displayed. It corresponds to the menu for selecting the format for exporting the report. After selecting the format, the dialog to enter the send email parameters, such as the recipient's email, subject, and text of the message, is displayed.



After confirmation of sending the email, the above described **EmailReport** event will be called. You can check and correct the data entered in this form. The exported report file will be attached to the email automatically.

The **HTML5 Viewer** component allows you to set default values for the send email form. The **DefaultEmailAddress**, **DefaultEmailSubject**, and **DefaultEmailMessage** properties can be used for this. By default, these properties are empty.

Index.cshtml

© 2003-2024 Stimulsoft

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Email =
   {
      DefaultEmailAddress = "recipient_address@gmail.com",
      DefaultEmailSubject = "New Invoice",
      DefaultEmailMessage = "Please check the new invoice in the
      attachment"
   }
})
...
```

5.1.16 Calling Designer from Viewer

The **HTML5 Viewer** component has the ability to call the report designer. The special **Design** button in the toolbar of the viewer (the button is disabled by default) should be used. To use this feature, you should set the **ShowDesignButton** property to true and define the **DesignReport** event handler.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      DesignReport = "DesignReport"
   },
   Toolbar =
   {
      ShowDesignButton = true
   }
})
...
```

HomeController.cs

```
...
public IActionResult DesignReport()
{
   StiReport report = StiNetCoreViewer.GetReportObject(this);
   ViewBag.ReportName = report.ReportName;

   return View("Designer");
}
...
```

Information

The viewer does not run the designer. It only calls the specified action, in which you can get all the necessary parameters. Then, in action, you can implement a

© 2003-2024 Stimulsoft

redirection to another View, which contains the report designer.

5.1.17 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to use the server cache to store rendered reports. If you do not use caching, you should load the report, connect data, and render it again every time you request a page. If you use caching, the previously rendered report will be loaded from the cache every time you refresh the page.

When using caching, it should be taken into account that every report saved in the cache takes up server memory and, with a large number of requests to reports, this can become a critical issue. Therefore, you need to choose between two options: low memory requirements but high in performance or low-performance requirements but high in memory.

You need to connect modules to work with a session or cache on the server-side to use caching. To do this, just add the following services to the project in the start file of a project.

Startup.cs

```
...
public void ConfigureServices(IServiceCollection services)
{
   services.AddMemoryCache();
   services.AddSession();
   services.AddMvc();
}
...
```

You can manage caching with the following properties.

© 2003-2024 Stimulsoft

The CacheMode property

This property of the viewer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled. Each action of the viewer requires loading the report from the file and, if it is a report template, then render it;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this cache (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** – for caching, the current session is used. The report is saved as a packed string in this cache.

The CacheItemPriority property

This property sets the priority of the report stored in the server's cache. It affects the automatic clearing of the server memory in case of a lack of memory. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes for which you want to save the report in the server cache. If you use caching and the requested report is not found in the cache (the objects storage time has expired), then it will be requested again using a special **GetReport** event, then connect the report data and render it.

StiCacheHelper

The **HTML5 Viewer** component provides the ability to define your methods of working with report caching. For this purpose, a special class **StiCacheHelper** is used. It contains methods for obtaining a report from the cache and saving the report to

© 2003-2024 Stimulsoft

the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetReport**, **SaveReport** and **RemoveReport**.

HomeController.cs

```
...
public class ViewerController : Controller
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override StiReport GetReport(string guid)
      {
         string path =
         System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
         ), guid);
         if (System.IO.File.Exists(path))
         {
            StiReport report = new StiReport();
            string packedReport = System.IO.File.ReadAllText(path);
            if (guid.EndsWith("template"))
            report.LoadPackedReportFromString(packedReport);
            else report.LoadPackedDocumentFromString(packedReport);

            return report;
         }
         return null;

         //return base.GetReport(guid);
      }

      public override void SaveReport(StiReport report, string guid)
      {
         string packedReport = guid.EndsWith("template") ?
         report.SavePackedReportToString() :
         report.SavePackedDocumentToString();
         string path =
         System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
         ), guid);
         System.IO.File.WriteAllText(path, packedReport);

         //base.SaveReport(report, guid);
      }

      public override void RemoveReport(string guid)
      {
         var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
         guid);
         if (File.Exists(path))
            File.Delete(path);
      }
   }

   static ViewerController()
   {
      StiNetCoreViewer.CacheHelper = new StiMyCacheHelper();
```

© 2003-2024 Stimulsoft

```
    }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as a value of the static **StiNetCoreViewer.CacheHelper** property in the controller constructor.

Information

If report caching is disabled (the **CacheMode** property of the viewer is set to **None**), the specified class will not be used.

5.1.18 Additional Methods

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

For **HTML5 Viewer**, several additional methods are used to get the object of the currently viewed report, parameters of the current state of the viewer, and other useful data. These methods can be used in any actions of the viewer.

The GetReportObject() method

Returns the report object with which the viewer is currently working. It is possible to perform the necessary actions - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the viewer, specifying it as a parameter in the resulting action method.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
```

© 2003-2024 Stimulsoft

```
   StiReport report = StiNetCoreViewer.GetReportObject(this);
   report.ReportName = "MyReportName";

   return StiNetCoreViewer.InteractionResult(this, report);
}
...
```

The GetRouteValues() method

Returns values for URLs with which the viewer page was opened. Thus, it is possible to get the initial collection of run page parameters in any viewer action and use these values for any checks and conditions.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   RouteValueDictionary routeValues =
   StiNetCoreViewer.GetRouteValues(this);

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

You can also get values of URL parameters by parameter name, specifying it as the parameter of the called action of the viewer.

HomeController.cs

```
...
public IActionResult ViewerInteraction(string id)
{
   return StiNetCoreViewer.InteractionResult(this);
}
...
```

The GetFormValues() method

Returns the values of the form that initiated (opened by the POST request) a page of the viewer. Thus, it is possible to get a collection of form parameters in any action of the viewer.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   NameValueCollection formValues = StiNetCoreViewer.GetFormValues(this);

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

By default, this feature is disabled to optimize requests of the client-side of the viewer to the server. To enable it, set the **PassFormValues** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Server =
   {
      PassFormValues = true
   }
})
...
```

The GetRequestParams() method

Returns all parameters of the current state of the viewer passed to the server-side. They can be useful for determining the type of action that the viewer is currently executing - for example, to determine the type of export and all action parameters.

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiNetCoreViewer.GetReportObject(this);

      // Some action with report for the PDF export
      // ...

      return StiNetCoreViewer.ExportReportResult(this, report);
   }

   return StiNetCoreViewer.ExportReportResult(this);
```

© 2003-2024 Stimulsoft

```
}
...
```

You can change the values of some parameters. After making changes, for the correct operation of the viewer, you should transfer the modified parameter object to the input of the resulting method.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      requestParams.Interaction.Variables["Variable1"] = "MyValue";
      return StiNetCoreViewer.InteractionResult(this, requestParams);
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

The GetExportSettings() method

Returns all the parameters of the current report export. The type of the parameter object will correspond to the type of export selected in the viewer menu. Any export parameters can be changed and passed to the input of the resulting method. In this case, the report will be exported with the parameters transferred.

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiExportSettings settings = StiNetCoreViewer.GetExportSettings(this);
   if (settings.GetExportFormat() == StiExportFormat.Pdf)
   {
      StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
      pdfSettings.EmbeddedFonts = true;
      pdfSettings.AllowEditable = StiPdfAllowEditable.No;
      return StiNetCoreViewer.ExportReportResult(this, settings);
   }

   return StiNetCoreViewer.ExportReportResult(this);
}
...
```

© 2003-2024 Stimulsoft

The MapPath() and MapWebRootPath() methods

Returns the absolute path, respectively, to the application or wwwroot directory. You can use this to upload report templates files, data files, etc. These methods are located in the StiNetCoreHelper static class.

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

5.1.19 Export and Printing from Code

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** provides the ability to print reports in various ways and export reports to various formats. These actions are performed using the viewer menu. If you want to print or export a report using the code, for example, in the controller action, you can use the special **StiNetCoreReportResponse** class. This class contains a set of static methods that allow you to print or export a report from the code, and the report viewer is not required.

Index.cshtml

```
...
@Html.ActionLink("Print Report from Code", "PrintReport")
<br />
@Html.ActionLink("Export Report from Code", "ExportReport")
...
```

HomeController.cs

© 2003-2024 Stimulsoft

```
...
private StiReport LoadSimpleList()
{
   DataSet dataSet = new DataSet();
   dataSet.ReadXml(Server.MapPath("Reports/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(Server.MapPath("Reports/SimpleList.mrt"));
   report.RegData(dataSet);

   return report;
}

public IActionResult PrintReport()
{
   StiReport report = LoadSimpleList();

   return StiNetCoreReportResponse.PrintAsPdf(report);
   //return StiNetCoreReportResponse.PrintAsHtml(report);
}

public IActionResult ExportReport()
{
   StiReport report = LoadSimpleList();

   return StiNetCoreReportResponse.ResponseAsPdf(report);
   //return StiNetCoreReportResponse.ResponseAsExcel2007(report);
   //return StiNetCoreReportResponse.ResponseAsText(report);
   //StiNetCoreReportResponse.ResponseAsJson(report);
}
...
```

The **StiNetCoreReportResponse** class contains methods for printing in PDF and HTML formats and methods to export the report in any of the supported formats. As arguments, methods can take various export settings, displaying modes and options for saving received files.

5.1.20 Timeout

When working with the **StiNetCoreViewer** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

© 2003-2024 Stimulsoft

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Server =
   {
      CacheTimeout = 10
   }
})
...
```

Using the cache will increase the speed of the report viewer. See the chapter Caching for more information.

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Server =
   {
      RequestTimeout = 30
   }
})
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection and data sources in the report.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
```

© 2003-2024 Stimulsoft

```
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent"
   }
   })
...
```

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiNetCoreViewer.GetReportResult(this, report);
}

public IActionResult ViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}
...
```

5.1.21 Viewer Settings

The **HTML5 Viewer** is configured using properties that are located in the **StiNetCoreViewerOptions** class. All properties are divided into groups. Some of the groups contain subgroups for ease of use. The following is an example of setting the properties of the viewer.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Theme = StiViewerTheme.Office2022WhiteTeal,
   Localization = "Localization/en.xml",
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent"
   },
   Appearance =
   {
      InterfaceType = StiInterfaceType.Auto,
      ScrollbarsMode = true,
      ShowTooltips = false
   },
   Exports =
   {
      DefaultSettings =
      {
```

© 2003-2024 Stimulsoft

```
        ExportToPdf =
        {
           CreatorString = "Company Name",
           ImageQuality = 0.75f
        }
     },
     ShowExportToDbf = false,
     ShowExportToDif = false
   }
})
...
```

Please note that all dashboard elements have their own save options and full-screen buttons for preview. There are no special options to control displaying them, but they can be disabled through the properties of the element. The code below should be added after loading the report before passing it to the viewer.

HomeController.cs

```
...
var dbsElementInteraction = (report.GetComponentByName("RegionMap1") as
Stimulsoft.Report.Dashboard.IStiElementInteraction).DashboardInteraction;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowFullScreenButton =
false;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowSaveButton = false;
...
```

Main settings (without groups)

Name	Description
Theme	Sets the viewer theme. The list of available themes can be found in the **StiViewerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Sets the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used. It is built into the viewer and does not require additional XML files.
Width	Sets the width of the component in the required

© 2003-2024 Stimulsoft

	units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default width is 100%.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the automatic height is set depending on the size of the report page, or 650 pixels in the view mode of the viewer with scrollbars.

Actions

Name	Description
GetReport	Specifies the name of the action method for preparing the rendered report. Specifies the name of the action method for preparing the constructed report. If report caching is enabled, this action will be called only once when the report is requested or if the requested report is not found in the server cache.
PrintReport	Specifies the name of the action method of report printing. This is not relevant when viewing dashboards.
ExportReport	Specifies the name of the action method of the export the report to the specified format.
EmailReport	Specifies the name of the action method of sending the report by email. This is not relevant when viewing dashboards.
Interaction	Specifies the name of the action method for the viewer to work with interactive operations, such as using parameters, dynamic sorting, collapsing, and drill-down.

© 2003-2024 Stimulsoft

DesignReport	Specifies the name of the action method to go to the specified view by clicking the Design button on the viewer panel.
ViewerEvent	Specifies the name of the action method of basic viewer events and the processing actions of the viewer, such as printing and exporting a report, working with parameters, and interactivity, if these actions are not specified separately. In addition, this action is used to load scripts and styles of the viewer. This action is mandatory.

Server

Name	Description
Controller	Specifies the name of the report controller for the report viewer. If this property is not specified, then the current controller will be used to process requests.
RouteTemplate	Sets the route template that is returned when the report viewer actions are executed. If the property is not set, then the MVC project template will be used instead. The default value of the property is null.
RequestTimeout	Sets the response timeout from the server in seconds, after which an error will be generated. The default value is 20 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the report since the last action of the viewer. The default value is 20 minutes.
CacheMode	Sets the report caching mode. It can take one of

© 2003-2024 Stimulsoft

	the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled, the report will be reloaded each time using the **GetReport** event; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as storage, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or the server session will be automatically re-saved after a certain period of time when the viewer is idle (every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the viewer mode in which relative URLs are used for AJAX requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value that specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the viewer. If **false**, only the necessary parameters

© 2003-2024 Stimulsoft

	are used to request the resources of the viewer. This corresponds to the correct work of the browser cache. By default, the property is set to **true**.
PassQueryParametersToReport	Enables using all the URL parameters of the request as the variable values. The variable names must match the parameters. The default value of the property is false.
PassFormValues	Enables passing the values of the POST form to the client-side, if these values are required to be used in the actions of the viewer. If you enable this property, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
ShowServerErrorPage	Enables displaying an HTML page with the details of the error that occurred on the server-side. When the property is enabled, the details of the error will be displayed in the viewer window. If the property is disabled, only the numeric error code and a short error text in the dialog box will be displayed. By default, the property is set to **true**.
UseCompression	Enables compression of the viewer requests into the GZip stream. Enables compression of the viewer requests into the GZip stream. That allows to decrease the amount of internet traffic but slows down the viewer slightly. The default value of the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server-side. The following resources are supported - scripts, styles, and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
UseLocalizedCache	Sets a value that enables the use of a different cache depending on the selected localization.

© 2003-2024 Stimulsoft

	The default value of the property is **false**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Sets the path to the CSS file of the viewer's styles. The standard styles of the chosen theme will not be loaded if this property has got a value. The default value of the property is an empty string.
BackgroundColor	Sets the background color of the viewer. By default, it is set to **White**.
PageBorderColor	Sets the border color of the viewer. By default it is set to **Gray**.
RightToLeft	Sets the **Right to Left** mode for viewer controls. By default, the property is set to **false**. By default, the property is set to **false**.
FullScreenMode	Sets the full-screen display mode of the viewer. By default, the property is set to **false**.
ScrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to **false**.
OpenLinksWindow	Sets the target window for opening links contained in the report. By default, the property is set to **Blank** (new window).
OpenExportedReportWindow	Sets the target window for opening the export file from the viewer. By default, the property is set to **Blank** (new window).
DesignWindow	Sets the destination window for launching the report designer. The default value of the property is **Self** (which is the current window).

© 2003-2024 Stimulsoft

ShowTooltips	Enables showing tips for the viewer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enables showing links to online documentation for the viewer controls. By default, the property is set to **true**.
ShowDialogsHelp	Sets a value that indicates that showing or hiding the help button in dialogs. By default, the property is set to **true**.
PageAlignment	Sets the position of the report page in the viewer window. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – the page will be aligned left; › **Center** – the page will be centered (default value); › **Right** – the page will be aligned right.
ShowPageShadow	Enables displaying shadow for report pages. By default, the property is set to **true**.
BookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default, the property is set to **false**.
BookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
ParametersPanelPosition	Specifies the position of the report parameters panel. It can take one of the following **StiParametersPanelPosition** enumeration values: › **Top** - the panel will be docked to the top margin (default value); › **Left** - the panel will be docked to the left margin.
ParametersPanelMaxHeight	Sets the maximum height of the parameters bar in pixels. By default, the maximum height is 300

© 2003-2024 Stimulsoft

	pixels.
ParametersPanelColumnsCount	Sets the number of columns to display report parameters. By default, there are 2 columns.
ParametersPanelSortDataItems	Gets or sets a value which indicates that variable items will be sorted. By default, the property is set to **true**.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
InterfaceType	Sets the type of interface used for the viewer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the viewer's interface is determined automatically depending of the device that is report is displayed on. That is the default value. › **Mouse** – the standard interface with a mouse control will be used for all the screen types. › **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability. › **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.
AllowMobileMode	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to **false**, then the mobile view will not be used. If the option is set to **true**, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option

© 2003-2024 Stimulsoft

	is set to **true**.
ChartRenderType	Sets the displaying mode of charts on the report page. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages. It can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
DatePickerFirstDayOfWeek	Sets the first day of the week for the date picker. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Monday** – the first day of the week is Monday (default value); › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
AllowTouchZoom	Sets ability to change the scale of the report page by using the two-fingers gesture (Pinch to

© 2003-2024 Stimulsoft

	Zoom) for the touch-screens. The default value of the property is **true**.
ShowReportIsNotSpecifiedMessage	Sets a value which indicates that 'The report is not specified' message will be shown. The default value of the property is **true**.
PrintToPdfMode	Sets the Print to PDF mode. It has the following values: › **StiPrintToPdfMode.Hidden** - hidden print mode (default value); › **StiPrintToPdfMode.Popup** - the PDF document will be displayed before printing in a pop-up window.
ImagesQuality	Gets or sets the image quality that will be used on the viewer page. It has the following values: › **StiImagesQuality.Low** - low quality, used to speed up loading reports and saves memory; › **StiImagesQuality.Normal** - normal quality, suitable for most cases (default value); › **StiImagesQuality.High** - high quality, used for ultra high-definition displays, but may slow down the loading of pages.
CombineReportPages	Sets a value which indicates that if a report contains several pages, then they will be combined in preview. By default, the property is set to **false**.

Toolbar

Name	Description
Visible	Enables displaying the viewer toolbar. By default, the property is set to **true**.
DisplayMode	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the **StiToolbarDisplayMode** enumeration: › **Simple** - all controls are located on the same

© 2003-2024 Stimulsoft

	control panel (default value); › **Separated** - the control panel is split into top and bottom panels.
BackgroundColor	Specifies the background color of the viewer toolbar. The default color of the selected theme is used.
BorderColor	Specifies the border color of the viewer toolbar. The default color of the selected theme is used.
FontColor	Specifies the text color for the toolbar and the viewer menu. The default color of the selected theme is used.
FontFamily	Specifies the font for the toolbar and the viewer menu. The default font of the selected theme is used.
Alignment	Sets the alignment mode for the controls on the viewer toolbar. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – elements will be aligned left; › **Center** – elements will be centered; › **Right** – elements will be aligned right; › **Default** – the alignment depends on the RightToLeft property (default value).
ShowButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default, the property is set to **true**.
ShowPrintButton	Enables showing the button - **Print** - on the viewer toolbar. By default, the property is set to **true**.
ShowOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to

© 2003-2024 Stimulsoft

	true.
ShowSendEmailButton	Enables showing the button - **Send Email** - on the viewer toolbar. By default, the property is set to **false**. Also, you should add the EmailReport action.
ShowFindButton	Enables showing the button - **Find** - on the viewer toolbar. By default, the property is set to **true**.
ShowBookmarksButton	Enables showing the button - **Bookmarks** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
ShowParametersButton	Enables showing the button - **Parameters** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.
ShowResourcesButton	Enables showing the button - **Resources** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.
ShowEditorButton	Enables showing the button - **Editor** - on the viewer toolbar. By default, the property is set to **true**.
ShowFullScreenButton	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports or dashboards. . By default, the property is set to **true**.
ShowFirstPageButton	Enables showing the button - **First Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowPreviousPageButton	Enables showing the button - **Previous Page** - on the viewer toolbar. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowCurrentPageControl	Enables showing the current report page indicator. By default, the property is set to **true**.
ShowNextPageButton	Enables showing the button - **Next Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowLastPageButton	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowZoomButton	Enables showing the button to select the report zoom. By default, the property is set to **true**.
ShowViewModeButton	Enables showing the button to select the view mode of the report page. By default, the property is set to **true**.
ShowDesignButton	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **false**.
ShowAboutButton	Enables showing the button - **About** - on the viewer toolbar. By default, the property is set to **true**.
ShowRefreshButton	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to **true**.
ShowPinToolbarButton	Enables displaying of the **Pin Toolbar** button on the viewer's toolbar. The button is available only in the Mobile mode of the viewer's interface. The default value of the property is **true**.
PrintDestination	Sets the report printing mode. It can take one of the following values of the **StiPrintDestination** enumeration: › **Default** – a menu with a choice of printing modes will be displayed (default value); › **Pdf** – printing will be done in the PDF format; › **Direct** – printing will be done to the HTML

© 2003-2024 Stimulsoft

	format directly to the printer, the system print dialog will be displayed; › **PopupWindow** – printing will be done in the HTML format via the preview window of the report.
ViewMode	Sets the mode for displaying report pages. It can take one of the following **StiWebViewMode** enumeration values: › **SinglePage** - displays one page of the report selected in the toolbar of the viewer (default value); › **Continuous** - displays all pages of the report; › **MultiplePages** - displays all report pages as a table.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of the following values: › **StiZoomMode.PageWidth** – when the viewer runs, the zoom, necessary to display the report by the page width, will be set; › **StiZoomMode.PageHeight** – when the viewer runs, the zoom, necessary to display the report by the page height, will be set.
MenuAnimation	Enables animation when the viewer menu shows/hides. By default the property is set to **true**.
ShowMenuMode	Sets the display mode of the viewer menu. It can take one of the following values of the **StiShowMenuMode** enumeration: › **Click** – shows menu by mouse click (default value); › **Hover** – shows menu by hovering the mouse cursor.

© 2003-2024 Stimulsoft

AutoHide	Enables auto-hiding of the viewer's toolbar. The property will work only for the Mobile mode of the viewer's interface. The default value of the property is **false**.

Export

Name	Description
DefaultSettings	This group of properties provides the ability to specify the default export settings for each export type. These settings will be applied to the export dialogs when the viewer runs or to the report, if export dialogs are disabled.
StoreExportSettings	Enables saving selected settings in the export dialogs. Settings will be stored in browser cookies. By default the property is set to **true**.
ShowExportDialog	Enables showing the export options dialog box. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
ShowExportToDocument	Enables the export menu item - **Document File**. By default, the property is set to **true**.
ShowExportToPdf	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToXps	Enables the export menu item - **Microsoft XPS File**. By default, the property is set to **false**.
ShowExportToPowerPoint	Enables the export menu item - **Microsoft PowerPoint 2007/2010 File**. By default, the property is set to **true**.
ShowExportToHtml	Enables the export menu item - **HTML File**. By default, the property is set to **true**.
ShowExportToHtml5	Enables the export menu item - **HTML5 File**. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
ShowExportToMht	Enables the export menu item - **MHT Web Archive**. By default, the property is set to **true**.
ShowExportToText	Enables the export menu item - **Text File**. By default, the property is set to **true**.
ShowExportToRtf	Enables the export menu item - **Rich Text File**. By default, the property is set to **true**.
ShowExportToWord2007	Enables the export menu item - **Microsoft Word 2007/2010 File**. By default, the property is set to **true**.
ShowExportToOpenDocumentWriter	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to **true**.
ShowExportToExcel	Enables the export menu item - **Microsoft Excel File**. By default, the property is set to **true**.
ShowExportToExcelXml	Enables the export menu item - **Microsoft Excel Xml File**. By default, the property is set to **true**.
ShowExportToExcel2007	Enables displaying the **Microsoft Excel 2007/2010 File** export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToOpenDocumentCalc	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to **true**.
ShowExportToCsv	Enables the export menu item - **CSV File**. By default, the property is set to **true**.
ShowExportToDbf	Enables the export menu item - **DBF File**. By default, the property is set to **true**.
ShowExportToXml	Enables the export menu item - **XML File**. By default, the property is set to **true**.
ShowExportToDif	Enables the export menu item - **Data Interchange Format (DIF) File**. By default, the

© 2003-2024 Stimulsoft

	property is set to **true**.
ShowExportToSylk	Enables the export menu item - **Symbolic Link (SYLK) File**. By default, the property is set to **true**.
ShowExportToJson	Enables the export menu item - **JSON File**. By default, the property is set to **true**.
ShowExportToImageBmp	Enables displaying the **BMP Image** export menu item when viewing reports, and the **BMP Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageGif	Enables displaying the **GIF Image** export menu item when viewing reports, and the **GIF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageJpeg	Enables displaying the **JPEG Image** export menu item when viewing reports, and the **JPEG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePcx	Enables displaying the **PCX Image** export menu item when viewing reports, and the **PCX Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePng	Enables displaying the **PNG Image** export menu item when viewing reports, and the **PNG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageTiff	Enables displaying the **TIFF Image** export menu item when viewing reports, and the **TIFF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvg	Enables displaying the **Scalable Vector Graphics (SVG) File** export menu item when viewing reports, and the **Scalable Vector Graphics (SVG) File** item when viewing dashboards. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToImageSvgz	Enables displaying the **Compressed SVG (SVGZ) File** export menu item when viewing reports, and the **Compressed SVG (SVGZ) File** item when viewing dashboards. By default, the property is set to **true**.
ShowOpenAfterExport	Enables displaying the **Open After Export** parameter in export settings menu. By default, the property is set to **true**.

Email

Name	Description
ShowEmailDialog	Enables displaying settings for sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server side in the **EmailReport** action. By default the property is set to **true**.
ShowExportDialog	Enables displaying export options dialog box when sending email. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
DefaultEmailAddress	Sets the default recipient email, i.e. the address to which the email with the attached report will be sent.
DefaultEmailSubject	Sets the default email subject (header).
DefaultEmailMessage	Sets the default email message (text).

5.2 HTML5 Designer

YouTube

Watch videos for working with .NET Core HTML5 Designer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

© 2003-2024 Stimulsoft

Samples

See on GitHub examples of working with the .NET Core HTML5 Designer component. All examples are separate projects, grouped into one solution for Visual Studio.

The **HTML5 Designer** (**StiNetCoreDesigner**) component is designed to create reports in the web browser. You do not need to install the .NET Framework, ActiveX components or any special plug-ins on the client side. All that is needed is any modern Web browser.

With help of **HTML5 Designer** you can create, edit, save and preview reports on any computer with any operating system installed. Since the designer only uses HTML and JavaScript technologies, it can be run on devices where there is no Flash or Silverlight support - tablets, smartphones. Also, the designer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Designer** component uses the AJAX technology to perform all actions on reports, which allows you to get rid of reloading the entire page, save Web traffic and speed up work. The report engine built using the .NET Core technology is used to render reports. This is a cross-platform technology. It allows you to deploy the application on servers that use the operating systems like Windows, macOS, and Linux.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Designer** in a Web project, you need to install the NuGet package of Stimulsoft.Reports.Web.NetCore:

- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Reports.Web.NetCore in the search bar on the Browse tab;

© 2003-2024 Stimulsoft

› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

To add the ability to create and edit dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web.NetCore (this package is associated with the package Stimulsoft.Reports.Web.NetCore. If it is missed it will be installed automatically):
› Select "Manage NuGet Packages ..." in the context menu of the project;
› Specify Stimulsoft.Dashboards.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

- How this Works?
- Activation
- Editing Reports and Dashboards
- Creating New Reports and New Dashboards
- Saving Reports and Dashboards
- Preview
- Settings
- Additional Features of Preview
- Timeout
- Localization
- Using Themes
- Caching
- Additional Methods

5.2.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the designer, you need to place the **StiNetCoreDesigner** component on the page, set the necessary settings to it, and set the necessary actions in the view controller. When the report designer runs, the following actions occur:

© 2003-2024 Stimulsoft

› The .NET Core component generates HTML and JavaScript code that is necessary for displaying and running the designer;
› When the component is output, the JavaScript method is launched. It requests the report template on the server side displays it in the designer window;
› Various actions in the designer (for example, report preview, saving the report template, export reports, sorting, drill-down etc.) calls a certain action on the server side, in which you can perform the necessary manipulations with the report.

5.2.2 Activation

YouTube

Watch videos which show how to activate the ASP.NET Core HTML5 Designer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiNetCoreDesigner** component.

HomeController.cs

```
...
//Activation with using license code
public class HomeController : Controller
{
   static HomeController()
   {
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";
   }
}

//Activation with using license file
public class HomeController : Controller
{
   public HomeController(IHostingEnvironment hostEnvironment)
   {
      var path = Path.Combine(hostEnvironment.ContentRootPath, "Content\
      \license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
...
```

© 2003-2024 Stimulsoft

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

5.2.3 Editing Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To edit a report template, you need to add the **StiNetCoreDesigner** component to the page, specify the minimum necessary settings for it, and define the necessary actions in the view controller.

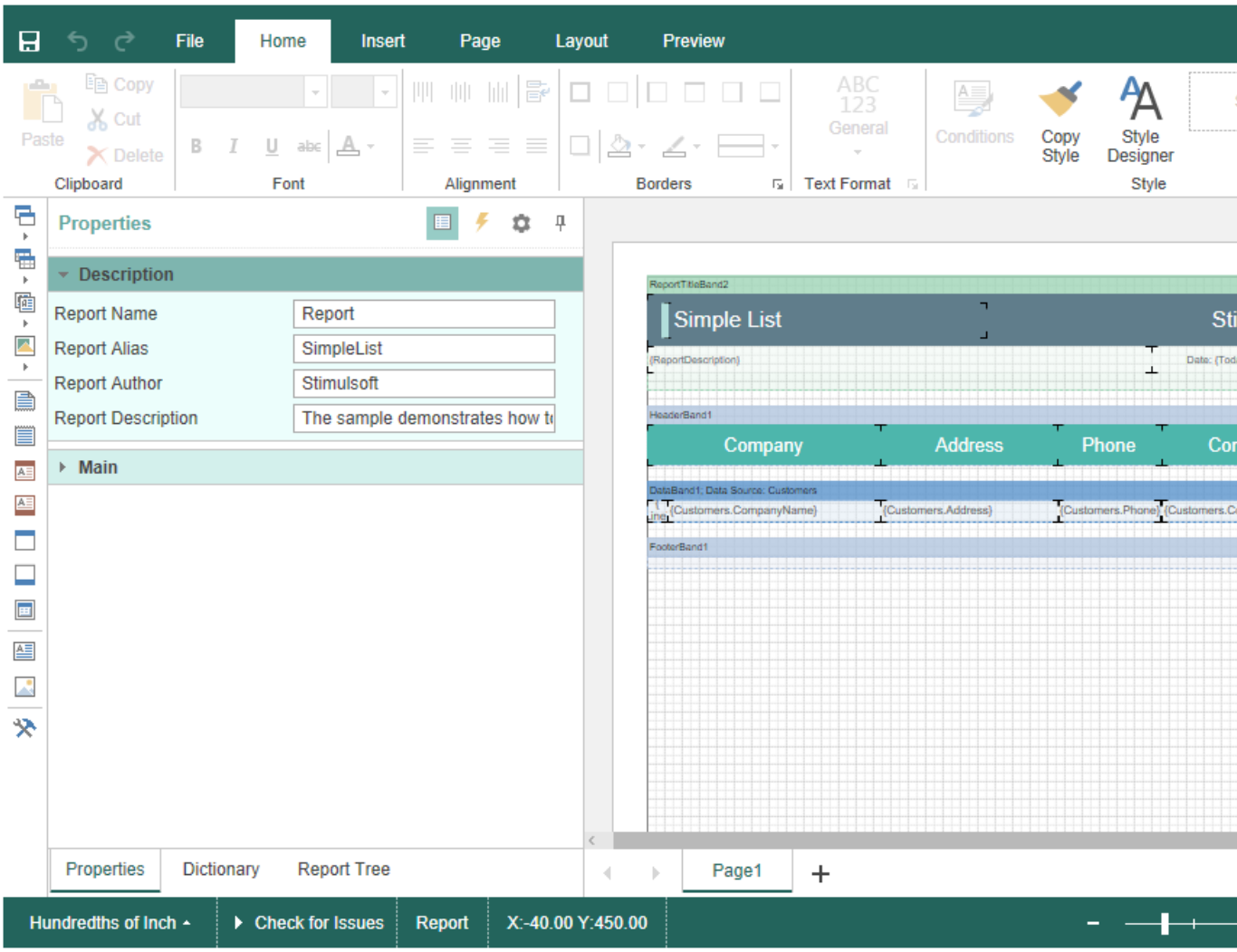
Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      DesignerEvent = "DesignerEvent"
   }
})
...
```

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
   //report.Load(StiNetCoreHelper.MapPath(this, "Reports/Dashboard.mrt"));

   return StiNetCoreDesigner.GetReportResult(this, report);
}

public IActionResult DesignerEvent()
{
   return StiNetCoreDesigner.DesignerEventResult(this);
}
...
```

© 2003-2024 Stimulsoft



The **GetReport** action is used to load an editable report template. It is called automatically after the report designer is loaded. The **DesignerEvent** action is designed to process various additional designer actions, such as working with data and components, previewing reports and others.

Information

The **DesignerEvent** action is mandatory. Without it, the correct work of the designer is impossible.

© 2003-2024 Stimulsoft

5.2.4 Creating New Reports and New Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the report designer with a new (empty) report, it is enough to create a new report in the **GetReport** action and return it to the designer. If necessary, you can load data for the report, or perform any other necessary actions.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetReport = "GetReport"
   }
})
...
```

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();

   return StiNetCoreDesigner.GetReportResult(this, report);
}
...
```

You can also create a new report using the main menu of the designer. The **CreateReport** action is used to load data for a new report or perform any other necessary actions. This action will be called when creating a new empty report or when creating a report using the wizard.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
```

© 2003-2024 Stimulsoft

```
   Actions =
   {
      CreateReport = "CreateReport"
   }
})
...
```

HomeController.cs

```
...
public IActionResult CreateReport()
{
   StiReport report = new StiReport();
   //var newDashboard = StiReport.CreateNewDashboard();

   // Register data for the new report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);
   //newDashboard.RegData(data);
   report.Dictionary.Synchronize();
   //newDashboard.Dictionary.Synchronize();

   return StiNetCoreDesigner.GetReportResult(this, report);
   //return StiNetCoreDesigner.GetReportResult(this, newDashboard);
}
...
```

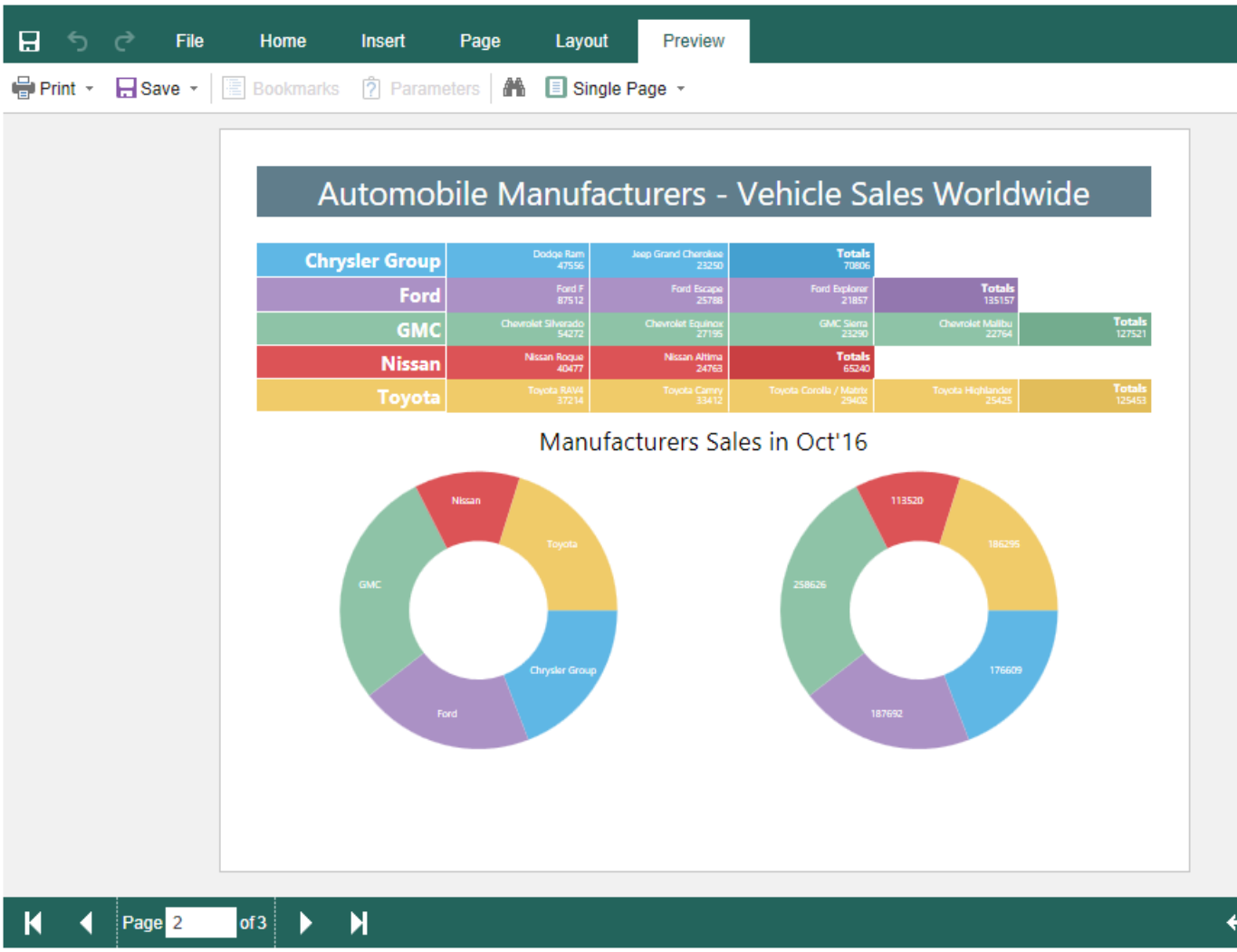
5.2.5 Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides the ability to preview reports. To preview the report, just go to the appropriate tab in the designer window. The report template will be transferred to the server side, rendered and displayed in the embedded viewer.

© 2003-2024 Stimulsoft



Before previewing the report, it is possible to perform any necessary actions, for example, connect data for the report. To do this, you can use the special **PreviewReport** action that will be called before previewing the report. The **PreviewReport** action is called before preparing and rendering a report for viewing till its saving to the cash.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      PreviewReport = "PreviewReport"
   }
})
...
```

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult PreviewReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);

   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);

   return StiNetCoreDesigner.PreviewReportResult(this, report);
}
...
```

If you need to make actions on your report immediately before displaying the report, you can use the **GetPreviewReport** action, which is called after the request of the prepared report from the cash.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetPreviewReport = "GetPreviewReport"
   }
})
...
```

HomeController.cs

```
...
public IActionResult GetPreviewReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);

   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);
   //report.IsRendered = false;

   return StiNetCoreDesigner.PreviewReportResult(this, report);
}
...
```

Information

© 2003-2024 Stimulsoft

So as in this event a prepared report for viewing is transferred, if you need to render again you should set the **report.IsRendered = false** flag.

5.2.6 Additional Features of Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The preview window of the **HTML5 Designer** component has a fully functional interactive **HTML5 Viewer** that can print and export reports, supports working with report parameters, dynamic sorting, interactive reports, collapsing and etc. To use these features, you do not need any additional settings for the report designer.

In any of the above actions, you can work with the report template, for example, change its properties and parameters, connect new data for rendering.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      ExportReport = "ExportReport"
   }
})
...
```

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);
   // ...

   return StiNetCoreDesigner.ExportReportResult(this, report);
}
...
```

© 2003-2024 Stimulsoft

Information

If you do not need any of these additional options to preview the report (for example, exporting or printing a report), you can disable them using the appropriate properties of the **HTML5 Designer** component.

5.2.7 Saving Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides two ways of saving the report which are available in the main menu and in the main panel of the designer - **Save Report** and **Save As**. In turn, each of these ways has its own modes and settings.

Saving a report and dashboard on the server side

To save the editable report on the server side, you need to set the **SaveReport** action, which will be called when you select **Save** in the main menu, or click the **Save** button on the main panel of the designer.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      SaveReport = "SaveReport"
   }
})
...
```

HomeController.cs

```
...
public IActionResult SaveReport()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);
```

© 2003-2024 Stimulsoft

```
    // Save the report template
    // ...

    return StiNetCoreDesigner.SaveReportResult(this);
}
...
```

This action returns a response to the client side of the designer about the result of saving the report. After saving the report, it is possible to display a dialog box with an error or a text message.

HomeController.cs

```
...
public IActionResult SaveReport()
{
    StiReport report = StiNetCoreDesigner.GetReportObject(this);

    // Save the report template
    // ...

    // Completion of the report saving with message dialog box
    return StiNetCoreDesigner.SaveReportResult(this, "Some message after
    saving");
    //return Content("{"infoMessage":"Some info message after saving"}");
    //return Content("{\"warningMessage\":\"Some info message after saving
    \"}");
}
...
```

You can get a report name from the designer save dialog or an original report name.

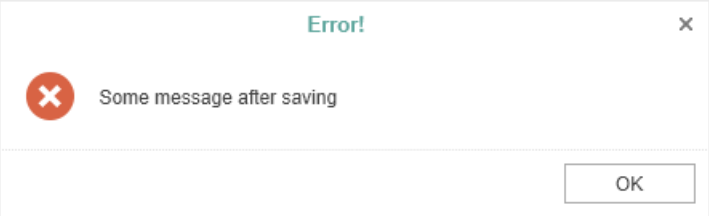
HomeController.cs

```
...
public IActionResult SaveReport()
{
    var requestParams = StiNetCoreDesigner.GetRequestParams();
    var report = StiNetCoreDesigner.GetReportObject();

    //Report name from designer save dialog
    var savingReportName = requestParams.Designer.FileName;

    //Original report name from properties
    var originalReportName = report.ReportName;

    return StiNetCoreDesigner.SaveReportResult(this, "Some message after
    saving");
}
```

© 2003-2024 Stimulsoft

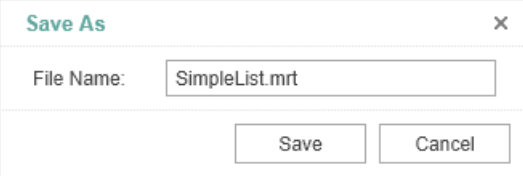
```
...
```

In this case, the dialog with the specified text will be displayed. The text can contain both an error message of saving or a warning, or any other message.



Saving reports and dashboards on the client side

To save the edited report on the client side as a file, no additional designer settings are required. It is enough to click the **Save As** main menu item. The dialog box will be displayed. In this dialog you can change the name of the report file. The file will be saved to the local disk of the computer.



The **HTML5 Designer** component provides the ability to change the behavior of the specified save option. For this purpose, the special **SaveReportAs** action is used in the designer. If you use this event, the report will be saved on the server side. Work of this event will be similar to the **SaveReport** action.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      SaveReportAs = "SaveReportAs"
   }
})
...
```

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult SaveReportAs()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);

   // Save the report template
   // ...

   return StiNetCoreDesigner.SaveReportResult(this);
}
...
```

Use the following code to get the report name from the Save dialog.

HomeController.cs

```
public IActionResult SaveReport()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);
   var requestParams = StiNetCoreDesigner.GetRequestParams(this);
   var reportName = requestParams.Designer.FileName;

   return StiNetCoreDesigner.SaveReportResult(this);
}
```

Saving settings

The report is saved in the background mode without reloading the page in the web browser window. If you need to visually control the process of saving the report, you should change the value of the **SaveReportMode** (or **SaveReportAsMode**) property of the designer to one of the three specified values - **Hidden** (default value), **Visible** or **NewWindow**.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      SaveReportAs = "SaveReportAs"
   },
   Behavior =
   {
      SaveReportAsMode = StiSaveMode.Visible
```

© 2003-2024 Stimulsoft

```
    }
})
...
```

If the **SaveReportMode** property is set to **Visible**, the report save action will be called in the current browser window in the normal (visible) mode using the POST request. If the **SaveReportMode** property is set to **NewWindow**, the report save event will be called in a new window of the web browser. By default, this property is set to **Hidden** - the report save event is called in the background using the AJAX request and is not displayed in the browser window. The same values and behavior are applicable to the **SaveReportAsMode** property.

5.2.8 Localization

The **HTML5 Designer** component supports the complete localization of its interface. Use the special **Localization** property to localize the report designer interface. As a value of this property, you should specify the path to the localization XML file (relative or absolute).

Index.cshtml

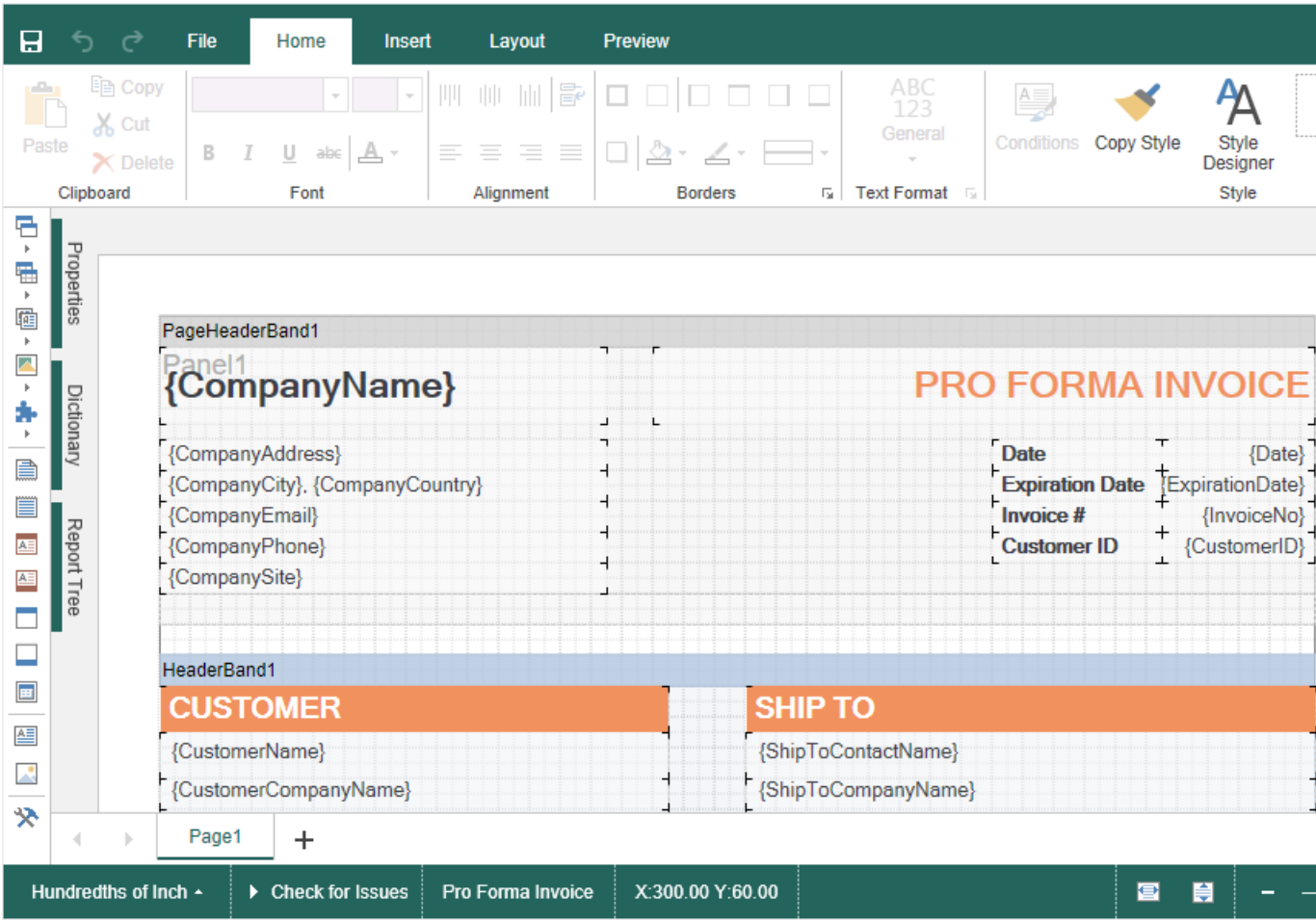
```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
    Localization = "Localization/en.xml"
})
...
```

The interface of the report designer allows you to select the necessary localization from an accessible list. To do this, set value for the **LocalizationDirectory** property as the folder in which the localization XML files are stored.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
    Localization = "Localization/en.xml",
    LocalizationDirectory = "Localization"
})
...
```

© 2003-2024 Stimulsoft



Information

If the value for the **Localization** property is set, then when you run the report designer, the localization specified in this property will always be applied. If the property value is not set, the localization selected from the list of available localizations in the report designer panel will be automatically loaded.

5.2.9 Using Themes

In the **HTML5 Designer** component you can change the appearance of visual controls. To change the theme, you should use the **Theme** property.

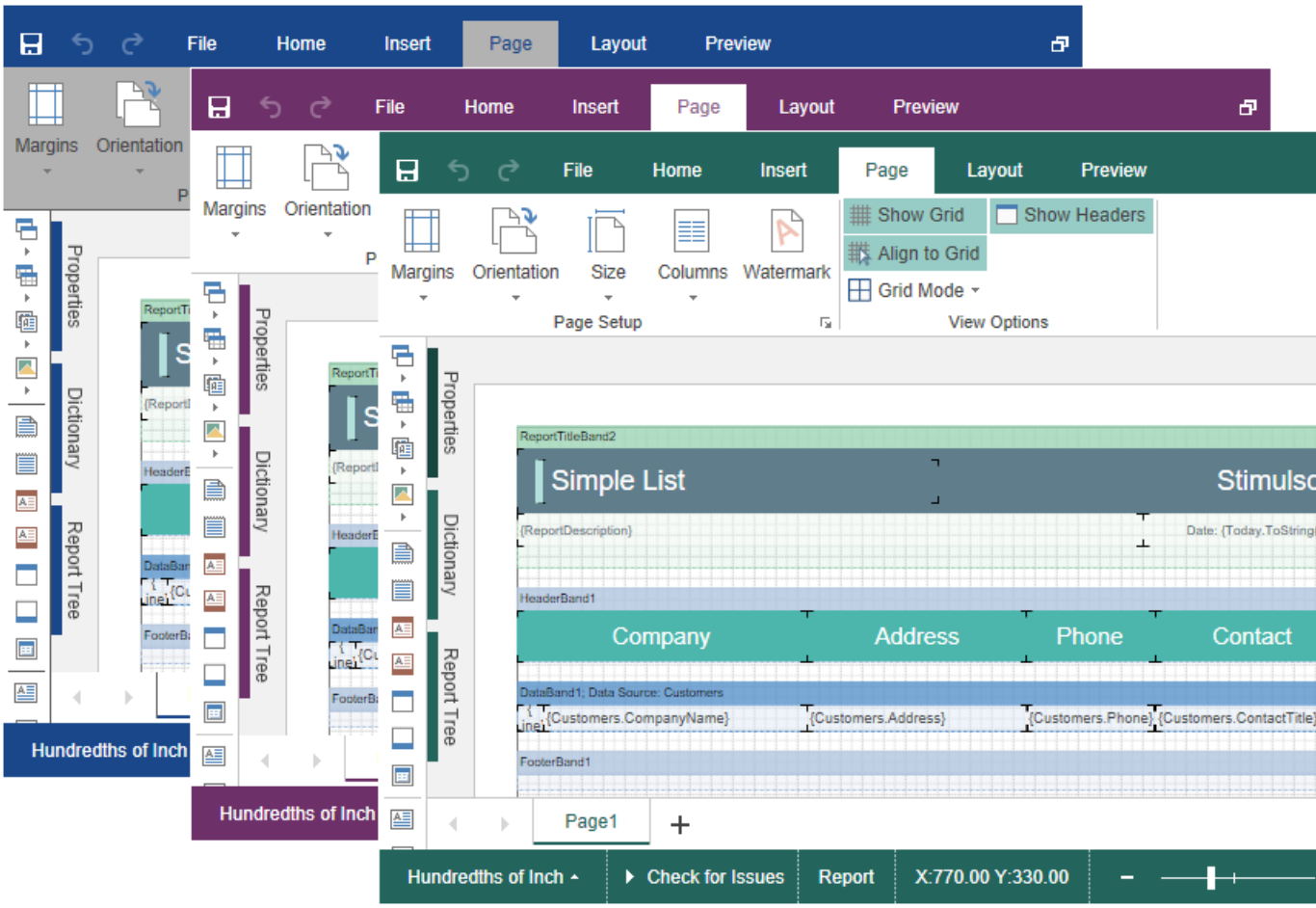
Default.aspx

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Theme = StiDesignerTheme.Office2022WhiteTeal
```

© 2003-2024 Stimulsoft

```
})
...
```

There are currently **2 themes** available with different color accents. As a result, **more than 50** variants of the appearance are available. This allows you to customize the appearance of the designer for almost any design of the Web project.



5.2.10 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

HTM5 Designer uses the server cache to store the editable report template. It is necessary because the client part of the designer contains only a visual representation of components of a report template. The report object itself with all the parameters and properties is stored on the server side.

To use caching, you need to connect modules to work with a session or cache on the server side. To do this, just add the following services to the project in the start file of a project.

Startup.cs

```
...
public void ConfigureServices(IServiceCollection services)
{
   services.AddMemoryCache();
   services.AddSession();
   services.AddMvc();
}
...
```

You can manage caching with the following properties.

The CacheMode property

This property of the designer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** - for caching, the current session is used, in which the report is saved as a packed string.

The CacheItemPriority property

© 2003-2024 Stimulsoft

This property sets the priority of the report stored in the cache of the server. It affects the automatic clearing of the server memory in case of memory shortage. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes you want to store the report in the server cache. If, when using caching, the requested report is not found in the cache (time of storing this report expired), then it will be requested again using the special **GetReport** action. In this case the unsaved changes may be lost.

The **HTML5 Designer** component provides the ability to specify its own methods for working with report caching. For this purpose, a special **StiCacheHelper** class is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods which respectively have the names - **GetObject, SaveObject** and **RemoveObject**..

HomeController.cs

```
...
public class DesignerController : Controller
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override object GetObject(string guid)
      {
         string path =
         System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
         ), guid);
         if (System.IO.File.Exists(path))
         {
            byte[] cacheData = System.IO.File.ReadAllBytes(path);
            return StiCacheHelper.GetObjectFromCacheData(cacheData);
         }
         return null;

         //return base.GetObject(guid);
      }

      public override void SaveObject(object obj, string guid)
      {
         byte[] cacheData = StiCacheHelper.GetCacheDataFromObject(obj);
```

© 2003-2024 Stimulsoft

```
            string path =
            System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
            ), guid);
            System.IO.File.WriteAllBytes(path, cacheData);

            //base.SaveObject(obj, guid);
        }

        public override void RemoveObject(string guid)
        {
            string path =
            System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
            ), guid);
            if (File.Exists(path))
            {
                File.Delete(path);
            }
        }
    }

    static DesignerController()
    {
        StiNetCoreDesigner.CacheHelper = new StiMyCacheHelper();
    }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as the value of the **StiNetCoreDesigner.CacheHelper** static property in the constructor of the controller.

5.2.11 Additional Methods

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

For **HTML5 Designer**, there are several additional methods that are used to get the object of the currently edited report, parameters of the current state of the designer and other useful data. These methods can be used in any actions of the designer.

The GetReportObject() Method

© 2003-2024 Stimulsoft

Returns the report object with which the designer is currently working. It is possible to perform the necessary actions with it - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the designer, specifying it as a parameter in the resulting action method.

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);
   report.ReportName = "MyReportName";

   return StiNetCoreDesigner.ExportReportResult(this, report);
}
...
```

The GetActionReportObject() method

Returns the report object that will be used for the particular action. For example, for the **OpenReport** action, this method returns a report loaded from the local disk of the computer. For the **PreviewReport** action, the method returns a prepared copy of the report for preview.

HomeController.cs

```
...
public IActionResult OpenReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);

   // Register data for the opened report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);
   report.Dictionary.Synchronize();

   return StiNetCoreDesigner.GetReportResult(this, report);
}
...
```

The GetRouteValues() method

Returns values for URLs with which the designer page was opened. Thus, it is

© 2003-2024 Stimulsoft

possible to get the initial collection of page parameters to run the designer and use these values for any checks and conditions.

HomeController.cs

```
...
public IActionResult ExportReport()
{
   RouteValueDictionary routeValues =
   StiNetCoreDesigner.GetRouteValues(this);

   return StiNetCoreDesigner.ExportReportResult(this);
}
...
```

You can also get values of URL parameters by parameter name, specifying it as the parameter of the called action of the designer.

HomeController.cs

```
...
public IActionResult ExportReport(string id)
{
   return StiNetCoreDesigner.ExportReportResult(this);
}
...
```

The GetFormValues() method

Returns the values of the form that initiated (opened by the POST request) a page of the designer. Thus, it is possible to get a collection of form parameters in any action of the designer.

Index.cshtml.cs

```
...
public IActionResult DesignerInteraction()
{
   NameValueCollection formValues = StiNetCoreDesigner.GetFormValues(this);

   return StiNetCoreDesigner.InteractionResult(this);
}
...
```

By default, this feature is disabled to optimize requests of the client-side of the

© 2003-2024 Stimulsoft

designer to the server. To enable it, set the **PassFormValues** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Server =
   {
      PassFormValues = true
   }
})
...
```

The GetRequestParams() method

Returns all parameters of the current state of the designer passed to the server side. They can be useful for determining the type of action that the designer is currently executing - for example, to determine the type of export, as well as all action parameters.

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiRequestParams requestParams =
   StiNetCoreDesigner.GetRequestParams(this);
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiNetCoreDesigner.GetReportObject(this);

      // Some action with report for the PDF export
      // ...

      return StiNetCoreDesigner.ExportReportResult(this, report);
   }

   return StiNetCoreDesigner.ExportReportResult(this);
}
...
```

The GetExportSettings() method

Returns all parameters of the current report export. The type of the parameter object will correspond to the type of export selected in the report preview menu. Any export parameters can be changed and passed to the input of the resulting method.

© 2003-2024 Stimulsoft

In this case, the report will be exported with the parameters transferred.

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiExportSettings settings = StiNetCoreDesigner.GetExportSettings(this);
   if (settings.GetExportFormat() == StiExportFormat.Pdf)
   {
      StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
      pdfSettings.EmbeddedFonts = true;
      pdfSettings.AllowEditable = StiPdfAllowEditable.No;
      return StiNetCoreDesigner.ExportReportResult(this, settings);
   }

   return StiNetCoreDesigner.ExportReportResult(this);
}
...
```

The MapPath() and MapWebRootPath() methods

Returns the absolute path, respectively, to the application or wwwroot directory. You can use this to upload report templates files, data files, etc. These methods are located in the **StiNetCoreHelper** static class.

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));

   return StiNetCoreDesigner.GetReportResult(this, report);
}
...
```

5.2.12 Timeout

When working with the **StiNetCoreDesigner** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last

© 2003-2024 Stimulsoft

action of the viewer. The default setting is 10 minutes.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Server =
   {
      CacheTimeout = 10
   }
})
...
```

Using cache will increase the speed of the report designer. See the chapter Caching for more information

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Server =
   {
      RequestTimeout = 30
   }
})
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection, and data sources in the report.

Index.cshtml

© 2003-2024 Stimulsoft

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      DesignerEvent = "DesignerEvent"
   }
})
...
```

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiNetCoreDesigner.GetReportResult(this, report);
}

public IActionResult DesignerEvent()
{
   return StiNetCoreDesigner.DesignerEventResult(this);
}
...
```

5.2.13 Add custom functions

Information

See on GitHub example of adding a custom function in the ASP.NET MVC HTML5 Designer component.

You can add a custom function to the Dictionary in the report designer when you integrate it into your application. After adding the custom function, you can use this in creating reports and dashboards. Below is the example of adding a function for calculating the sum total.

DesignerController.cs

```
...
public static decimal MySum(object value)
{
   if (!ListExt.IsList(value))
   return Stimulsoft.Base.Helpers.StiValueHelper.TryToDecimal(value);
```

© 2003-2024 Stimulsoft

```
    return Stimulsoft.Data.Functions.Funcs.SkipNulls(ListExt.ToList(value))
    .TryCastToDecimal()
    .Sum();
}
...
static DesignerController()
{
    StiFunctions.AddFunction("MyCategory", "MySum",
    "description", typeof(DesignerController),
    typeof(decimal), "Calculates a sum of the specified set of values.",
    new[] { typeof(object) },
    new[] { "values" },
    new[] { "A set of values" }).UseFullPath = false;
}
...
```

5.2.14 Settings

The **HTML5 Designer** configuration is done using properties that are located in the **StiNetCoreDesignerOptions** class. All properties are divided into groups, some of the groups contain subgroups of properties for ease of use. Below is an example of setting some properties of the designer.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
    Theme = Stimulsoft.Report.Web.StiDesignerTheme.Office2022WhiteTeal,
    Localization = "Localization/en.xml",
    Actions =
    {
        GetReport = "GetReport",
        PreviewReport = "PreviewReport",
        SaveReport = "SaveReport",
        DesignerEvent = "DesignerEvent"
    },
    Appearance =
    {
        InterfaceType = StiInterfaceType.Auto,
        ShowTooltipsHelp = false,
        ShowDialogsHelp = false,
        DefaultUnit = Stimulsoft.Report.StiReportUnitType.Centimeters
    },
    Dictionary =
    {
        PermissionBusinessObjects =
        Stimulsoft.Report.Web.StiDesignerPermissions.None,
        PermissionDataConnections =
        Stimulsoft.Report.Web.StiDesignerPermissions.View
    },
    Bands =
```

© 2003-2024 Stimulsoft

```
    {
        ShowChildBand = false,
        ShowEmptyBand = false,
        ShowOverlayBand = false
    }
})
...
```

Basic settings (without groups)

Name	Description
Theme	Specifies the theme of the report designer. The list of available themes is located in the **StiDesignerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Specifies the path to the XML localization file. The path can be absolute or relative. By default, English localization is used. It is built into the designer and does not require additional XML files.
LocalizationDirectory	Specifies the path to the directory with XML localization files. The localization files located in the specified folder will be loaded to the localization list in the designer panel.
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the component is expanded to the entire area of the browser window.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the component is expanded to the entire area of the browser window.

© 2003-2024 Stimulsoft

Actions

Name	Description
GetReport	Specifies the name of the action method to prepare the report template when loading the designer.
OpenReport	Specifies the name of the action method to open the report template from the designer menu.
CreateReport	Specifies the name of the action method to prepare the report template when creating the new report in the designer.
SaveReport	Specifies the name of the action method to save the report template on the server side..
SaveReportAs	Specifies the name of the action method to store the report template on the server side when using the **Save As** menu item. If no action is specified, the built-in method of saving the report template to the local disk will be used.
PreviewReport	Specifies the name of the action method to prepare the rendered report in the preview window.
GetPreviewReport	Specifies the name of the action method just before a report is displayed in the preview window.
ExportReport	Specifies the name of the action method to export reports to the specified format.
Exit	Specifies the name of the action method to go to the desired view by clicking the Exit button in the main menu of the report designer.
DesignerEvent	Specifies the name of the action method of the report designer to handle additional designer actions, such as working with data, report

© 2003-2024 Stimulsoft

	components, and others. Also, this action is used to load scripts and designer styles.

Server

Name	Description
Controller	Sets the name of the controller to process requests. If this property is not set, then the current controller will be used to process the requests.
RouteTemplate	Sets the route template that is returned when the report designer actions are executed. If the property is not set, then the MVC project template will be used instead. The default value of the property is null.
ShowServerErrorPage	Enables the display of an HTML page with error details that occurred on the server side. The error details will be displayed in the preview window if the property is enabled. If the property is disabled, the numeric error code and a short error description will be displayed in the dialog window. By default, the property is set to **true**.
AllowAutoUpdateCookies	Allows the designer to update the cookies automatically on every request to the server. By default, cookies are set when creating the designer, if they are not specified in the report. By default, the property is set to **false**.
AllowAntiforgeryToken	Allow the designer to automatically request and send the antiforgery token. By default, the property is set to **true**.
RequestTimeout	Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

© 2003-2024 Stimulsoft

CacheTimeout	Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled in **HTML5 Designer**; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as a repository, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or server session will be automatically re-saved after a certain period of time if the designer is idle (about every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the designer mode in which relative URLs are used for requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value which specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-**

© 2003-2024 Stimulsoft

	1 removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the designer. If **false**, only the necessary parameters are used to request the resources of the designer. This corresponds to the more correct operation of the browser cache. By default, the property is set to **true**.
PassFormValues	Enables transferring the POST form values to the client side, if these values are to be used in the actions of the designer. If you enable this feature, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
UseCompression	Enables compression of designer requests in the GZip stream. This allows you to reduce the amount of Internet traffic, but slows down the designer. By default, the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server side. The following resources are supported: scripts, styles and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Specifies the path to the CSS file of styles for the report designer. If this property is set, the standard styles of the selected theme will not be

© 2003-2024 Stimulsoft

	loaded. The default value is an empty value.
DefaultUnit	Sets the units for the size of the report and all its components. By default, centimeters are used.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the **StiZoomMode** enumeration: › **PageWidth** – when the designer runs, the zoom, necessary to display the report by the page width, will be set; › **PageHeight** – when the designer runs, the zoom, required to display the page height of the report, will be set.
ShowAnimation	Enables animation for various elements of the designer interface. By default, the property is set to **true**.
ShowOpenDialog	Allows to display the open dialog, or to open with the open event. By default, the property is set to **true**.
ShowTooltips	Enables displaying tooltips for designer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enable displaying links to online documentation in tooltips for designer controls. By default, the property is set to **true**.
ShowDialogsHelp	Enables displaying links to online documentation on the titles of dialog forms of the designer. By default, the property is set to **true**.
InterfaceType	Sets the type of interface used for the designer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the interface type of the designer will

© 2003-2024 Stimulsoft

	be selected automatically depending on the device used (default value); › **Mouse** – forced use of the interface to control the designer with the mouse; › **Touch** – forced use of the Touch interface to control the designer via the touch screen (mobile devices), also in this mode, the interface elements are increased.
DatePickerFirstDayOfWeek	Sets the first day of the week for the select date item. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Auto** – automatic detection of the first day of the week from the browser settings (default value); › **Monday** – the first day of the week is Monday; › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
ShowReportTree	Enables displaying the tree of report components. By default, the property is set to **true**.
ChartRenderType	Gets or sets the type of the chart in the preview. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages in the preview tab. Can take one of the

© 2003-2024 Stimulsoft

	following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
CloseDesignerWithoutAsking	Sets a value which indicates that the designer will be closed without asking. By default, the property is set to **true**.
ShowSystemFonts	Sets a visibility of the system fonts in the fonts list. By default, the property is set to **true**.
WizardTypeRunningAfterLoad	Calls the Report wizard after starting the report designer. It may have one of the following **StiWizardType** enumeration values: › **None** - runs the report designer without running the report wizard; › **StandardReport** - runs the Standard wizard; › **MasterDetailReport** - runs the Master-Detail wizard; › **LabelReport** - runs the Label wizard; › **InvoicesReport** - runs the Invoice wizard; › **OrdersReport** - runs the Order wizard; › **QuotationReport** - runs the Quote wizard.
AllowWordWrapTextEditors	Allows word wrap in the text editors. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

Behavior

Name	Description
ShowSaveDialog	Enables displaying the dialog to insert a report name when it is saved. The name of the report will be transferred in the parameters of the report designer. By default, the property is set to **true**.
UndoMaxLevel	Sets the maximum number to cancel actions with the report (the Undo/Redo function). A big value of this property will consume memory on the server side to store the undo parameters. The default value is **6**.
AllowChangeWindowTitle	Allows using the title of the browser window to display the file name of the edited report. By default, the property is set to **true**.
SaveReportMode	Sets the mode to save the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
SaveReportAsMode	Sets the mode for saving the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the

© 2003-2024 Stimulsoft

	current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
CheckReportBeforePreview	Sets the value that allows running the report checker before preview.

FileMenu

Name	Description
Visible	Enables displaying the main menu of the report designer. By default, the property is set to **true**.
ShowNew	Enables showing the main menu item - **New**. By default, the property is set to **true**.
ShowFileMenuNewReport	Sets a visibility of the new report button in the designer. By default, the property is set to **true**.
ShowFileMenuNewDashboard	Sets a visibility of the new dashboard button in the designer. By default, the property is set to **true**.
ShowOpen	Enables showing the main menu item - **Open**. By default, the property is set to **true**.
ShowSave	Enables showing the main menu item - **Save**. By default, the property is set to **true**.
ShowSaveAs	Enables showing the main menu item - **Save As**. By default, the property is set to **true**.
ShowClose	Enables showing the main menu item - **Close**. By default, the property is set to **true**.
ShowExit	Enables showing the main menu item - **Exit**. By default, the property is set to **true**.
ShowReportSetup	Enables showing the main menu item - **Report Setup**. By default, the property is set to **true**.
ShowOptions	Enables showing the main menu item - **Options**. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowInfo	Enables showing the main menu item - **Info**. By default, the property is set to **true**.
ShowAbout	Enables showing the main menu item - **About**. By default, the property is set to **true**.
ShowHelp	Enables showing the main menu item - **Help**. By default, the property is set to **true**.

Dictionary

Name	Description
Visible	Enables showing the data dictionary of the report. By default, the property is set to **true**.
UseAliases	Allows you to use aliases in the data dictionary. It has the three values of the **StiUseAliases** enumeration: › **Auto** - defines the mode of using aliases from a saved value in cookies (default value); › **True** - sets the mode of using aliases in the data dictionary; › **False** - disables the mode of using aliases in the data dictionary.
NewReportDictionary	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It has the three values of the **StiNewReportDictionary** enumeration: › **Auto** - defines the mode to create or join the data dictionary from a saved value in cookies (default value); › **DictionaryNew** - sets the mode to create a new data dictionary when creating a new report; › **DictionaryMerge** - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
ShowDictionaryContextMenuPro	Sets a visibility of the **Properties** item in the

© 2003-2024 Stimulsoft

perties	dictionary context menu. By default, the property is set to **true**.
ShowDictionaryActions	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.
PermissionDataConnections	Sets the available actions to connect data to the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataSources	Sets available actions on report data sources. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataColumns	Sets the available actions on data columns in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionBusinessObjects	Sets the available actions on the business objects in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataRelations	Sets the available actions to linking data in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionVariables	Sets available actions on report variables. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionResources	Sets the available actions for the resources in the Report Dictionary. Takes one or several values from the **StiDesignerPermissions** enumeration.
PermissionSqlParameters	Sets the available actions for the parameters of the SQL queries for the Report DataSources. Takes one or several values from **StiDesignerPermissions** enumeration.
DataTransformationsPermissions	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

© 2003-2024 Stimulsoft

The table below shows all available values for the **StiDesignerPermissions** enumeration, which can be set for the dictionary elements of the report.

Value	Description
None	Disables any action on the item of the data dictionary.
All	Allows any action on the item of the data dictionary.
Create	Allows creating a specific data dictionary item.
Delete	Allows deleting a specific data dictionary item.
Modify	Allows modifying a specific data dictionary item.
View	Allows viewing a specific data dictionary item.
ModifyView	Allows modifying and viewing a specific data dictionary item.

Toolbar

Name	Description
ShowToolbar	Enables displaying the toolbar of the designer. By default, the property is set to **true**.
ShowSetupToolboxButton	Enables displaying the button to call the dialog box with settings for the side toolbar. By default, the property is set to **true**.
ShowInsertButton	Enables displaying the **Insert** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowLayoutButton	Enables displaying the tab **Layout** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPageButton	Enables displaying the tab **Page** tab on the toolbar of the designer. By default, the property

© 2003-2024 Stimulsoft

	is set to **true**.
ShowPreviewButton	Enables displaying the tab **Preview** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the designer. By default, the property is set to **true**.
ShowAboutButton	Enables displaying the **About** on the toolbar of the designer. By default, the property is set to **false**.

PropertiesGrid

Name	Description
Visible	Enables displaying the property panel. By default, the property is set to **true**.
Width	Sets the width of the property panel. By default, the width is set to **370 px**.
LabelWidth	Specifies the width of the labels on the properties panel. By default, the width is set to **160 px**.
PropertiesGridPosition	Sets **Left** or **Right** position of the properties grid in the designer. It has the three values of the **StiPropertiesGridPosition** enumeration: › **Left**; › **Right**.
ShowPropertiesWhichUsedFromStyles	Sets a visibility of the properties which used from styles in the designer. By default, the property is set to **false**.

Components

Name	Description

© 2003-2024 Stimulsoft

ShowText	Enables displaying the **Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowTextInCells	Enables displaying the **Text in Cells** component in the insert menu for report components. By default, the property is set to **true**.
ShowRichText	Enables displaying the **Rich Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowImage	Enables displaying the **Image** component in the insert menu for report components. By default, the property is set to **true**.
ShowBarCode	Enables displaying the **Bar Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowShape	Enables displaying the **Shape** component in the insert menu for report components. By default, the property is set to **true**.
ShowHorizontalLinePrimitive	Enables displaying the **Horizontal Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowVerticalLinePrimitive	Enables displaying the **Vertical Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowRectanglePrimitive	Enables displaying the **Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowRoundedRectanglePrimitive	Enables displaying the **Rounded Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowPanel	Enables displaying the **Panel** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowClone	Enables displaying the **Clone** component in the insert menu for report components. By default, the property is set to **true**.
ShowCheckBox	Enables displaying the **Check Box** component in the insert menu for report components. By default, the property is set to **true**.
ShowSubReport	Enables displaying the **Sub Report** component in the insert menu for report components. By default, the property is set to **true**.
ShowZipCode	Enables displaying the **Zip Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowTable	Enables displaying the **Table** component in the insert menu for report components. By default, the property is set to **true**.
ShowCrossTab	Enables displaying the **Cross-Tab** component in the insert menu for report components. By default, the property is set to **true**.
ShowChart	Enables displaying the **Chart** component in the insert menu for report components. It affects on all chart types. By default, the property is set to **true**.
ShowMap	Enables displaying the **Map** component in the insert menu for report components. By default, the property is set to **false**.
ShowGauge	Enables displaying the **Gauge** component in the insert menu for report components. By default, the property is set to **false**.
ShowSparkline	Enables displaying the **Sparkline** component in the insert menu for report components. By default, the property is set to **true**.
ShowMathFormula	Enables displaying the **Math Formula** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowElectronicSignature	Enables displaying the **Electronic Signature** component in the insert menu for report components. By default, the property is set to **true**.
ShowPdfDigitalSignature	Enables displaying the **PDF Digital Signature** component in the insert menu for report components. By default, the property is set to **true**.

Bands

Name	Description
ShowReportTitleBands	Enables displaying the **Report Title** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowReportSummaryBand	Enables displaying the **Report Summary** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageHeaderBand	Enables displaying the **Page Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageFooterBand	Enables displaying the **Page Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupHeaderBand	Enables displaying the **Group Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupFooterBand	Enables displaying the **Group Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHeaderBand	Enables displaying the **Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowFooterBand	Enables displaying the **Footer** item in the **Bands**

© 2003-2024 Stimulsoft

	menu of the designer. By default, the property is set to **true**.
ShowColumnHeaderBand	Enables displaying the **Column Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnFooterBand	Enables displaying the **Column Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowDataBand	Enables displaying the **Data** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHierarchicalBand	Enables displaying the **Hierarchical** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowChildBand	Enables displaying the **Child** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowEmptyBand	Enables displaying the **Empty** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowOverlayBand	Enables displaying the **Overlay** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowTableOfContents	Enables displaying the **Table of Contents** item in the **Bands** menu of the designer. By default, the property is set to **true**.

DashboardElements

Name	Description
ShowTableElement	Enables displaying the **Table** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowCardsElement	Enables displaying the **Cards** element in the

© 2003-2024 Stimulsoft

	Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowChartElement	Enables displaying the **Chart** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowGaugeElement	Enables displaying the **Gauge** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPivotTableElement	Enables displaying the **Pivot** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowIndicatorElement	Enables displaying the **Indicator** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowProgressElement	Enables displaying the **Progress** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowRegionMapElement	Enables displaying the **Region Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowOnlineMapElement	Enables displaying the **Online Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowImageElement	Enables displaying the **Image** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTextElement	Enables displaying the **Text** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPanelElement	Enables displaying the **Panel** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowShapeElement	Enables displaying the **Shape** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowButtonElement	Enables displaying the **Button** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowListBoxElement	Enables displaying the **List Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowComboBoxElement	Enables displaying the **Combo Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewElement	Enables displaying the **Tree View** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewBoxElement	Enables displaying the **Tree View Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowDatePickerElement	Enables displaying the **Date Picker** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.

CrossBands

Name	Description
ShowCrossGroupHeaderBand	Enables displaying the **Cross Group Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossGroupFooterBand	Enables displaying the **Cross Group Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossHeaderBand	Enables displaying the **Cross Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossFooterBand	Enables displaying the **Cross Footer** section in

© 2003-2024 Stimulsoft

	the insert menu for report components. By default, the property is set to **true**.
ShowCrossDataBand	Enables displaying the **Cross Data** section in the insert menu for report components. By default, the property is set to **true**.

Dashboards

Name	Description
ShowNewDashboardButton	Sets a visibility of the **New Dashboard** button in the designer. By default, the property is set to **true**.

Pages

Name	Description
ShowNewPageButton	Sets a visibility of the **New Page** button in the designer. By default, the property is set to **true**.

When designing a report or dashboard in the report designer, you can also define **ExportOptions**, **EmailOptions**, and **PreviewToolbarOptions** on the **Preview** tab. These options are similar to the report viewer options.

6 Reports and Dashboards for ASP.NET Core Razor

.NET Core is a cross-platform technology for creating Web applications for Windows, Linux, and macOS. **ASP.NET Core Razor Pages** is a technology that is similar to the Model-View-Controller (MVC) system. **Razor Pages** allow you to create pages with code that can process various requests. This technology makes the creation of scripts for pages easier and more efficient in comparison with the use of the MVC solution. Stimulsoft company provides tools for creating, displaying, and transforming reports and dashboards using this technology.

© 2003-2024 Stimulsoft

Tools for creating and editing reports:	Tools for viewing and converting reports:	Tools for creating and editing dashboards:	Tools for viewing and converting dashboards:
HTML5 designer	HTML5 viewer	HTML5 designer	HTML5 viewer

6.1 HTML5 Viewer

Samples

See on GitHub examples for working with the ASP.NET Core Razor HTML5 Viewer component. All examples are separate projects, grouped into one solution for Visual Studio.

The **HTML5 Viewer** (**StiNetCoreViewer**) component is designed for viewing reports in the web browser. You do not need to install the .NET Framework, ActiveX components, or any special plug-ins on the client-side. All that is needed is any modern Web browser.

With the help of **HTML5 Viewer**, you can view, print, and export reports on any computer with any operating system installed. Since the viewer only uses HTML and JavaScript technologies, it can be run on devices with no Flash or Silverlight support - tablets, smartphones. Also, the viewer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Viewer** component uses the AJAX technology to perform all actions (uploading a report, paging, scaling, interactivity in reports, etc.), which allows you to get rid of reloading the entire page save Web traffic, and speed up work. The report engine built using the .NET Core technology is used to render reports. This is a cross-platform technology. It allows you to deploy the application on servers that use operating systems like Windows, macOS, and Linux.

The **HTML5 Viewer** supports many themes, animated interface, bookmarks,

© 2003-2024 Stimulsoft

interactive reports, editing of report elements on the page, full-screen mode, search panel, and other necessary features for viewing reports.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Viewer** in a Web project, you need to install the NuGet package of Stimulsoft.Reports.Web.NetCore:
› Select "Manage NuGet Packages ..." in the context menu of the project;
› Specify Stimulsoft.Reports.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

To add the ability to view and export dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web.NetCore (this package is associated with the package Stimulsoft.Reports.Web.NetCore. If it is missed, it will be installed automatically):
› Select "Manage NuGet Packages ..." in the context menu of the project;
› Specify Stimulsoft.Dashboards.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

- How this works?
- Activation
- Showing Reports and Dashboards
- Connecting data
- Localization
- Printing Reports
- Exporting Reports and Dashboards
- Interactive Reports
- Timeout
- Editing Rendered Reports
- Sending Reports by Email
- Calling Designer from Viewer
- Caching
- Using Themes

© 2003-2024 Stimulsoft

- Viewing Modes
- Basic Features
- Work with Parameters
- Additional Methods
- Work with Bookmarks
- Export and Printing from Code
- Viewer Settings

6.1.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word "report" will be used in the documentation text.

To run the viewer, you should place the **StiNetCoreViewer** component on a page, set the necessary properties, and define necessary actions in the page event handler. When running the report viewer, the following actions occur:

- The .NET Core component generates HTML and JavaScript code that is necessary for displaying and running the viewer;
- When the component is output, the JavaScript method is launched. It requests the first page of the report on the server-side or the entire report (depending on the selected mode) and the required report parameters;
- Each action in the viewer (for example, paging, printing or exporting a report, etc.) calls a certain action on the server-side, in which you can perform the necessary manipulations with the report;
- The viewer saves the report in cache or server session to speed up the work, which makes it impossible to re-build the report.

6.1.2 Activation

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiNetCoreViewer** component.

Index.cshtml.cs

```
...
```

© 2003-2024 Stimulsoft

```
//Activation with using license code
public class IndexModel : PageModel
{
   static IndexModel()
   {
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";
   }
}

//Activation with using license file
public class IndexModel : PageModel
{
   public IndexModel(IWebHostEnvironment webHostEnvironment)
   {
      var path = Path.Combine(webHostEnvironment.ContentRootPath, "Content\
      \license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

6.1.3 Showing Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

© 2003-2024 Stimulsoft

To display a report, you should add the **StiNetCoreViewer** component to a page, set the minimum required properties, and define necessary actions in the page event handler.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent"
   }
})
...
```

Index.cshtml.cs

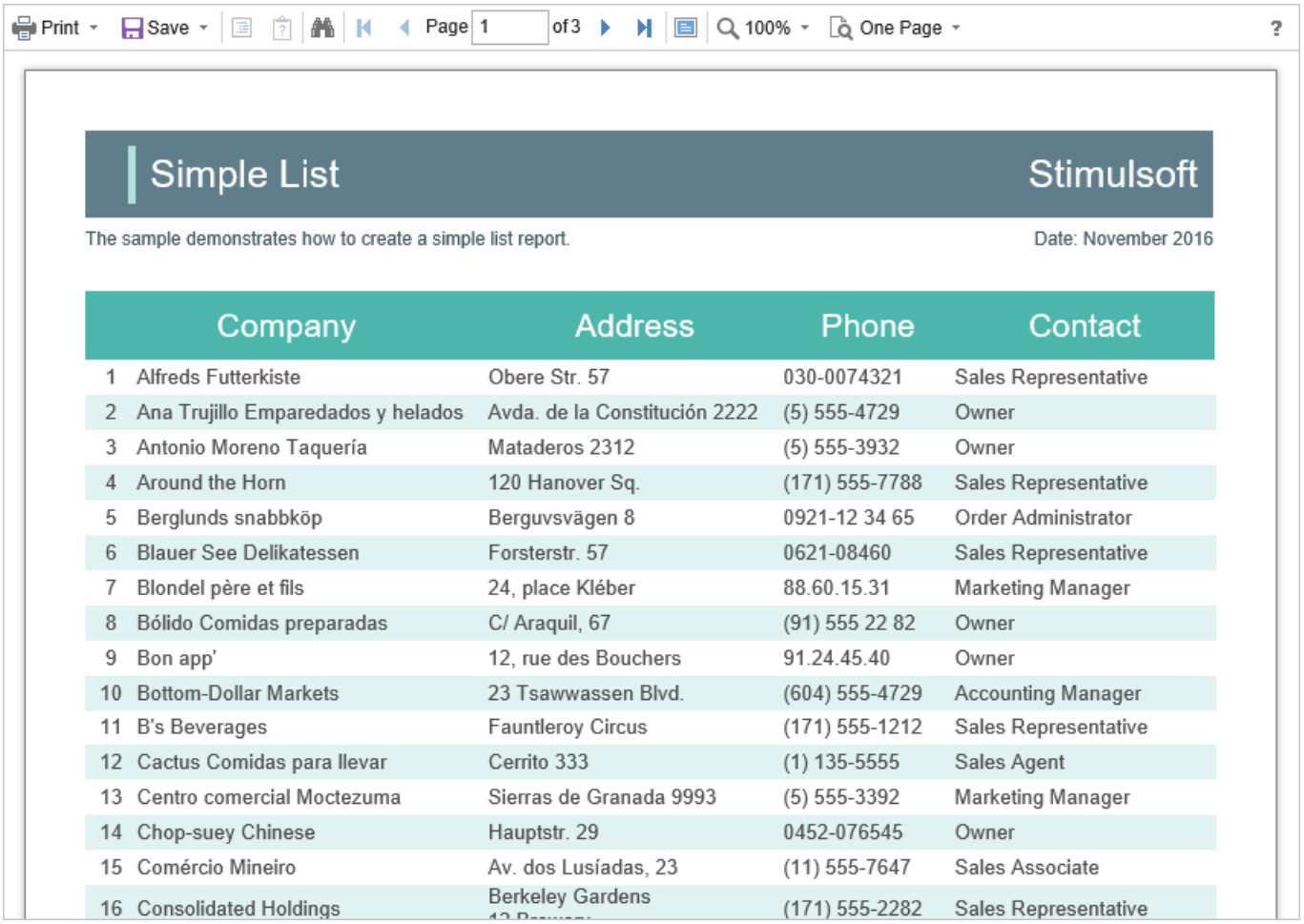
```
...
public IActionResult OnPostGetReport()
{
   // Create the report object
   StiReport report = new StiReport();

   // Load report or dashboard
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
   //report.Load(StiNetCoreHelper.MapPath(this, "Reports/Dashboard.mrt"));

   return StiNetCoreViewer.GetReportResult(this, report);
}

public IActionResult OnGetViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}

public IActionResult OnPostViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}
...
```

In the above example, the processing of two actions of the viewer is added. The **GetReport** action returns the report prepared for preview. The **ViewerEvent** action handles the viewer events.

Information

© 2003-2024 Stimulsoft

The **ViewerEvent** action is mandatory. Without it, the viewer can't work correctly. The action is called for two types of requests: **OnGet** – the component requests the resources necessary for work, such as CSS styles, JS scripts and images; **OnPost** – all other actions of the viewer.



If the report was not rendered before showing, the **HTML5 Viewer** component will automatically render it. So you can use report templates and rendered reports to display reports.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
```

© 2003-2024 Stimulsoft

```
    StiReport report = new StiReport();
    report.LoadDocument(StiNetCoreHelper.MapPath(this, "Reports/
    SimpleList.mdc"));

    return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Since the dashboard is not a static document and requires data to work, the format of the rendered MDC document is not available for it. Instead, it is possible to use a snapshot of the report in the MRT format, which contains all the data necessary for the dashboard to work and be correctly displayed in the viewer.

Index.cshtml.cs

```
...
public ActionResult OnPostGetReport()
{
    StiReport report = new StiReport();
    report.Load(StiNetCoreHelper.MapPath("~/Reports/Snapshot.mrt"));

    return StiNetCoreViewer.GetReportResult(report);
}
...
```

Loading custom fonts

You can connect custom fonts using the **StiFontCollection** class, having specified a file that contains a font. To do this, you should invoke the static method in the page constructor to load the font.

Index.cshtml.cs

```
...
public class ViewerController : Controller
{
    static ViewerController()
    {
        Stimulsoft.Base.StiFontCollection.AddFontFile(StiNetCoreHelper.MapPat
        h(this, "/fonts/my-font/font-name.ttf"));
    }
}
...
```

6.1.4 Connecting Data

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Data to a report can be connected in various ways. The easiest way is to store connection settings in the report template. You can also connect the data from the code. This can be done when the report is loaded in the **GetReport** action.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   DataSet ds = new DataSet();
   ds.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/
   TwoSimpleLists.mrt"));
   report.Dictionary.Databases.Clear();
   report.RegData("Demo", ds);

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Data for the report can be connected not only when the report is loaded. For example, you can connect new data at the moment of interactive actions in the viewer (applying report parameters, sorting, drill-down, collapsing). To do this, you should set the **Interaction** action for the **HTML5 Viewer** component and, in the action handler, connect the data for the current report. In the same way, you can connect data in other actions of the viewer.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent",
      Interaction = "ViewerInteraction"
   }
```

© 2003-2024 Stimulsoft

```
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   DataSet data = new DataSet();
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = StiNetCoreViewer.GetReportObject(this);
   report.RegData("Demo", data);

   return StiNetCoreViewer.InteractionResult(this, report);
}
...
```

If you want to connect new data only for a certain interactive action of the viewer, for example, only when you apply report parameters you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side on any request and contain all necessary information and states of the client part of the viewer. To determine the type of action of the viewer, it is enough to check the Action property of the viewer parameters.

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      DataSet data = new DataSet();
      data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

      StiReport report = StiNetCoreViewer.GetReportObject(this);
      report.RegData("Demo", data);

      return StiNetCoreViewer.InteractionResult(this, report);
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

© 2003-2024 Stimulsoft

SQL data sources

The connection parameters to the SQL data source and any other ones can be stored in the report template. Suppose you want to set the connection parameters from the code before rendering the report (for example, for security reasons or depending on the authorized user). In that case, you can use the example below.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   OracleConnection connection = new OracleConnection("Data
   Source=Oracle8i;Integrated Security=yes");
   connection.Open();
   OracleDataAdapter adapter = new OracleDataAdapter();
   adapter.SelectCommand = new OracleCommand("SELECT * FROM Products",
   connection);

   DataSet dataSet = new DataSet("productsDataSet");
   adapter.Fill(dataSet, "Products");

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/
   SqlSampleReport.mrt"));
   report.RegData("Products", dataSet);

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Also, for SQL data sources used in the report, you can specify the Query Timeout in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection and data sources in the report.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath("Report.mrt"));
   ((StiSqlDatabase)
```

© 2003-2024 Stimulsoft

```
    report.Dictionary.Databases["Connection"]).ConnectionString = @"Data
    Source=server;Integrated Security=True;Initial Catalog=DataBase";
    ((StiSqlSource)
    report.Dictionary.DataSources["DataSourceName"]).SqlCommand = "select *
    from Table where Column = 100";
    ((StiSqlSource)
    report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

    return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Information

For SQL data sources of other types, the connection is created similarly, and an adapter corresponding to the data source type is connected. For example, for the MS SQL data source, you should connect SqlDataAdapter. For OracleDataAdapter is required for Oracle. Also, you should specify a connection string that matches the connection type.

You can also use data for designing reports and dashboards obtained from OData storage. In this case, you can do the authorization using a username, user password, or using a token. Authorization parameters are specified in the connection string to the OData storage using the ";" separator.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{

    var report = new StiReport();

    //Authorization using a user account
    var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
    services.odata.org/V4/Northwind/
    Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Password;C
    lient_Id=Your Client ID", false, null);

    //Authorization using a user token
    var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
    services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your token",
    false, null);

    report.Dictionary.Databases.Add(oDataDatabase);
    oDataDatabase.Synchronize(report);
```

© 2003-2024 Stimulsoft

```
    //Query with data filter
    ((StiSqlSource)report.Dictionary.DataSources["Products"]).SqlCommand =
    "Products?$filter=ProductID eq 2";

    return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

The table below shows the connection string templates for different types of data sources.

Data Source	Connection String Template
MS SQL	Integrated Security=False; Data Source=myServerAddress;Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
MySQL	Server=myServerAddress; Database=myDataBase;UserId=myUsername; Pwd=myPassword;
ODBC	Driver={SQL Server}; Server=myServerAddress;Database=myDataBase ; Uid=myUsername; Pwd=myPassword;
OLE DB	Provider=SQLOLEDB.1; Integrated Security=SSPI;Persist Security Info=False; Initial Catalog=myDataBase;Data Source=myServerAddress
Oracle	Data Source=TORCL;User Id=myUsername;Password=myPassword;
MS Access	Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Password=pass;Data Source=C:\ \myAccessFile.accdb;
PostgreSQL	Server=myServerAddress; Port=5432; Database=myDataBase;User Id=myUsername; Password=myPassword;
Firebird	User=SYSDBA; Password=masterkey; Database=SampleDatabase.fdb;DataSource=my

© 2003-2024 Stimulsoft

	ServerAddress; Port=3050; Dialect=3; Charset=NONE;Role=; Connection lifetime=15; Pooling=true; MinPoolSize=0;MaxPoolSize=50; Packet Size=8192; ServerType=0;
SQL CE	Data Source=c:\MyData.sdf; Persist Security Info=False;
SQLite	Data Source=c:\mydb.db; Version=3;
DB2	Server=myAddress:myPortNumber;Database=myDataBase;UID=myUsername;PWD=myPassword; Max Pool Size=100;Min Pool Size=10;
Infomix	Database=myDataBase;Host=192.168.10.10;Server=db_engine_tcp;Service=1492;Protocol=onsoctcp;UID=myUsername;Password=myPassword;
Sybase	Data Source=myASEserver;Port=5000;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
Teradata	Data Source=myServerAddress;User ID=myUsername;Password=myPassword;
VistaDB	Data Source=D:\folder \myVistaDatabaseFile.vdb4;Open Mode=ExclusiveReadWrite;
Universal(dotConnect)	Provider=Oracle;direct=true;data source=192.168.0.1;port=1521;sid=sid;user=user; password=pass
MongoDB	mongodb://<user>:<password>@localhost/test
OData	http://services.odata.org/v3/odata/OData.svc/
Other...	The table shows the most commonly used templates for the connection string. You can view various connection string options at the special website.

Data from XML, JSON, Excel files

Connecting to XML and JSON data sources can be stored in the report template. If

© 2003-2024 Stimulsoft

you want to specify data files from the code, you can use the example below.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   DataSet data = new DataSet();
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
   report.RegData(data);

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   DataSet data
   = StiJsonToDataSetConverterV2.GetDataSetFromFile(StiNetCoreHelper.MapPat
   h(this, "Data/Demo.json"));

   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
   report.RegData(data);

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

Information

The viewer has the possibility of obtaining data from an Excel file. To do this, you can use the following method.

```
DataSet dataSet = StiExcelConnector.Get().GetDataSet(new StiExcelOptions(ar
```

6.1.5 Localization

The **HTML5 Viewer** component supports the complete localization of its interface. To localize the report viewer interface, use the special **Localization** property. The value of this property should specify the path to the localization XML file (relative or absolute).

© 2003-2024 Stimulsoft

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Localization = "Localization/en.xml"
})
...
```

When you load the report viewer, the localization file will be loaded automatically.

6.1.6 Using Themes

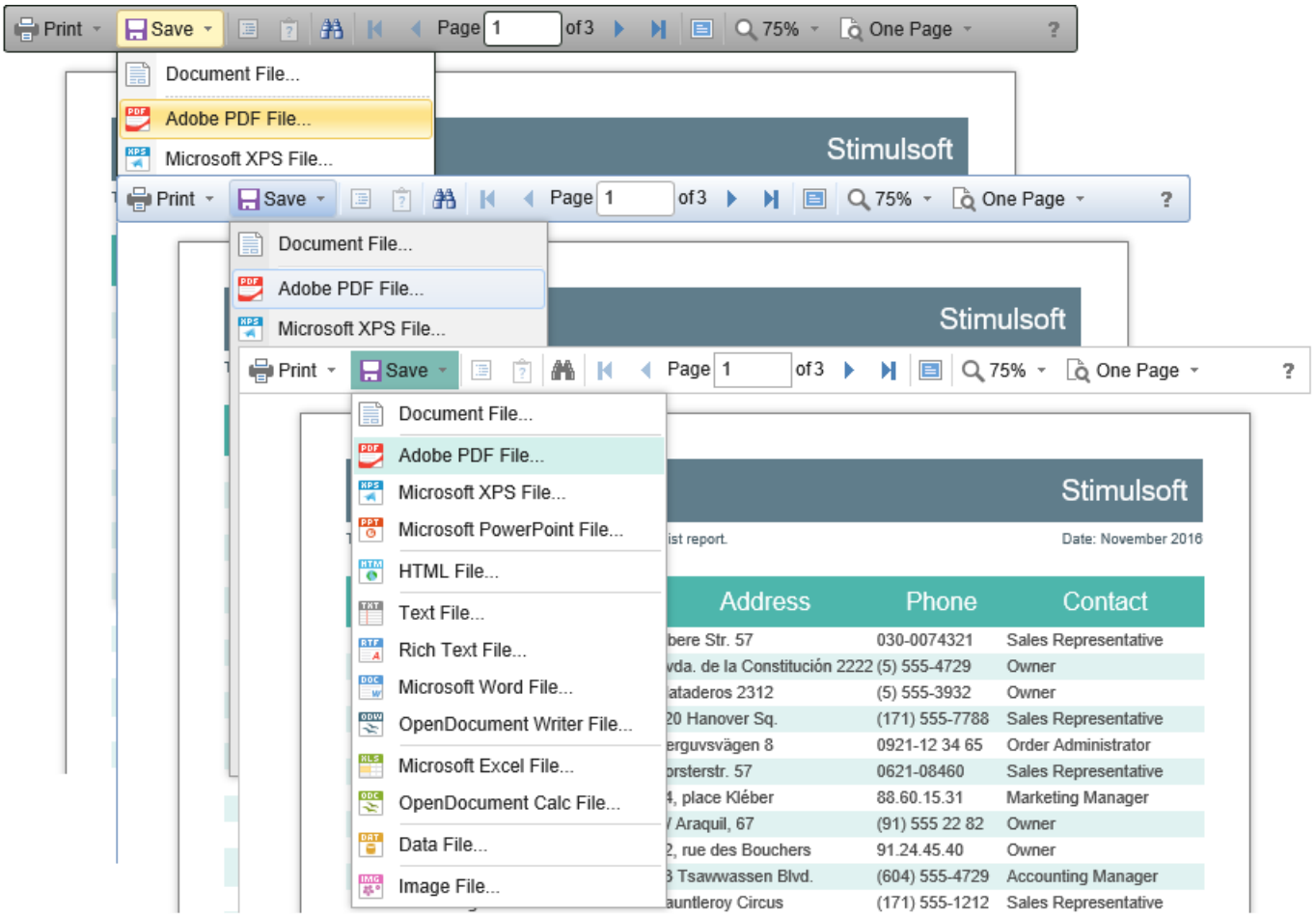
The **HTML5 Viewer** component can change the appearance of visual controls. To change the theme, use the **Theme** property, which can take one of the values of the **StiViewerTheme** enumeration.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Theme = StiViewerTheme.Office2022WhiteTeal
})
...
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.

© 2003-2024 Stimulsoft



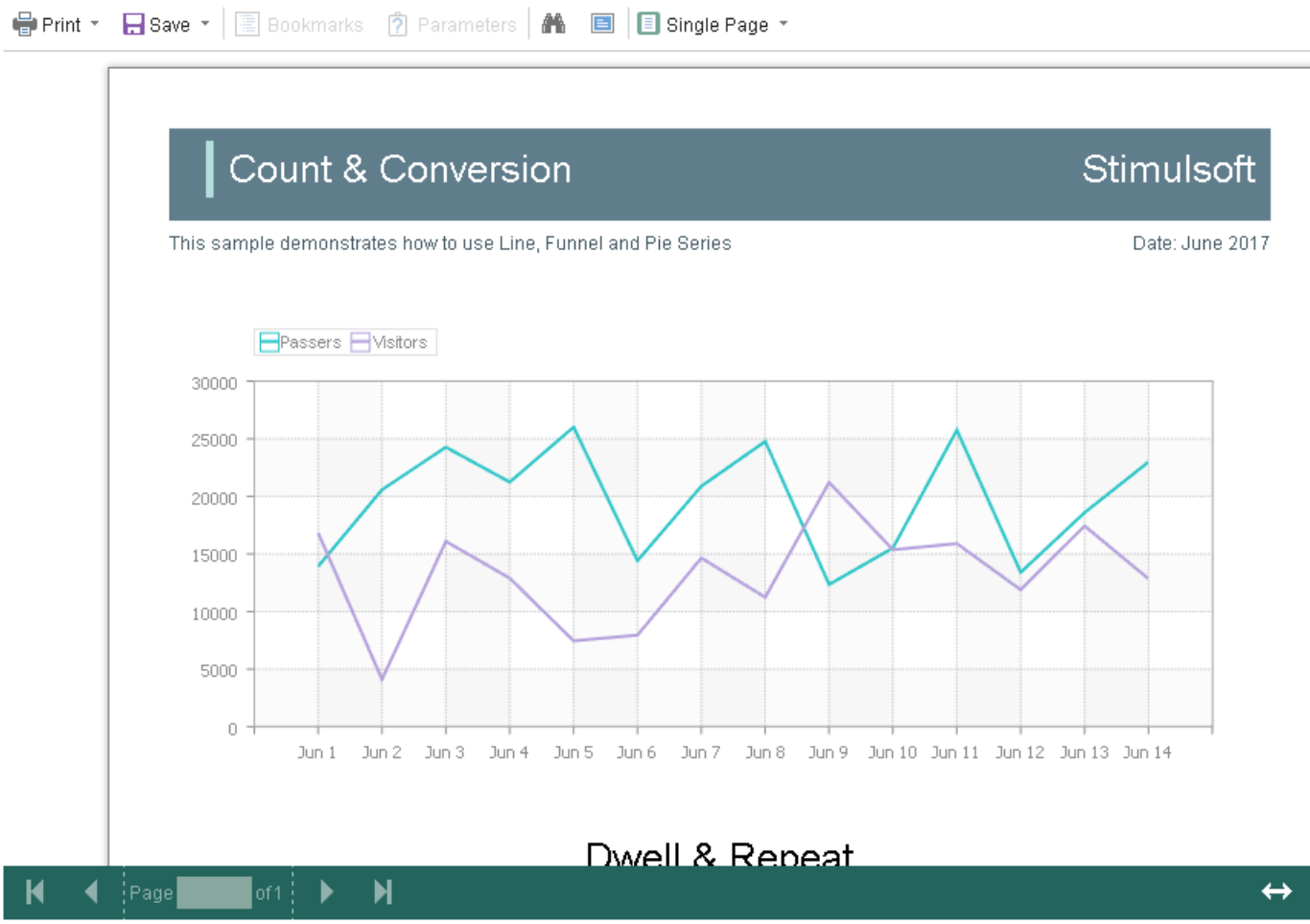
By default, the viewer has only the top toolbar on which all the report controls are located. If necessary, the toolbar can be split into top and bottom parts. The top panel will contain the menu for printing and exporting the report and the buttons for working with parameters and bookmarks. The bottom panel will contain controls to switch between the report pages and setting the zoom of pages. To enable this mode, enable the **ToolbarDisplayMode** property. It has values **Simple** and **Separated**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      ScrollbarsMode = true
   },
   Toolbar =
   {
```

© 2003-2024 Stimulsoft

```
        DisplayMode = StiToolbarDisplayMode.Separated
    }
})
...
```



In addition, it is possible to set the appearance parameters for the main elements of the viewer. For example, you can change the font and color of the control panel inscriptions of the viewer, set the background of the viewer, set the color of page borders, etc. Below is a list of available properties that change the appearance of the viewer, and their default values.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      BackgroundColor = Color.White,
```

© 2003-2024 Stimulsoft

```
      PageBorderColor = Color.Blue,
      ShowPageShadow = true
   },
   Toolbar =
   {
      BackgroundColor = Color.White,
      BorderColor = Color.Gray,
      FontColor = Color.Black,
      FontFamily = "Arial"
   }
})
...
```

6.1.7 Basic Features

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The main features of the viewer include the following operations:

- Displaying the report.
- Switching between the report pages.
- Changing the scale.
- Displaying the preview mode.

All specified operations are performed in the AJAX mode without restarting the browser page. For the correct work of these operations, you should define a special ViewerEvent action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      ViewerEvent = "ViewerEvent"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnGetViewerEvent()
```

© 2003-2024 Stimulsoft

```
{
   // Some code before loading the viewer resources
   // ...

   return StiNetCoreViewer.ViewerEventResult(this);
}

public IActionResult OnPostViewerEvent()
{
   // Some code before the viewer actions
   // ...

   return StiNetCoreViewer.ViewerEventResult(this);
}
...
```

Information

The **ViewerEvent** action is mandatory. Without it, the viewer can't work correctly. The action is called for two types of requests: **OnGet** – the component requests the resources necessary for work, such as CSS styles, JS scripts and images; **OnPost** – all other actions of the viewer.

The **ViewerEvent** action returns a prepared HTML page of the report (or set of pages), built taking into account the current state of the viewer. If necessary, you can change the parameters of the current report in the specified action and update the report data in case of interactive actions of the viewer.

6.1.8 Printing Reports

Information

Please note that the print option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides several options for printing a report. Each has its advantages and disadvantages.

Print to PDF

Printing will be done by exporting the report to PDF. The advantages are greater

© 2003-2024 Stimulsoft

accuracy of positioning and printing of the report elements compared to other printing options. Among the drawbacks, one can mention the mandatory presence of a plug-in installed in a web browser for viewing PDF files (modern browsers have embedded PDF viewer and printer).

Print with Preview
The report will be printed in a separate pop-up browser window in HTML. The report can be previewed and then sent to the printer or copied to another location as text or HTML code. Advantages - cross-browser compatibility when printing, no need to install special plug-ins. The disadvantage is the relatively low accuracy of the position of the report elements due to the peculiarities of the implementation of HTML formatting.

Print without Preview
The report will be printed directly to the printer without preview. After selecting this menu item, the system print dialog is displayed. Since printing in this mode is carried out in the HTML format, the print quality is similar to printing a report with a preview.

Information

When printing to the **HTML format**, you should check the compliance of report page settings and printer parameters (paper size, orientation, margins, indents), as well as check your browser print settings, such as margins, headers, footers, watermarks printing, color printing.

The print function does not require additional settings for the viewer. If you need to perform any actions before printing a report, you can define a special **PrintReport** action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      PrintReport = "PrintReport"
   }
})
...
```

© 2003-2024 Stimulsoft

Index.cshtml.cs

```
...
public IActionResult OnPostPrintReport()
{
   // Some code before print
   // ...

   return StiNetCoreViewer.PrintReportResult(this);
}
...
```

Print setup

If you choose printing a report in the viewer panel, a menu with printing options is displayed. The **HTML5 Viewer** component is able to force the required printing mode. To do this, set the **PrintDestination** property to one of the following values of the **StiPrintDestination** enumeration.

- **Default** – the menu will be displayed (the default property value);
- **Pdf** – print to the PDF format;
- **Direct** – printing to the HTML format directly to the printer, the system print dialog will be displayed;
- **WithPreview** – print to the HTML format with preview in a pop-up window.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      PrintDestination = StiPrintDestination.Default
   }
})
...
```

The **HTML5 Viewer** component is able to completely disable report printing. To do this, set the value of the **ShowPrintButton** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
```

© 2003-2024 Stimulsoft

```
      ShowPrintButton = false
   }
})
...
```

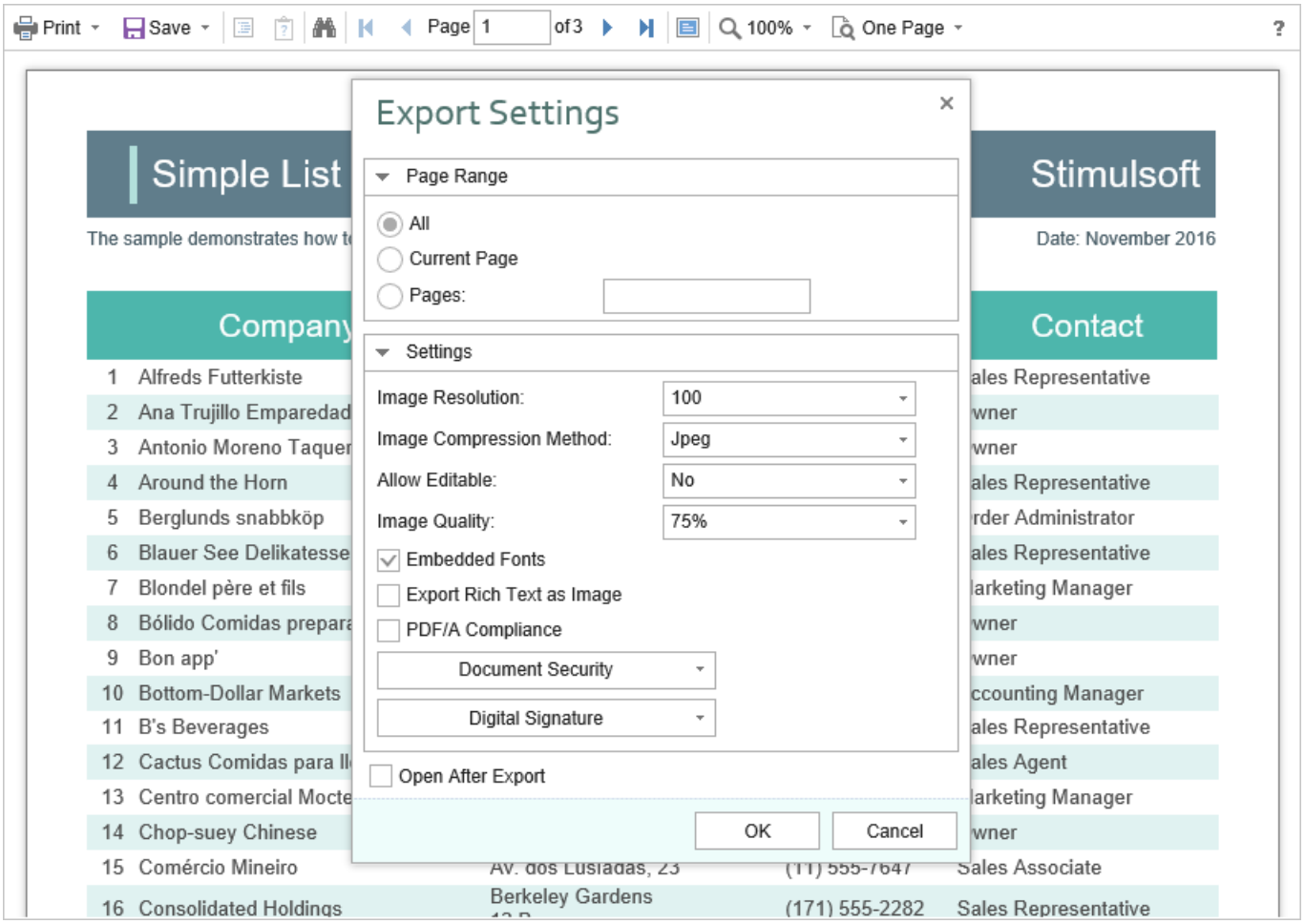
6.1.9 Exporting Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to export the displayed report to three dozen various formats, such as **PDF**, **HTML**, **Word**, **Excel**, **XPS**, **RTF**, **images**, **text**, and others. You may export the dashboard to PDF, Excel, image files.

© 2003-2024 Stimulsoft



The export function does not require additional settings for the viewer. If you need to perform any actions before exporting the report, you can define a special **ExportReport** action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      ExportReport = "ExportReport"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
```

© 2003-2024 Stimulsoft

```
{
   // Some code before export
   // ...

   return StiNetCoreViewer.ExportReportResult(this);
}
...
```

Export settings

Each report export format of the **HTML5 Viewer** component has a lot of settings, and each setting has its default values. Sometimes you need to set other default values. For this purpose, a special **DefaultSettings** property of the viewer is used. It is a container for all the default export settings.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Exports =
   {
      DefaultSettings =
      {
         ExportToPdf =
         {
            ImageQuality = 0.75f,
            ImageFormat = Stimulsoft.Report.Export.StiImageFormat.Color
         },
         ExportToHtml =
         {
            ExportMode = Stimulsoft.Report.Export.StiHtmlExportMode.Div,
            UseEmbeddedImages = true
         }
      }
   }
})
...
```

If it is required, you can completely hide export dialogs. Exporting will always be done with default settings. For this, it is enough to set the value of the **ShowExportDialog** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Exports =
   {
```

© 2003-2024 Stimulsoft

```
      ShowExportDialog = false
   }
})
...
```

The **HTML5 Viewer** component contains 30+ export formats, and sometimes you need to disable unwanted formats. This allows you to simplify UI and the use of the viewer. To disable unused export formats, it is enough to set the values for the corresponding properties of the viewer listed in the list below to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Exports =
   {
      ShowExportToDocument = true,
      ShowExportToPdf = true,
      ShowExportToXps = true,
      ShowExportToPowerPoint = true,
      ShowExportToHtml = true,
      ShowExportToHtml5 = true,
      ShowExportToMht = true,
      ShowExportToText = true,
      ShowExportToRtf = true,
      ShowExportToWord2007 = true,
      ShowExportToOpenDocumentWriter = true,
      ShowExportToExcel = true,
      ShowExportToExcelXml = true,
      ShowExportToExcel2007 = true,
      ShowExportToOpenDocumentCalc = true,
      ShowExportToCsv = true,
      ShowExportToDbf = true,
      ShowExportToXml = true,
      ShowExportToDif = true,
      ShowExportToSylk = true,
      ShowExportToImageBmp = true,
      ShowExportToImageGif = true,
      ShowExportToImageJpeg = true,
      ShowExportToImagePcx = true,
      ShowExportToImagePng = true,
      ShowExportToImageTiff = true,
      ShowExportToImageMetafile = true,
      ShowExportToImageSvg = true,
      ShowExportToImageSvgz = true
   }
})
...
```

The **HTML5 Viewer** component can completely disable the export menu. To do this,

© 2003-2024 Stimulsoft

set the value of the **ShowSaveButton** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowSaveButton = false
   }
})
...
```

6.1.10 Viewing Modes

The **HTML5 Viewer** component has two modes for displaying reports - with and without scrollbars. By default, the view mode without scrollbars is set. To enable the scrollbar view mode, set the value of the **ScrollbarsMode** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      ScrollbarsMode = true
   }
})
...
```

In the first mode (without scrollbars), the viewer displays a page or report as a whole, automatically stretching the preview space. If the width and height are specified, the viewer will truncate the page that is out of bounds. In the second mode, unlike the first one, when the page is out of bounds of the viewer's size, no truncation will be performed. Scrollbars will appear, using which you can view the entire page or report.

Information

In the report mode with scrollbars, you should set the height of the viewer, otherwise the default height will be set to **650 pixels**.

The **HTML5 Viewer** component provides the full-screen report and dashboard

© 2003-2024 Stimulsoft

mode. By default, the standard view mode is enabled, the viewer has the specified dimensions in the settings. To enable the full-screen mode, set the **FullScreenMode** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      FullScreenMode = true
   }
})
...
```

Also, to enable or disable the full-screen mode, you can use the corresponding button on the control panel of the viewer.

The **HTML5 Viewer** component has three modes to display reports - page-by-page, entire report, and tabular display of report pages. To control the modes, the **ViewMode** property is used. It can have one of the specified values - **SinglePage**, **Continuous**, **MultiplePages**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ViewMode = StiWebViewMode.SinglePage
   }
})
...
```

The **HTML5 Viewer** component supports interaction on a regular PC display and works with a touchscreen of screens and mobile devices. The **InterfaceType** property allows controlling the interface modes. The property can have one of the following values:

- **Auto** – the viewer's interface is determined automatically depending on the device the report is displayed on. That is the default value.
- **Mouse** – the standard interface with a mouse control will be used for all the screen types.
- **Touch** – the Touch interface will be used to control the viewer. The interface

© 2003-2024 Stimulsoft

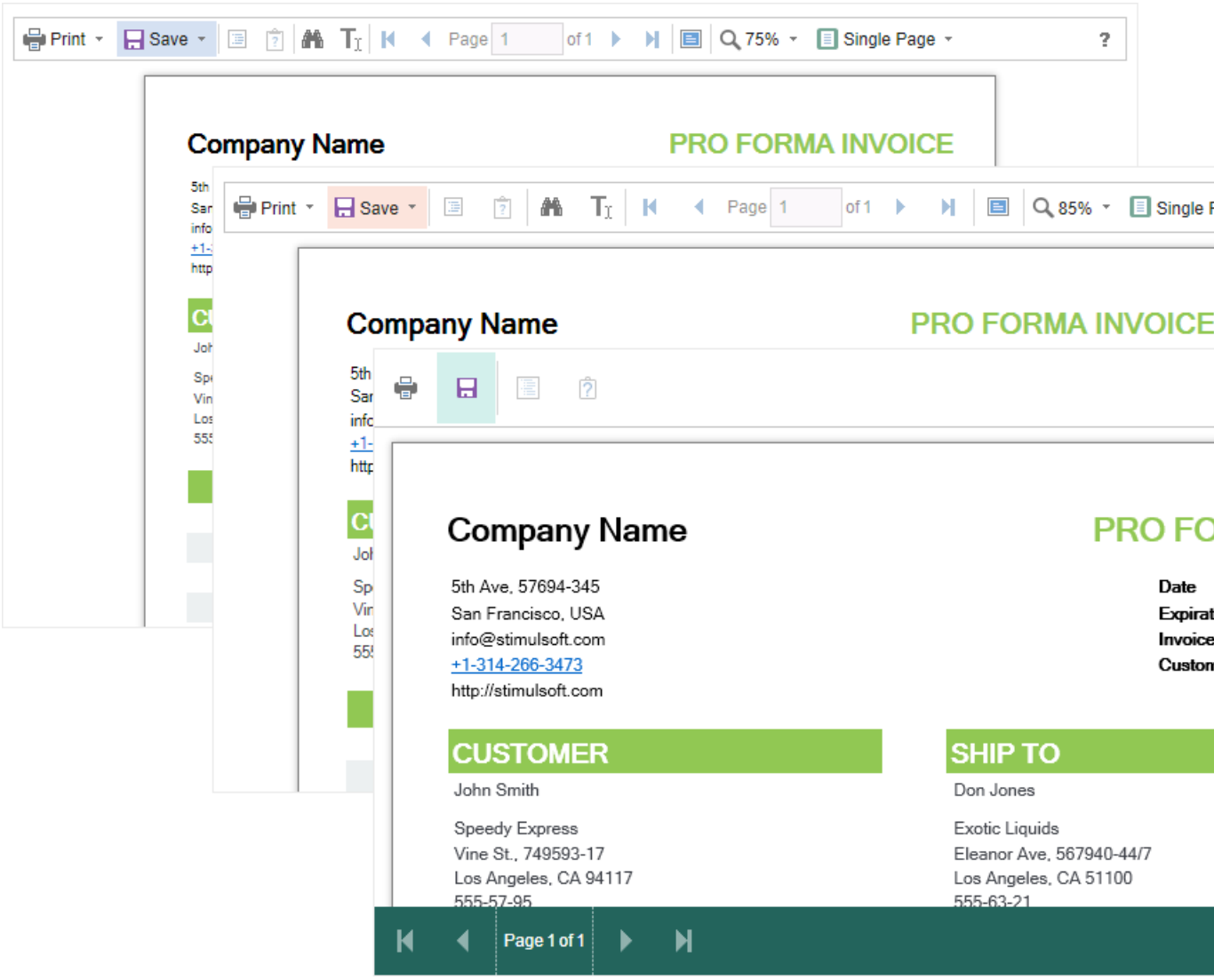
design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability.

› **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with smartphones.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      InterfaceType = StiInterfaceType.Auto
   }
})
...
```

© 2003-2024 Stimulsoft



6.1.11 Work with Parameters

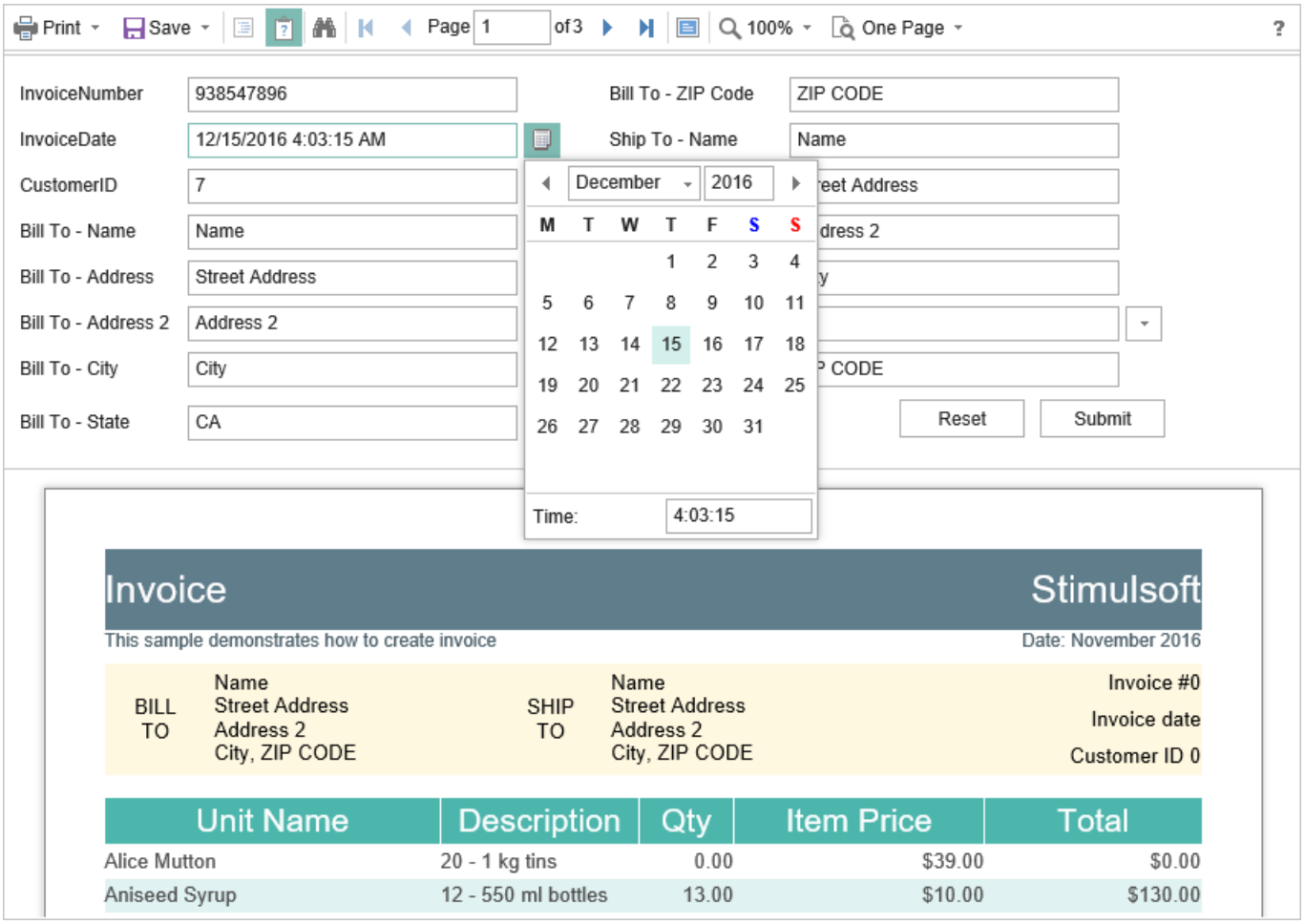
Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To work with report parameters in the **HTML5 Viewer**, there is a special settings panel. To add a parameter to the panel, you need to define a variable in a report requested by the user. When viewing a report in the viewer, such a variable will be

© 2003-2024 Stimulsoft

automatically added to the settings panel. It supports all report variables (normal variables, date and time, borders, lists, etc.).



To work with reports with parameters, no additional viewer settings are required. If you need to perform some actions before applying the parameters, you can define a special **Interaction** action.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      Interaction = "ViewerInteraction"
   }
})
...
```

© 2003-2024 Stimulsoft

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   // Some code before any interaction
   // ...

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

This action is called during any interactive actions of the viewer. If you need to perform any actions only when applying report parameters, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side on any request and contain all necessary information and states of the client part of the viewer. To determine the type of action of the viewer, it is enough to check the **Action** property of the viewer parameters.

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      // Some code before apply parameters
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

If you do not need to work with parameters, you can completely disable this feature. To do this, use the **ShowParametersButton** property in the **Toolbar** section of properties, which should be set to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowParametersButton = false
```

© 2003-2024 Stimulsoft

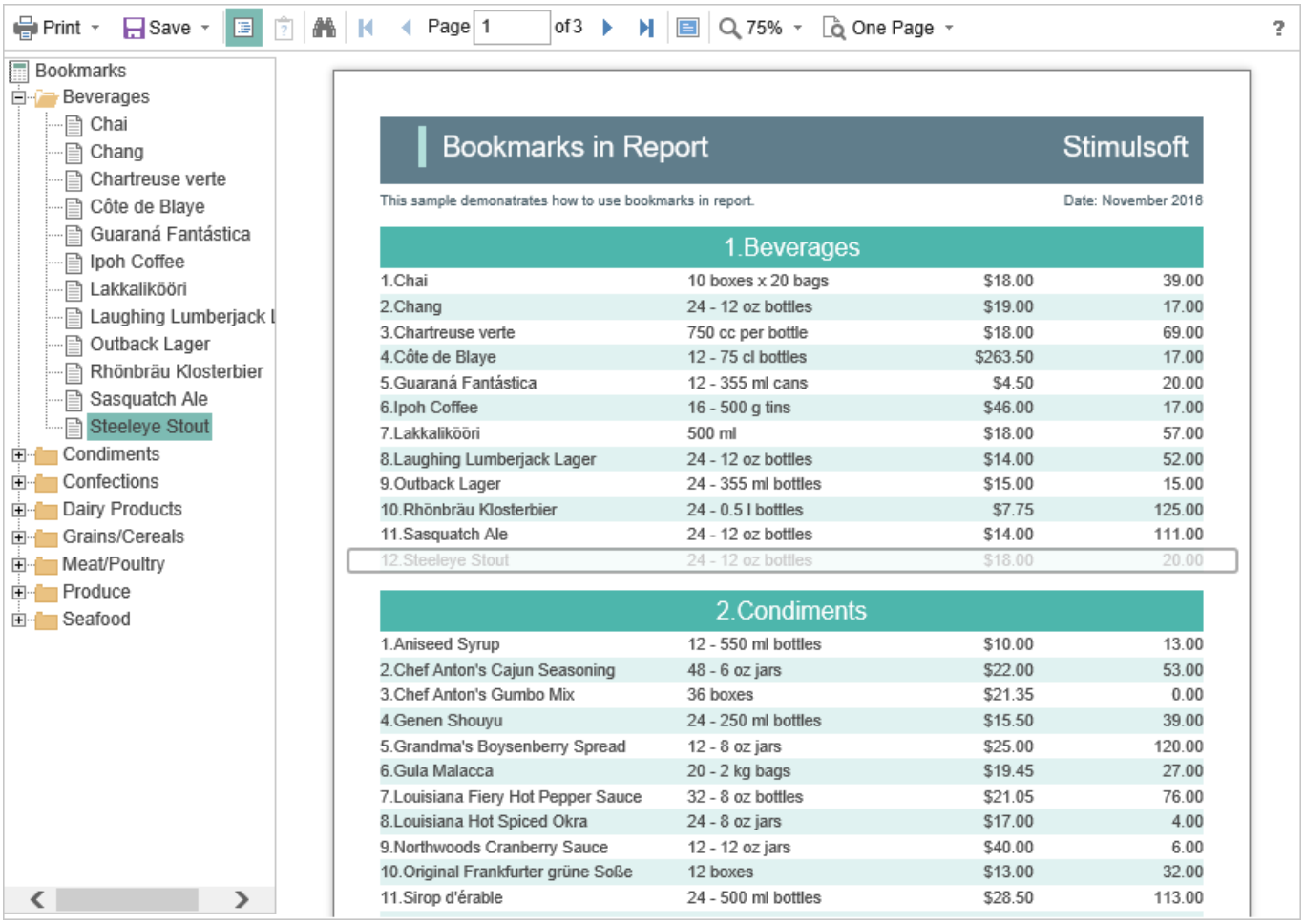
```
    }
})
...
```

Information

The options panel will not be displayed with such a viewer configuration, even if the parameters are present in the displayed report.

6.1.12 Work with Bookmarks

The **HTML5 Viewer** component supports report bookmarks. A panel with bookmarks will be displayed when displaying such a report on the left side of the page. When you select a bookmark of the report, the viewer will carry out an automatic transition to the specified page, and the report item with a bookmark is highlighted.



© 2003-2024 Stimulsoft

By default, the bookmarks bar width is 180 pixels. The **HTML5 Viewer** component allows you to change this value. For this, the **BookmarksTreeWidth** property, which value is specified in pixels, is used.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Appearance =
   {
      BookmarksTreeWidth = 200
   }
})
...
```

If work with report bookmarks is not required, you can disable this feature. For this, set the **ShowBookmarksButton** property to **false**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Toolbar =
   {
      ShowBookmarksButton = false
   }
})
...
```

Information

In this case, report bookmarks will not be displayed, even if they are present in the displayed report. This property has no effect on printing and exporting reports.

When printing a report with bookmarks, the bookmark tree will be hidden. If you want to print bookmarks with the report, it is necessary to set the **BookmarksPrint** property to **true**.

Index.cshtml

```
...
```

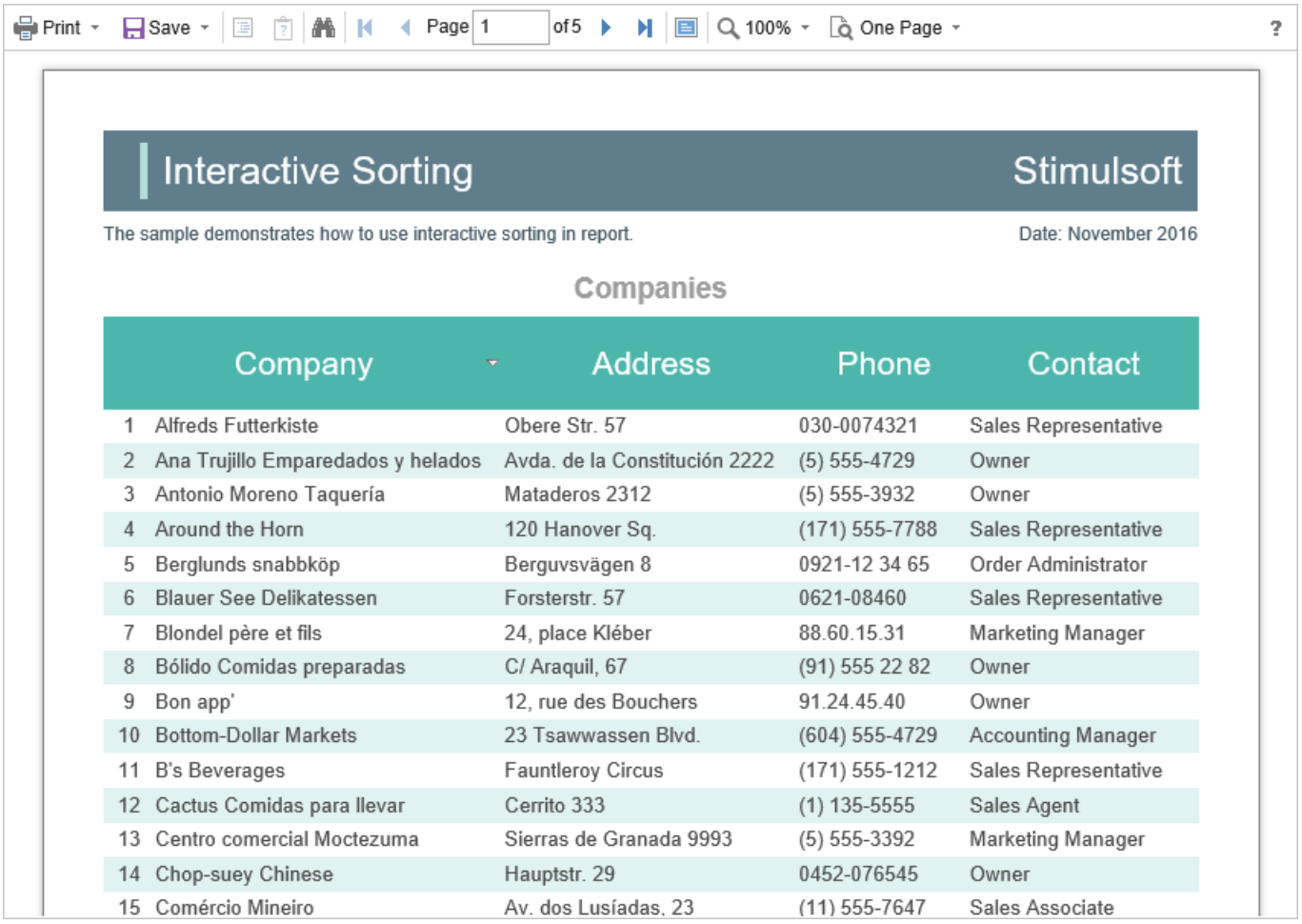
© 2003-2024 Stimulsoft

```
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions () {
   Appearance =
   {
      BookmarksPrint = true
   }
})
...
```

6.1.13 Dynamic Sorting, Collapsing, and Drill-Down

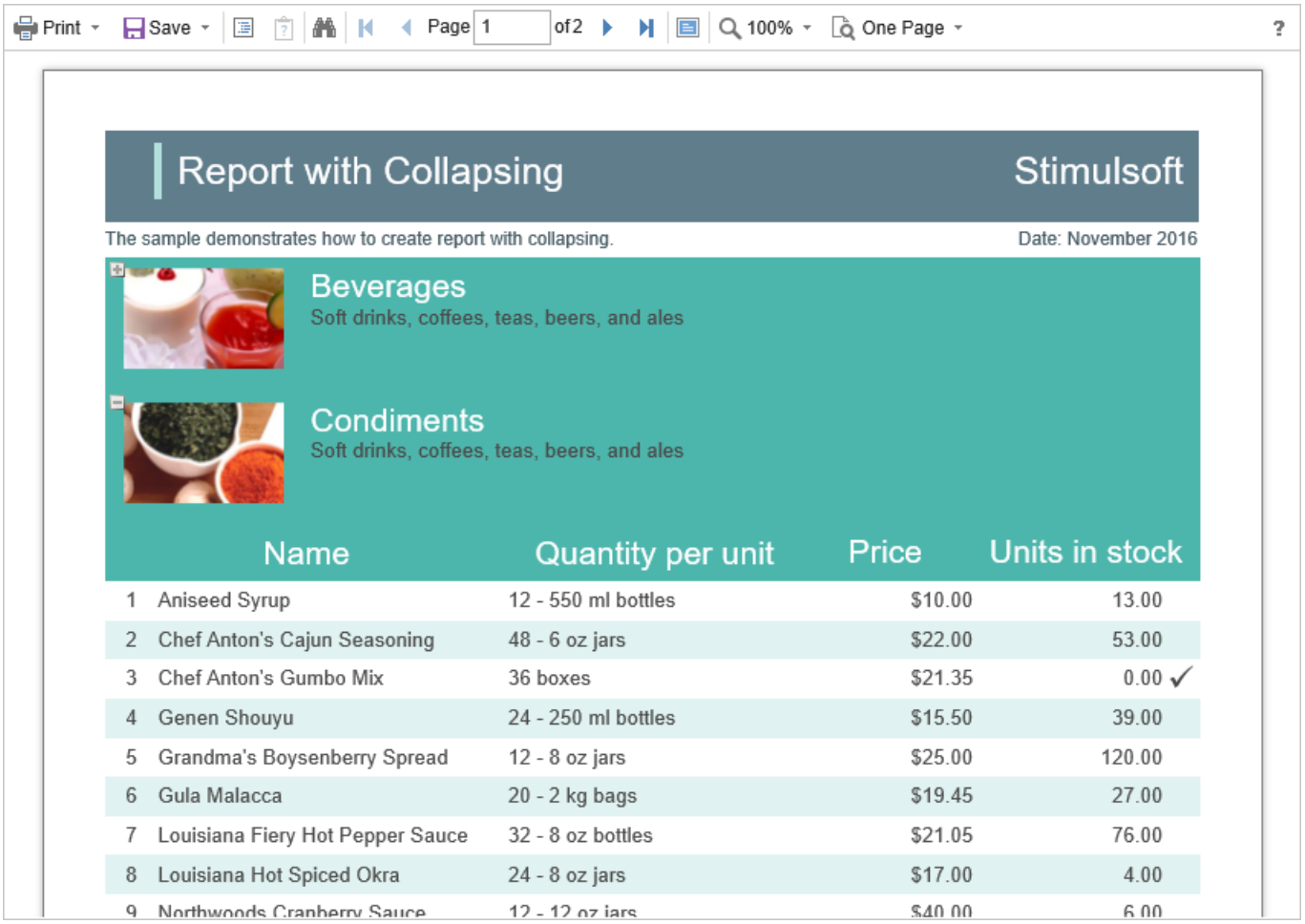
The **HTML5 Viewer** component supports dynamic sorting, collapsing, and drill-down of reports. Dynamic sorting provides the ability to change the direction of sorting in a rendered report. To do this, click on the component that has dynamic sorting enabled. Dynamic sorting is carried out in the following directions - **Ascending** and **Descending**. Each time the component is clicked, the sorting direction is reversed.

Multi-level sorting is allowed in the report. To do this, hold down the **Ctrl** key and sequentially click on the sorted components in the report. To reset sorting, you can click on any sorted component without holding down the **Ctrl** key.



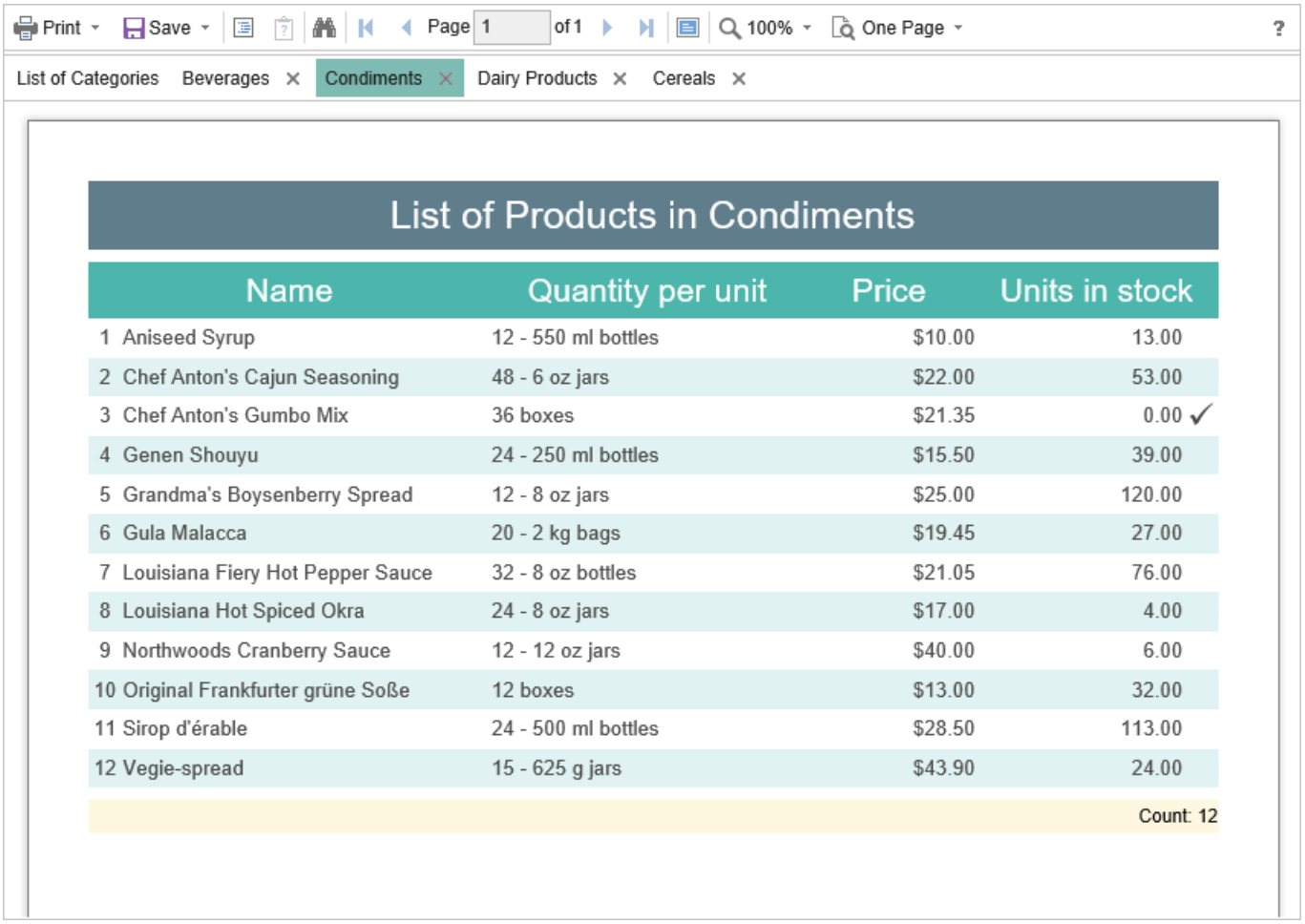
© 2003-2024 Stimulsoft

A report with dynamic collapsing is an interactive report in which blocks can collapse/expand their content when you click on the block title. Report elements, which can be collapsed/expanded, are indicated by special icons - **[-]** or **[+]**.



When using drill-down, under the main panel of the viewer, the drill-down panel with tabs for drill-down reports will be displayed. The currently displayed report will be highlighted.

© 2003-2024 Stimulsoft



To work with dynamic sorting, collapsing, and drill-down reports, no additional viewer settings are required. A special **Interaction** action is used to perform any actions before sorting, collapsing, or drill-down of the report. It will be called when interactive action of the viewer.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      Interaction = "ViewerInteraction"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   // Some code before any interaction
   // ...

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

To get the type of action, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class. They are passed to any server-side by any request and contain all necessary information and states of the client part of the viewer. For each type of interactivity, the viewer has a certain type of action:

- **Sorting** – when using column sorting;
- **DrillDown** – when using drill-down in reports;
- **Collapsing** – when using collapsing report blocks.

Index.cshtml.cs

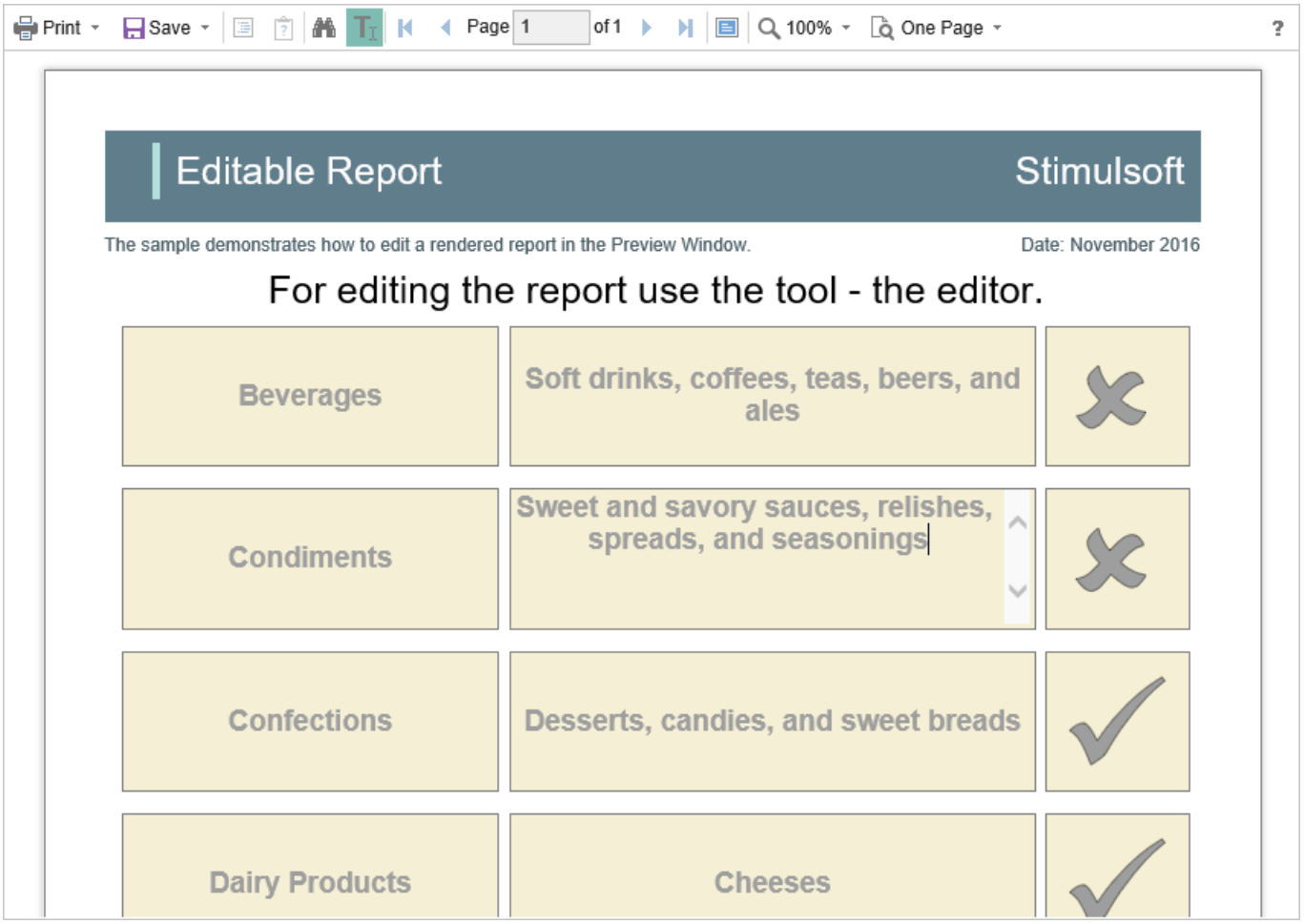
```
...
public IActionResult OnPostViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   switch (requestParams.Action)
   {
      case StiAction.Sorting:
         break;

      case StiAction.DrillDown:
         break;

      case StiAction.Collapsing:
         break;
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

6.1.14 Editing Report

The **HTML5 Viewer** component has the ability to edit report items, such as text boxes and checkboxes. You should mark the required components as editable in the report template for the editing to be possible. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be

© 2003-2024 Stimulsoft

applied to the report.



For the report edit mode, no special settings of the viewer required.

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

6.1.15 Sending Report by Email

Information

© 2003-2024 Stimulsoft

Please note that the Send Report by Email option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides the ability to send reports by email. To activate this feature, you should set the **ShowSendEmailButton** property of the viewer to **true** and define the **EmailReport** action.

Index.cshtml

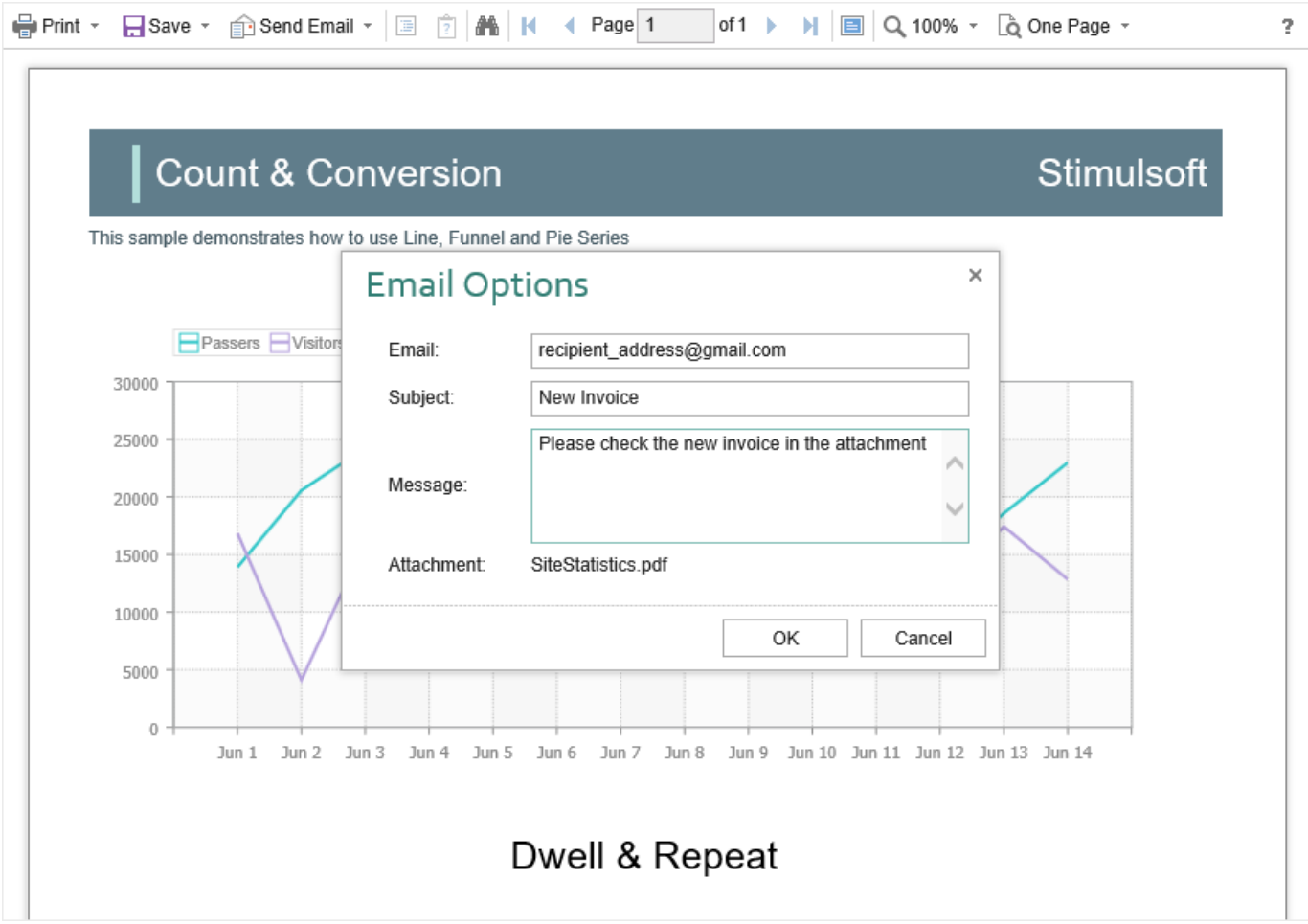
```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      EmailReport = "EmailReport"
   },
   Toolbar =
   {
      ShowSendEmailButton = true
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostEmailReport()
{
   StiEmailOptions options = StiNetCoreViewer.GetEmailOptions(this);

   // Passed from the viewer, can be checked and changed
   // options.AddressTo = "";
   // options.Subject = "";
   // options.Body = "";

   // Should be filled here
   options.AddressFrom = "admin_address@test.com";
   options.Host = "smtp.test.com";
   options.Port = 465;
   options.UserName = "admin_address@test.com";
   options.Password = "admin_password";

   // options.CC.Add("email@test.com");
   // options.BCC.Add("email@test.com");
   // options.EnableSsl = true;

   return StiNetCoreViewer.EmailReportResult(this, options);
}
...
```

© 2003-2024 Stimulsoft

When sending a report by email, the menu to select the attachment format is displayed. It corresponds to the menu for selecting the format for exporting the report. After selecting the format, the dialog to enter the send email parameters, such as the recipient's email, subject, and text of the message, is displayed.



After confirmation of sending the email, the above described **EmailReport** event will be called. You can check and correct the data entered in this form. The exported report file will be attached to the email automatically.

The **HTML5 Viewer** component allows you to set default values for the send email form. The **DefaultEmailAddress**, **DefaultEmailSubject**, and **DefaultEmailMessage** properties can be used for this. By default, these properties are empty.

Index.cshtml

© 2003-2024 Stimulsoft

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Email =
   {
      DefaultEmailAddress = "recipient_address@gmail.com",
      DefaultEmailSubject = "New Invoice",
      DefaultEmailMessage = "Please check the new invoice in the
      attachment"
   }
})
...
```

6.1.16 Calling Designer from Viewer

The **HTML5 Viewer** component has the ability to call the report designer. The special **Design** button in the toolbar of the viewer (the button is disabled by default) should be used. To use this feature, you should set the **ShowDesignButton** property to true and define the **DesignReport** event handler.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      DesignReport = "DesignReport"
   },
   Toolbar =
   {
      ShowDesignButton = true
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostDesignReport()
{
   StiReport report = StiNetCoreViewer.GetReportObject(this);
   TempData["ReportName"] = report.ReportName;

   return RedirectToPage("Designer");
}
...
```

Information

The viewer does not run the designer. It only calls the specified action, in which you can get all the necessary parameters. Then, in action, you can implement a

© 2003-2024 Stimulsoft

redirection to another View, which contains the report designer.

6.1.17 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to use the server cache to store rendered reports. If you do not use caching, you should load the report, connect data, and render it again every time you request a page. If you use caching, the previously rendered report will be loaded from the cache every time you refresh the page.

When using caching, it should be taken into account that every report saved in the cache takes up server memory and, with a large number of requests to reports, this can become a critical issue. Therefore, you need to choose between two options: low memory requirements but high in performance or low-performance requirements but high in memory.

You can manage caching with the following properties.

The CacheMode property

This property of the viewer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled. Each action of the viewer requires loading the report from the file and, if it is a report template, then render it;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this cache (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;

© 2003-2024 Stimulsoft

- **StringSession** – for caching, the current session is used. The report is saved as a packed string in this cache.

The CacheItemPriority property

This property sets the priority of the report stored in the server's cache. It affects the automatic clearing of the server memory in case of a lack of memory. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes for which you want to save the report in the server cache. If you use caching and the requested report is not found in the cache (the objects storage time has expired), then it will be requested again using a special **GetReport** event, then connect the report data and render it.

StiCacheHelper

The **HTML5 Viewer** component provides the ability to define your methods of working with report caching. For this purpose, a special class **StiCacheHelper** is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetReport**, **SaveReport** and **RemoveReport**.

Index.cshtml.cs

```
...
public class IndexModel : PageModel
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override StiReport GetReport(string guid)
      {
         var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
         guid);
         if (File.Exists(path))
         {
            StiReport report = new StiReport();
```

© 2003-2024 Stimulsoft

```
            var packedReport = File.ReadAllText(path);
            if (guid.EndsWith(GUID_ReportTemplate))
            report.LoadPackedReportFromString(packedReport);
            else report.LoadPackedDocumentFromString(packedReport);

            return report;
          }
          return null;
        }

        public override void SaveReport(StiReport report, string guid)
        {
          var packedReport = guid.EndsWith(GUID_ReportTemplate) ?
          report.SavePackedReportToString() :
          report.SavePackedDocumentToString();
          var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
          guid);
          File.WriteAllText(path, packedReport);
        }

        public override void RemoveReport(string guid)
        {
          var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
          guid);
          if (File.Exists(path))
            File.Delete(path);
        }
      }

      static IndexModel()
      {
        StiNetCoreViewer.CacheHelper = new StiMyCacheHelper();
      }
    }
    ...
```

To initialize the work with report caching using the created class, it is enough to set it as a value of the static **StiNetCoreViewer.CacheHelper** property in the controller constructor.

Information

If report caching is disabled (the **CacheMode** property of the viewer is set to **None**), the specified class will not be used.

6.1.18 Additional Methods

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

For **HTML5 Viewer**, several additional methods are used to get the object of the currently viewed report, parameters of the current state of the viewer, and other useful data. These methods can be used in any actions of the viewer.

The GetReportObject() method

Returns the report object with which the viewer is currently working. It is possible to perform the necessary actions - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the viewer, specifying it as a parameter in the resulting action method.

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   StiReport report = StiNetCoreViewer.GetReportObject(this);
   report.ReportName = "MyReportName";

   return StiNetCoreViewer.InteractionResult(this, report);
}
...
```

The GetFormValues() method

Returns the values of the form that initiated (opened by the POST request) a page of the viewer. Thus, it is possible to get a collection of form parameters in any action of the viewer.

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   NameValueCollection formValues = StiNetCoreViewer.GetFormValues(this);

   return StiNetCoreViewer.InteractionResult(this);
```

© 2003-2024 Stimulsoft

```
}
...
```

By default, this feature is disabled to optimize requests of the client-side of the viewer to the server. To enable it, set the **PassFormValues** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Server =
   {
      PassFormValues = true
   }
})
...
```

The GetRequestParams() method

Returns all parameters of the current state of the viewer passed to the server-side. They can be useful for determining the type of action that the viewer is currently executing - for example, to determine the type of export and all action parameters.

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiNetCoreViewer.GetReportObject(this);

      // Some action with report for the PDF export
      // ...

      return StiNetCoreViewer.ExportReportResult(this, report);
   }

   return StiNetCoreViewer.ExportReportResult(this);
}
...
```

You can change the values of some parameters. After making changes, for the correct operation of the viewer, you should transfer the modified parameter object to the input of the resulting method.

© 2003-2024 Stimulsoft

Index.cshtml.cs

```
...
public IActionResult OnPostViewerInteraction()
{
   StiRequestParams requestParams =
   StiNetCoreViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      requestParams.Interaction.Variables["Variable1"] = "MyValue";
      return StiNetCoreViewer.InteractionResult(this, requestParams);
   }

   return StiNetCoreViewer.InteractionResult(this);
}
...
```

The GetExportSettings() method

Returns all the parameters of the current report export. The type of the parameter object will correspond to the type of export selected in the viewer menu. Any export parameters can be changed and passed to the input of the resulting method. In this case, the report will be exported with the parameters transferred.

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
{
   StiExportSettings settings = StiNetCoreViewer.GetExportSettings(this);
   if (settings.GetExportFormat() == StiExportFormat.Pdf)
   {
      StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
      pdfSettings.EmbeddedFonts = true;
      pdfSettings.AllowEditable = StiPdfAllowEditable.No;
      return StiNetCoreViewer.ExportReportResult(this, settings);
   }

   return StiNetCoreViewer.ExportReportResult(this);
}
...
```

The MapPath() and MapWebRootPath() methods

Returns the absolute path, respectively, to the application or wwwroot directory. You can use this to upload report templates files, data files, etc. These methods are

© 2003-2024 Stimulsoft

located in the StiNetCoreHelper static class.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   StiReport report = new StiReport();
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));

   return StiNetCoreViewer.GetReportResult(this, report);
}
...
```

6.1.19 Export and Printing from Code

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** provides the ability to print reports in various ways and export reports to various formats. These actions are performed using the viewer menu. If you want to print or export a report using the code, for example, in the specific page event, you can use the special **StiNetCoreReportResponse** class. This class contains a set of static methods that allow you to print or export a report from the code, and the report viewer is not required.

Index.cshtml

```
page "{handler?}"
...
<a href="PrintReport">Print Report from Code</a>
<br />
<a href="ExportReport">Export Report from Code</a>
...
```

Index.cshtml.cs

```
...
private StiReport LoadSimpleList()
{
   DataSet dataSet = new DataSet();
   dataSet.ReadXml(Server.MapPath("Reports/Demo.xml"));
```

© 2003-2024 Stimulsoft

```
    StiReport report = new StiReport();
    report.Load(Server.MapPath("Reports/SimpleList.mrt"));
    report.RegData(dataSet);

    return report;
}

public IActionResult OnGetPrintReport()
{
    StiReport report = LoadSimpleList();

    return StiNetCoreReportResponse.PrintAsPdf(report);
    //return StiNetCoreReportResponse.PrintAsHtml(report);
}

public IActionResult OnGetExportReport()
{
    StiReport report = LoadSimpleList();

    return StiNetCoreReportResponse.ResponseAsPdf(report);
    //return StiNetCoreReportResponse.ResponseAsExcel2007(report);
    //return StiNetCoreReportResponse.ResponseAsText(report);
    //StiNetCoreReportResponse.ResponseAsJson(report);
}
...
```

The **StiNetCoreReportResponse** class contains methods for printing in PDF and HTML formats and methods to export the report in any of the supported formats. As arguments, methods can take various export settings, displaying modes and options for saving received files.

6.1.20 Timeout

When working with the **StiNetCoreViewer** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
    Server =
```

© 2003-2024 Stimulsoft

```
   {
      CacheTimeout = 10
   }
})
...
```

Using the cache will increase the speed of the report viewer. See the chapter Caching for more information.

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Server =
   {
      RequestTimeout = 30
   }
})
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection and data sources in the report.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent"
   }
})
```

© 2003-2024 Stimulsoft

```
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiNetCoreViewer.GetReportResult(this, report);
}

public IActionResult OnGetViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}

public IActionResult OnPostViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}
...
```

6.1.21 Viewer Settings

The **HTML5 Viewer** is configured using properties that are located in the **StiNetCoreViewerOptions** class. All properties are divided into groups. Some of the groups contain subgroups for ease of use. The following is an example of setting the properties of the viewer.

Index.cshtml

```
...
@Html.StiNetCoreViewer(new StiNetCoreViewerOptions() {
   Theme = StiViewerTheme.Office2022WhiteTeal,
   Localization = "Localization/en.xml",
   Actions =
   {
      GetReport = "GetReport",
      ViewerEvent = "ViewerEvent"
   },
   Appearance =
   {
      InterfaceType = StiInterfaceType.Auto,
      ScrollbarsMode = true,
      ShowTooltips = false
   },
   Exports =
   {
      DefaultSettings =
      {
```

© 2003-2024 Stimulsoft

```
        ExportToPdf =
        {
           CreatorString = "Company Name",
           ImageQuality = 0.75f
        }
     },
     ShowExportToDbf = false,
     ShowExportToDif = false
   }
})
...
```

Please note that all dashboard elements have their own save options and full-screen buttons for preview. There are no special options to control displaying them, but they can be disabled through the properties of the element. The code below should be added after loading the report before passing it to the viewer.

Index.cshtml.cs

```
...
var dbsElementInteraction = (report.GetComponentByName("RegionMap1") as
Stimulsoft.Report.Dashboard.IStiElementInteraction).DashboardInteraction;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowFullScreenButton =
false;
(dbsElementInteraction as
Stimulsoft.Report.Dashboard.IStiInteractionLayout).ShowSaveButton = false;
...
```

Main settings (without groups)

Name	Description
Theme	Sets the viewer theme. The list of available themes can be found in the **StiViewerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Sets the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used. It is built into the viewer and does not require additional XML files.
Width	Sets the width of the component in the required

© 2003-2024 Stimulsoft

	units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default width is 100%.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the automatic height is set depending on the size of the report page, or 650 pixels in the view mode of the viewer with scrollbars.

Actions

Name	Description
GetReport	Specifies the name of the action method for preparing the rendered report. Specifies the name of the action method for preparing the constructed report. If report caching is enabled, this action will be called only once when the report is requested or if the requested report is not found in the server cache.
PrintReport	Specifies the name of the action method of report printing. This is not relevant when viewing dashboards.
ExportReport	Specifies the name of the action method of the export the report to the specified format.
EmailReport	Specifies the name of the action method of sending the report by email. This is not relevant when viewing dashboards.
Interaction	Specifies the name of the action method for the viewer to work with interactive operations, such as using parameters, dynamic sorting, collapsing, and drill-down.

© 2003-2024 Stimulsoft

DesignReport	Specifies the name of the action method to go to the specified view by clicking the Design button on the viewer panel.
ViewerEvent	Specifies the name of the action method of basic viewer events and the processing actions of the viewer, such as printing and exporting a report, working with parameters, and interactivity, if these actions are not specified separately. In addition, this action is used to load scripts and styles of the viewer. This action is mandatory.

Server

Name	Description
RouteTemplate	Sets the route template that is returned when the report viewer actions are executed. If the property is not set, then the Razor project template will be used instead. The default value of the property is null.
AllowAutoUpdateCookies	Allows the viewer to update the cookies automatically on every request to the server. By default, cookies are set when creating the viewer, if they are not specified in the report. By default, the property is set to **false**.
AllowAntiforgeryToken	Allow the viewer to automatically request and send the antiforgery token. By default, the property is set to **true**.
RequestTimeout	Sets the response timeout from the server in seconds, after which an error will be generated. The default value is 20 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the report since the last action of the viewer. The default value is 20 minutes.

© 2003-2024 Stimulsoft

CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled, the report will be reloaded each time using the **GetReport** event; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as storage, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or the server session will be automatically re-saved after a certain period of time when the viewer is idle (every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the viewer mode in which relative URLs are used for AJAX requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value that specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersToReport	Enables using all the URL parameters of the request as the variable values. The variable

© 2003-2024 Stimulsoft

	names must match the parameters. The default value of the property is **false**.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the viewer. If **false**, only the necessary parameters are used to request the resources of the viewer. This corresponds to the correct work of the browser cache. By default, the property is set to **true**.
PassFormValues	Enables passing the values of the POST form to the client-side, if these values are required to be used in the actions of the viewer. If you enable this property, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
ShowServerErrorPage	Enables displaying an HTML page with the details of the error that occurred on the server-side. When the property is enabled, the details of the error will be displayed in the viewer window. If the property is disabled, only the numeric error code and a short error text in the dialog box will be displayed. By default, the property is set to **true**.
UseCompression	Enables compression of the viewer requests into the GZip stream. Enables compression of the viewer requests into the GZip stream. That allows to decrease the amount of internet traffic but slows down the viewer slightly. The default value of the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server-side. The following resources are supported - scripts, styles, and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
UseLocalizedCache	Sets a value that enables the use of a different

© 2003-2024 Stimulsoft

	cache depending on the selected localization. The default value of the property is **false**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Sets the path to the CSS file of the viewer's styles. The standard styles of the chosen theme will not be loaded if this property has got a value. The default value of the property is an empty string.
BackgroundColor	Sets the background color of the viewer. By default, it is set to **White**.
PageBorderColor	Sets the border color of the viewer. By default it is set to **Gray**.
RightToLeft	Sets the **Right to Left** mode for viewer controls. By default, the property is set to **false**. By default, the property is set to **false**.
FullScreenMode	Sets the full-screen display mode of the viewer. By default, the property is set to **false**.
ScrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to **false**.
OpenLinksWindow	Sets the target window for opening links contained in the report. By default, the property is set to **Blank** (new window).
OpenExportedReportWindow	Sets the target window for opening the export file from the viewer. By default, the property is set to **Blank** (new window).
DesignWindow	Sets the destination window for launching the report designer. The default value of the

© 2003-2024 Stimulsoft

	property is **Self** (which is the current window).
ShowTooltips	Enables showing tips for the viewer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enables showing links to online documentation for the viewer controls. By default, the property is set to **true**.
ShowDialogsHelp	Sets a value that indicates that showing or hiding the help button in dialogs. By default, the property is set to **true**.
PageAlignment	Sets the position of the report page in the viewer window. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – the page will be aligned left; › **Center** – the page will be centered (default value); › **Right** – the page will be aligned right.
ShowPageShadow	Enables displaying shadow for report pages. By default, the property is set to **true**.
BookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default, the property is set to **false**.
BookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
ParametersPanelPosition	Specifies the position of the report parameters panel. It can take one of the following **StiParametersPanelPosition** enumeration values: › **Top** - the panel will be docked to the top margin (default value); › **Left** - the panel will be docked to the left margin.
ParametersPanelMaxHeight	Sets the maximum height of the parameters bar

© 2003-2024 Stimulsoft

	in pixels. By default, the maximum height is 300 pixels.
ParametersPanelColumnsCount	Sets the number of columns to display report parameters. By default, there are 2 columns.
ParametersPanelSortDataItems	Gets or sets a value which indicates that variable items will be sorted. By default, the property is set to **true**.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
InterfaceType	Sets the type of interface used for the viewer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the viewer's interface is determined automatically depending of the device that is report is displayed on. That is the default value. › **Mouse** – the standard interface with a mouse control will be used for all the screen types. › **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability. › **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.
AllowMobileMode	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to **false**, then the mobile view will not be used. If the option is set to **true**, the mobile view mode will be used when opening the

© 2003-2024 Stimulsoft

	viewer on mobile devices. By default, the option is set to **true**.
ChartRenderType	Sets the displaying mode of charts on the report page. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages. It can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
DatePickerFirstDayOfWeek	Sets the first day of the week for the date picker. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Auto** – automatic detection of the first day of the week from the browser settings (default value); › **Monday** – the first day of the week is Monday; › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date

© 2003-2024 Stimulsoft

	picker. By default, the property is set to **false**.
AllowTouchZoom	Sets ability to change the scale of the report page by using the two-fingers gesture (Pinch to Zoom) for the touch-screens. The default value of the property is **true**.
ShowReportIsNotSpecifiedMessage	Sets a value which indicates that 'The report is not specified' message will be shown. The default value of the property is **true**.
PrintToPdfMode	Sets the Print to PDF mode. It has the following values: › **StiPrintToPdfMode.Hidden** - hidden print mode (default value); › **StiPrintToPdfMode.Popup** - the PDF document will be displayed before printing in a pop-up window.
ImagesQuality	Gets or sets the image quality that will be used on the viewer page. It has the following values: › **StiImagesQuality.Low** - low quality, used to speed up loading reports and saves memory; › **StiImagesQuality.Normal** - normal quality, suitable for most cases (default value); › **StiImagesQuality.High** - high quality, used for ultra high-definition displays, but may slow down the loading of pages.
CombineReportPages	Sets a value which indicates that if a report contains several pages, then they will be combined in preview. By default, the property is set to **false**.

Toolbar

Name	Description
Visible	Enables displaying the viewer toolbar. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

DisplayMode	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the **StiToolbarDisplayMode** enumeration: › **Simple** - all controls are located on the same control panel (default value); › **Separated** - the control panel is split into top and bottom panels.
BackgroundColor	Specifies the background color of the viewer toolbar. The default color of the selected theme is used.
BorderColor	Specifies the border color of the viewer toolbar. The default color of the selected theme is used.
FontColor	Specifies the text color for the toolbar and the viewer menu. The default color of the selected theme is used.
FontFamily	Specifies the font for the toolbar and the viewer menu. The default font of the selected theme is used.
Alignment	Sets the alignment mode for the controls on the viewer toolbar. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – elements will be aligned left; › **Center** – elements will be centered; › **Right** – elements will be aligned right; › **Default** – the alignment depends on the RightToLeft property (default value).
ShowButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default, the property is set to **true**.
ShowPrintButton	Enables showing the button - **Print** - on the viewer toolbar. By default, the property is set to **true**.
ShowOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports or

© 2003-2024 Stimulsoft

	dashboards. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to true.
ShowSendEmailButton	Enables showing the button - **Send Email** - on the viewer toolbar. By default, the property is set to **false**. Also, you should add the EmailReport action.
ShowFindButton	Enables showing the button - **Find** - on the viewer toolbar. By default, the property is set to **true**.
ShowBookmarksButton	Enables showing the button - **Bookmarks** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
ShowParametersButton	Enables showing the button - **Parameters** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.
ShowResourcesButton	Enables showing the button - **Resources** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.
ShowEditorButton	Enables showing the button - **Editor** - on the viewer toolbar. By default, the property is set to **true**.
ShowFullScreenButton	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports or dashboards. . By default, the property is set to **true**.
ShowFirstPageButton	Enables showing the button - **First Page** - on

© 2003-2024 Stimulsoft

	the viewer toolbar. By default, the property is set to **true**.
ShowPreviousPageButton	Enables showing the button - **Previous Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowCurrentPageControl	Enables showing the current report page indicator. By default, the property is set to **true**.
ShowNextPageButton	Enables showing the button - **Next Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowLastPageButton	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowZoomButton	Enables showing the button to select the report zoom. By default, the property is set to **true**.
ShowViewModeButton	Enables showing the button to select the view mode of the report page. By default, the property is set to **true**.
ShowDesignButton	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **false**.
ShowAboutButton	Enables showing the button - **About** - on the viewer toolbar. By default, the property is set to **true**.
ShowRefreshButton	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to **true**.
ShowPinToolbarButton	Enables displaying of the **Pin Toolbar** button on the viewer's toolbar. The button is available only in the Mobile mode of the viewer's interface. The default value of the property is **true**.
PrintDestination	Sets the report printing mode. It can take one of the following values of the **StiPrintDestination**

© 2003-2024 Stimulsoft

	enumeration: › **Default** – a menu with a choice of printing modes will be displayed (default value); › **Pdf** – printing will be done in the PDF format; › **Direct** – printing will be done to the HTML format directly to the printer, the system print dialog will be displayed; › **PopupWindow** – printing will be done in the HTML format via the preview window of the report.
ViewMode	Sets the mode for displaying report pages. It can take one of the following **StiWebViewMode** enumeration values: › **SinglePage** - displays one page of the report selected in the toolbar of the viewer (default value); › **Continuous** - displays all pages of the report; › **MultiplePages** - displays all report pages as a table.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of the following values: › **StiZoomMode.PageWidth** – when the viewer runs, the zoom, necessary to display the report by the page width, will be set; › **StiZoomMode.PageHeight** – when the viewer runs, the zoom, necessary to display the report by the page height, will be set.
MenuAnimation	Enables animation when the viewer menu shows/hides. By default the property is set to **true**.
ShowMenuMode	Sets the display mode of the viewer menu. It can take one of the following values of the **StiShowMenuMode** enumeration:

© 2003-2024 Stimulsoft

	› **Click** – shows menu by mouse click (default value); › **Hover** – shows menu by hovering the mouse cursor.
AutoHide	Enables auto-hiding of the viewer's toolbar. The property will work only for the Mobile mode of the viewer's interface. The default value of the property is **false**.

Export

Name	Description
DefaultSettings	This group of properties provides the ability to specify the default export settings for each export type. These settings will be applied to the export dialogs when the viewer runs or to the report, if export dialogs are disabled.
StoreExportSettings	Enables saving selected settings in the export dialogs. Settings will be stored in browser cookies. By default the property is set to **true**.
ShowExportDialog	Enables showing the export options dialog box. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
ShowExportToDocument	Enables the export menu item - **Document File**. By default, the property is set to **true**.
ShowExportToPdf	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToXps	Enables the export menu item - **Microsoft XPS File**. By default, the property is set to **false**.
ShowExportToPowerPoint	Enables the export menu item - **Microsoft PowerPoint 2007/2010 File**. By default, the

© 2003-2024 Stimulsoft

	property is set to **true**.
ShowExportToHtml	Enables the export menu item - **HTML File**. By default, the property is set to **true**.
ShowExportToHtml5	Enables the export menu item - **HTML5 File**. By default, the property is set to **true**.
ShowExportToMht	Enables the export menu item - **MHT Web Archive**. By default, the property is set to **true**.
ShowExportToText	Enables the export menu item - **Text File**. By default, the property is set to **true**.
ShowExportToRtf	Enables the export menu item - **Rich Text File**. By default, the property is set to **true**.
ShowExportToWord2007	Enables the export menu item - **Microsoft Word 2007/2010 File**. By default, the property is set to **true**.
ShowExportToOpenDocumentWriter	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to **true**.
ShowExportToExcel	Enables the export menu item - **Microsoft Excel File**. By default, the property is set to **true**.
ShowExportToExcelXml	Enables the export menu item - **Microsoft Excel Xml File**. By default, the property is set to **true**.
ShowExportToExcel2007	Enables displaying the **Microsoft Excel 2007/2010 File** export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToOpenDocumentCalc	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to **true**.
ShowExportToCsv	Enables the export menu item - **CSV File**. By default, the property is set to **true**.
ShowExportToDbf	Enables the export menu item - **DBF File**. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToXml	Enables the export menu item - **XML File**. By default, the property is set to **true**.
ShowExportToDif	Enables the export menu item - **Data Interchange Format (DIF) File**. By default, the property is set to **true**.
ShowExportToSylk	Enables the export menu item - **Symbolic Link (SYLK) File**. By default, the property is set to **true**.
ShowExportToJson	Enables the export menu item - **JSON File**. By default, the property is set to **true**.
ShowExportToImageBmp	Enables displaying the **BMP Image** export menu item when viewing reports, and the **BMP Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageGif	Enables displaying the **GIF Image** export menu item when viewing reports, and the **GIF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageJpeg	Enables displaying the **JPEG Image** export menu item when viewing reports, and the **JPEG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePcx	Enables displaying the **PCX Image** export menu item when viewing reports, and the **PCX Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePng	Enables displaying the **PNG Image** export menu item when viewing reports, and the **PNG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageTiff	Enables displaying the **TIFF Image** export menu item when viewing reports, and the **TIFF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvg	Enables displaying the **Scalable Vector**

© 2003-2024 Stimulsoft

	Graphics (SVG) File export menu item when viewing reports, and the **Scalable Vector Graphics (SVG) File** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvgz	Enables displaying the **Compressed SVG (SVGZ) File** export menu item when viewing reports, and the **Compressed SVG (SVGZ) File** item when viewing dashboards. By default, the property is set to **true**.
ShowOpenAfterExport	Enables displaying the **Open After Export** parameter in export settings menu. By default, the property is set to **true**.

Email

Name	Description
ShowEmailDialog	Enables displaying settings for sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server side in the **EmailReport** action. By default the property is set to **true**.
ShowExportDialog	Enables displaying export options dialog box when sending email. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
DefaultEmailAddress	Sets the default recipient email, i.e. the address to which the email with the attached report will be sent.
DefaultEmailSubject	Sets the default email subject (header).
DefaultEmailMessage	Sets the default email message (text).
DefaultEmailReplyTo	Gets or sets the default text of the replyTo of the message created in the viewer.

© 2003-2024 Stimulsoft

6.2 HTML5 Designer

Samples

See on GitHub examples of working with the ASP.NET Core Razor HTML5 Designer component. All examples are separate projects, grouped into one solution for Visual Studio.

The **HTML5 Designer** (**StiNetCoreDesigner**) component is designed to create reports in the web browser. You do not need to install the .NET Framework, ActiveX components or any special plug-ins on the client side. All that is needed is any modern Web browser.

With help of **HTML5 Designer** you can create, edit, save and preview reports on any computer with any operating system installed. Since the designer only uses HTML and JavaScript technologies, it can be run on devices where there is no Flash or Silverlight support - tablets, smartphones. Also, the designer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Designer** component uses the AJAX technology to perform all actions on reports, which allows you to get rid of reloading the entire page, save Web traffic and speed up work. The report engine built using the .NET Core technology is used to render reports. This is a cross-platform technology. It allows you to deploy the application on servers that use the operating systems like Windows, macOS, and Linux.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Designer** in a Web project, you need to install the NuGet package of Stimulsoft.Reports.Web.NetCore:

› Select "Manage NuGet Packages ..." in the context menu of the project;

© 2003-2024 Stimulsoft

› Specify Stimulsoft.Reports.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

To add the ability to create and edit dashboards in a Web project, install the NuGet package Stimulsoft.Dashboards.Web.NetCore (this package is associated with the package Stimulsoft.Reports.Web.NetCore. If it is missed it will be installed automatically):
› Select "Manage NuGet Packages ..." in the context menu of the project;
› Specify Stimulsoft.Dashboards.Web.NetCore in the search bar on the Browse tab;
› Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

- How this Works?
- Activation
- Editing Reports and Dashboards
- Creating New Reports and New Dashboards
- Saving Reports and Dashboards
- Preview
- Settings
- Additional Features of Preview
- Timeout
- Localization
- Using Themes
- Caching
- Additional Methods

6.2.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the designer, you should place the **StiNetCoreDesigner** component on a page, set the necessary settings, and define the actions in the page event handler.

© 2003-2024 Stimulsoft

When running the report designer, the following actions occur:
› The .NET Core component generates HTML and JavaScript code that is necessary for displaying and running the designer;
› When the component is output, the JavaScript method is launched. It requests the report template on the server side displays it in the designer window;
› Various actions in the designer (for example, report preview, saving the report template, export reports, sorting, drill-down etc.) calls a certain action on the server side, in which you can perform the necessary manipulations with the report.

6.2.2 Activation

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiNetCoreDesigner** component.

Index.cshtml.cs

```
...
//Activation with using license code
public class IndexModel : PageModel
{
   static IndexModel()
   {
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";
   }
}

//Activation with using license file
public class IndexModel : PageModel
{
   public IndexModel(IWebHostEnvironment webHostEnvironment)
   {
      var path = Path.Combine(webHostEnvironment.ContentRootPath, "Content\
      \license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

6.2.3 Editing Reports and Dashboards

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To edit a report template, you should add the **StiNetCoreDesigner** component to a page, set the minimum required settings for it, and define necessary actions in the page event handler.

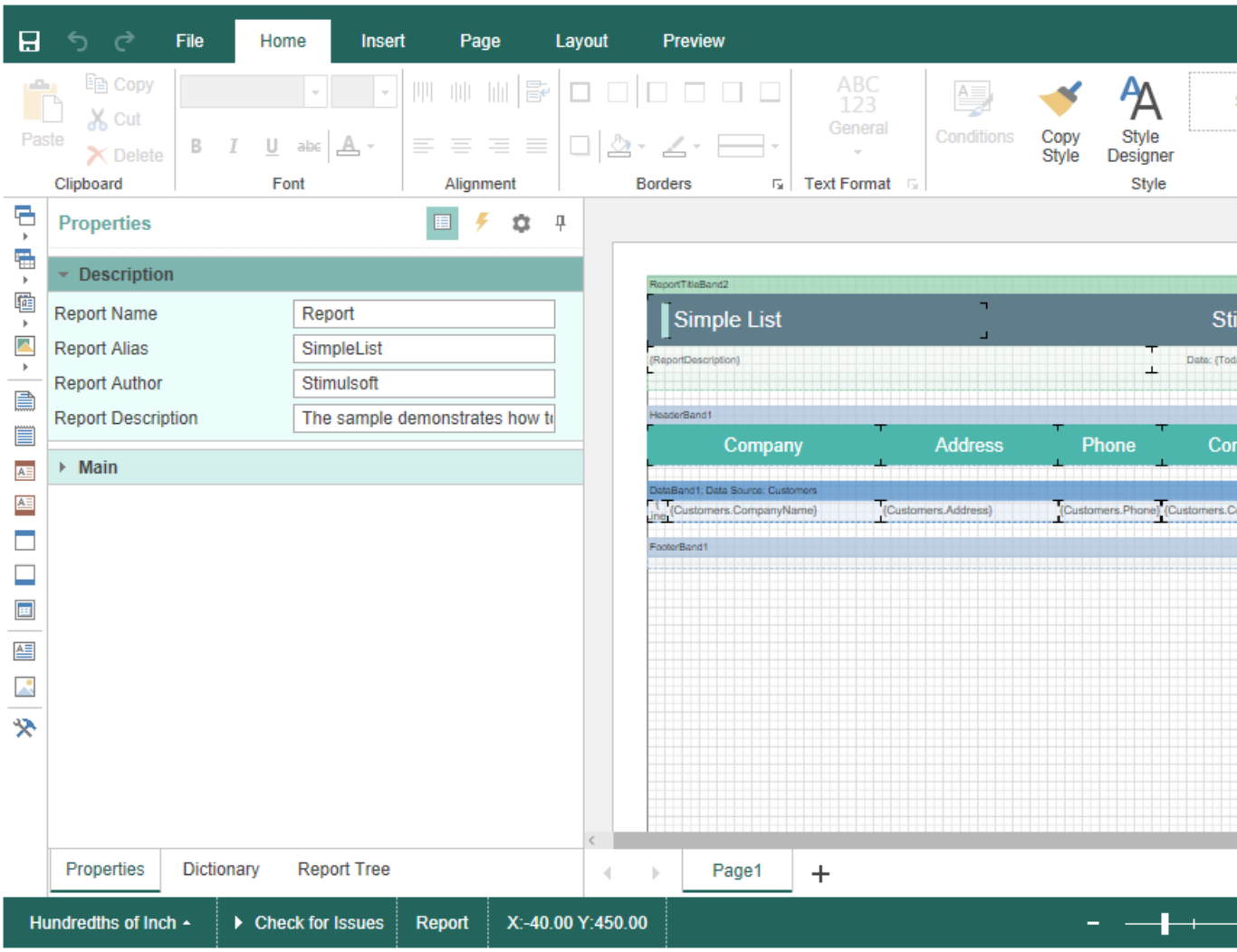
Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetReport = "GetReport",
      DesignerEvent = "DesignerEvent"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   // Create the report object
   StiReport report = new StiReport();

   // Load report or dashboard
   report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));
   //report.Load(StiNetCoreHelper.MapPath(this, "Reports/Dashboard.mrt"));

   return StiNetCoreDesigner.GetReportResult(this, report);
}

public IActionResult OnGetDesignerEvent()
{
   return StiNetCoreDesigner.DesignerEventResult(this);
}

public IActionResult OnPostDesignerEvent()
{
   return StiNetCoreDesigner.DesignerEventResult(this);
}
...
```

© 2003-2024 Stimulsoft



The **GetReport** action is used to load an editable report template. It is called automatically after the report designer is loaded. The **DesignerEvent** action is designed to process various additional designer actions, such as working with data and components, previewing reports and others.

Information

The **DesignerEvent** action is mandatory. Without it, the correct work of the designer is impossible. The action is called for two types of requests: **OnGet** – a component requests necessary resources for work, such as CSS styles, JS scripts, and images; **OnPost** – all other actions of the designer.

© 2003-2024 Stimulsoft

6.2.4 Creating New Reports and New Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To run the report designer with a new (empty) report, it is enough to create a new report in the **GetReport** action and return it to the designer. If necessary, you can load data for the report, or perform any other necessary actions.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetReport = "GetReport"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
   StiReport report = new StiReport();

   return StiNetCoreDesigner.GetReportResult(this, report);
}
...
```

You can also create a new report using the main menu of the designer. The **CreateReport** action is used to load data for a new report or perform any other necessary actions. This action will be called when creating a new empty report or when creating a report using the wizard.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
```

© 2003-2024 Stimulsoft

```
   Actions =
   {
      CreateReport = "CreateReport"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostCreateReport()
{
   StiReport report = new StiReport();
   //var newDashboard = StiReport.CreateNewDashboard();

   // Register data for the new report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);
   //newDashboard.RegData(data);
   report.Dictionary.Synchronize();
   //newDashboard.Dictionary.Synchronize();

   return StiNetCoreDesigner.GetReportResult(this, report);
   //return StiNetCoreDesigner.GetReportResult(this, newDashboard);
}
...
```

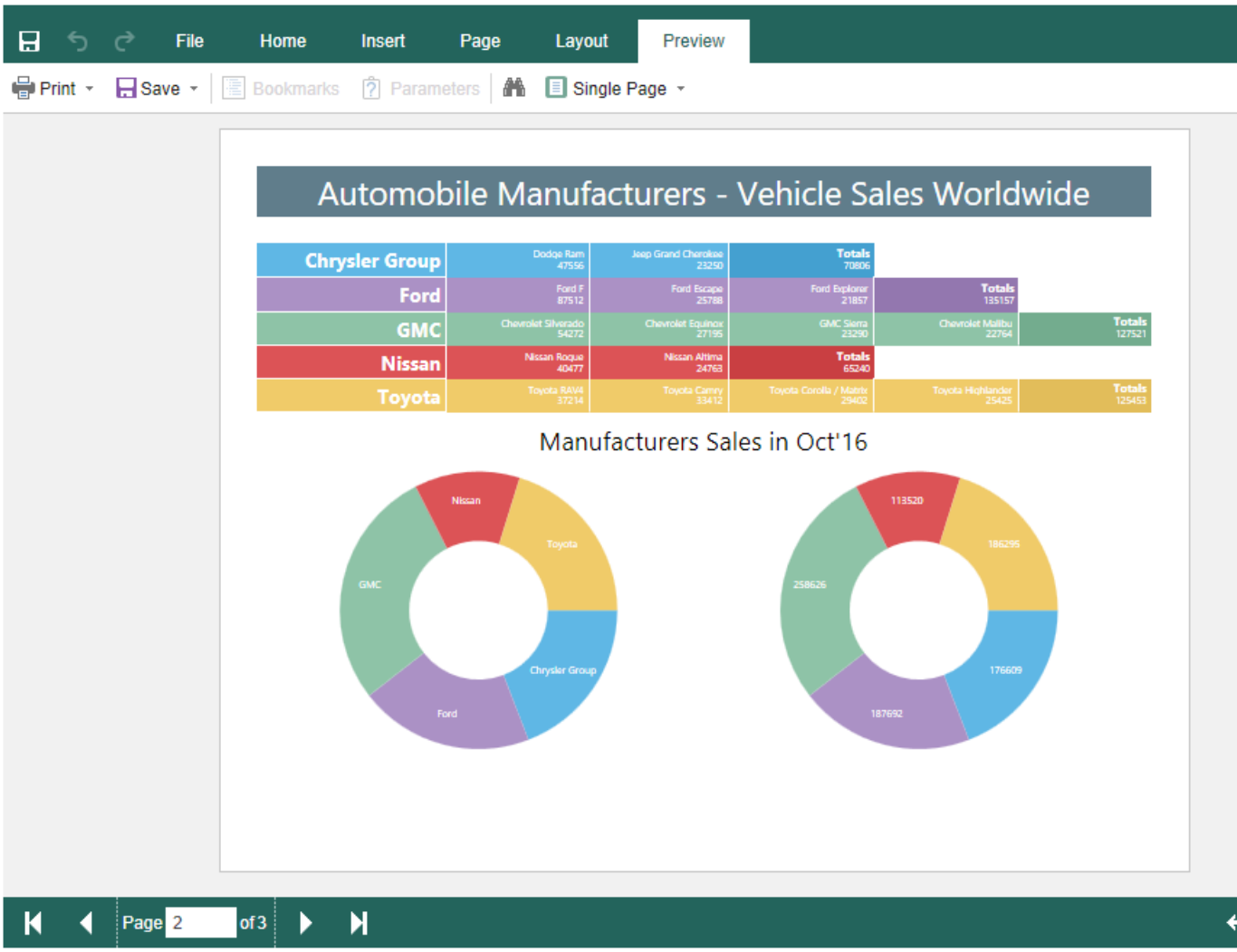
6.2.5 Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides the ability to preview reports. To preview the report, just go to the appropriate tab in the designer window. The report template will be transferred to the server side, rendered and displayed in the embedded viewer.

© 2003-2024 Stimulsoft



Before previewing the report, it is possible to perform any necessary actions, for example, connect data for the report. To do this, you can use the special **PreviewReport** action that will be called before previewing the report. The **PreviewReport** action is called before preparing and rendering a report for viewing till its saving to the cash.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      PreviewReport = "PreviewReport"
   }
})
...
```

© 2003-2024 Stimulsoft

Index.cshtml.cs

```
...
public IActionResult OnPostPreviewReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);

   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);

   return StiNetCoreDesigner.PreviewReportResult(this, report);
}
...
```

If you need to make actions on your report immediately before displaying the report, you can use the **GetPreviewReport** action, which is called after the request of the prepared report from the cash.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      GetPreviewReport = "GetPreviewReport"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostGetPreviewReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);

   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);
   //report.IsRendered = false;

   return StiNetCoreDesigner.PreviewReportResult(this, report);
}
...
```

Information

© 2003-2024 Stimulsoft

So as in this event a prepared report for viewing is transferred, if you need to render again you should set the **report.IsRendered = false** flag.

6.2.6 Additional Features of Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The preview window of the **HTML5 Designer** component has a fully functional interactive **HTML5 Viewer** that can print and export reports, supports working with report parameters, dynamic sorting, interactive reports, collapsing and etc. To use these features, you do not need any additional settings for the report designer.

In any of the above actions, you can work with the report template, for example, change its properties and parameters, connect new data for rendering.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      ExportReport = "ExportReport"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);
   // ...

   return StiNetCoreDesigner.ExportReportResult(this, report);
}
...
```

© 2003-2024 Stimulsoft

Information

If you do not need any of these additional options to preview the report (for example, exporting or printing a report), you can disable them using the appropriate properties of the **HTML5 Designer** component.

6.2.7 Saving Reports and Dashboards

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides two ways of saving the report which are available in the main menu and in the main panel of the designer - **Save Report** and **Save As**. In turn, each of these ways has its own modes and settings.

Saving a report and dashboard on the server side

To save the editable report on the server side, you need to set the **SaveReport** action, which will be called when you select **Save** in the main menu, or click the **Save** button on the main panel of the designer.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      SaveReport = "SaveReport"
   }
})
...
```

Index.cshtml.cs

```
...
public IActionResult OnPostSaveReport()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);
```

© 2003-2024 Stimulsoft

```
    // Save the report template
    // ...

    return StiNetCoreDesigner.SaveReportResult(this);
}
...
```

This action returns a response to the client side of the designer about the result of saving the report. After saving the report, it is possible to display a dialog box with an error or a text message.

Index.cshtml.cs

```
...
public IActionResult OnPostSaveReport()
{
    StiReport report = StiNetCoreDesigner.GetReportObject(this);

    // Save the report template
    // ...

    // Completion of the report saving with message dialog box
    return StiNetCoreDesigner.SaveReportResult(this, "Some message after
    saving");
    //return Content("{"infoMessage":"Some info message after saving"}");
    //return Content("{\"warningMessage\":\"Some info message after saving
    \"}");
}
...
```

You can get a report name from the designer save dialog or an original report name.

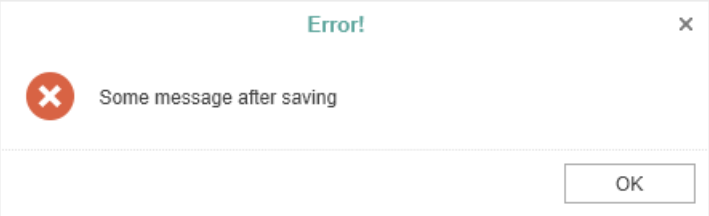
Index.cshtml.cs

```
...
public IActionResult OnPostSaveReport()
{
    var requestParams = StiNetCoreDesigner.GetRequestParams();
    var report = StiNetCoreDesigner.GetReportObject();

    //Report name from designer save dialog
    var savingReportName = requestParams.Designer.FileName;

    //Original report name from properties
    var originalReportName = report.ReportName;

    return StiNetCoreDesigner.SaveReportResult(this, "Some message after
    saving");
}
```

© 2003-2024 Stimulsoft

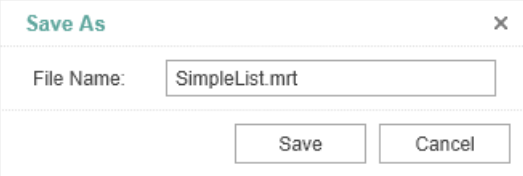
```
...
```

In this case, the dialog with the specified text will be displayed. The text can contain both an error message of saving or a warning, or any other message.



Saving reports and dashboards on the client side

To save the edited report on the client side as a file, no additional designer settings are required. It is enough to click the **Save As** main menu item. The dialog box will be displayed. In this dialog you can change the name of the report file. The file will be saved to the local disk of the computer.



The **HTML5 Designer** component provides the ability to change the behavior of the specified save option. For this purpose, the special **SaveReportAs** action is used in the designer. If you use this event, the report will be saved on the server side. Work of this event will be similar to the **SaveReport** action.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      SaveReportAs = "SaveReportAs"
   }
})
...
```

© 2003-2024 Stimulsoft

Index.cshtml.cs

```
...
public IActionResult OnPostSaveReportAs()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);

   // Save the report template
   // ...

   return StiNetCoreDesigner.SaveReportResult(this);
}
...
```

Use the following code to get the report name from the Save dialog.

Index.cshtml.cs

```
public IActionResult OnPostSaveReportAs()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);
   var requestParams = StiNetCoreDesigner.GetRequestParams(this);
   var reportName = requestParams.Designer.FileName;

   return StiNetCoreDesigner.SaveReportResult(this);
}
```

Saving settings

The report is saved in the background mode without reloading the page in the web browser window. If you need to visually control the process of saving the report, you should change the value of the **SaveReportMode** (or **SaveReportAsMode**) property of the designer to one of the three specified values - **Hidden** (default value), **Visible** or **NewWindow**.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Actions =
   {
      SaveReportAs = "SaveReportAs"
   },
   Behavior =
   {
      SaveReportAsMode = StiSaveMode.Visible
```

© 2003-2024 Stimulsoft

```
    }
})
...
```

If the **SaveReportMode** property is set to **Visible**, the report save action will be called in the current browser window in the normal (visible) mode using the POST request. If the **SaveReportMode** property is set to **NewWindow**, the report save event will be called in a new window of the web browser. By default, this property is set to **Hidden** - the report save event is called in the background using the AJAX request and is not displayed in the browser window. The same values and behavior are applicable to the **SaveReportAsMode** property.

6.2.8 Localization

The **HTML5 Designer** component supports the complete localization of its interface. Use the special **Localization** property to localize the report designer interface. As a value of this property, you should specify the path to the localization XML file (relative or absolute).

Index.cshtml

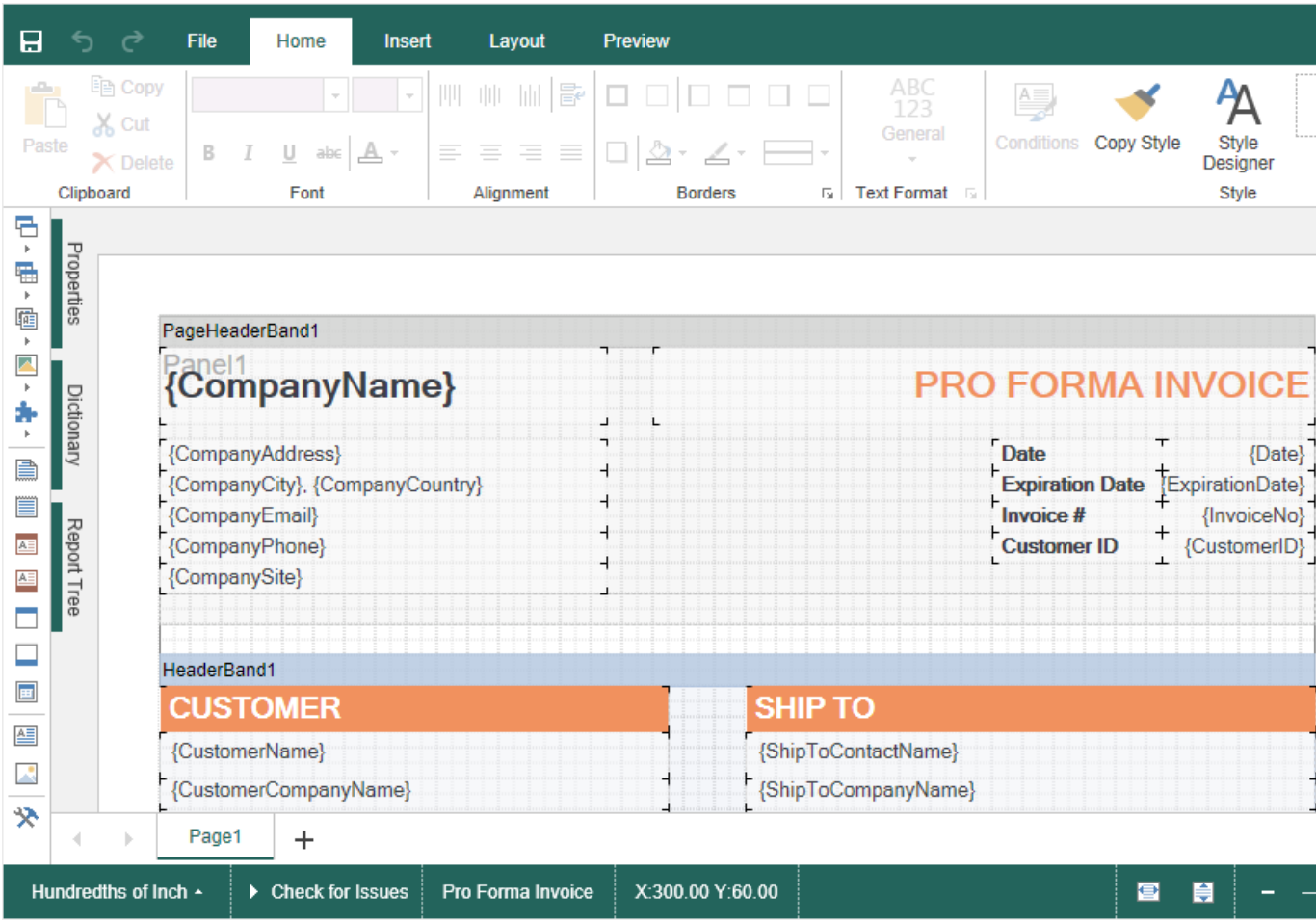
```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
    Localization = "Localization/en.xml"
})
...
```

The interface of the report designer allows you to select the necessary localization from an accessible list. To do this, set value for the **LocalizationDirectory** property as the folder in which the localization XML files are stored.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
    Localization = "Localization/en.xml",
    LocalizationDirectory = "Localization"
})
...
```

© 2003-2024 Stimulsoft



Information

If the value for the **Localization** property is set, then when you run the report designer, the localization specified in this property will always be applied. If the property value is not set, the localization selected from the list of available localizations in the report designer panel will be automatically loaded.

6.2.9 Using Themes

In the **HTML5 Designer** component you can change the appearance of visual controls. To change the theme, you should use the **Theme** property.

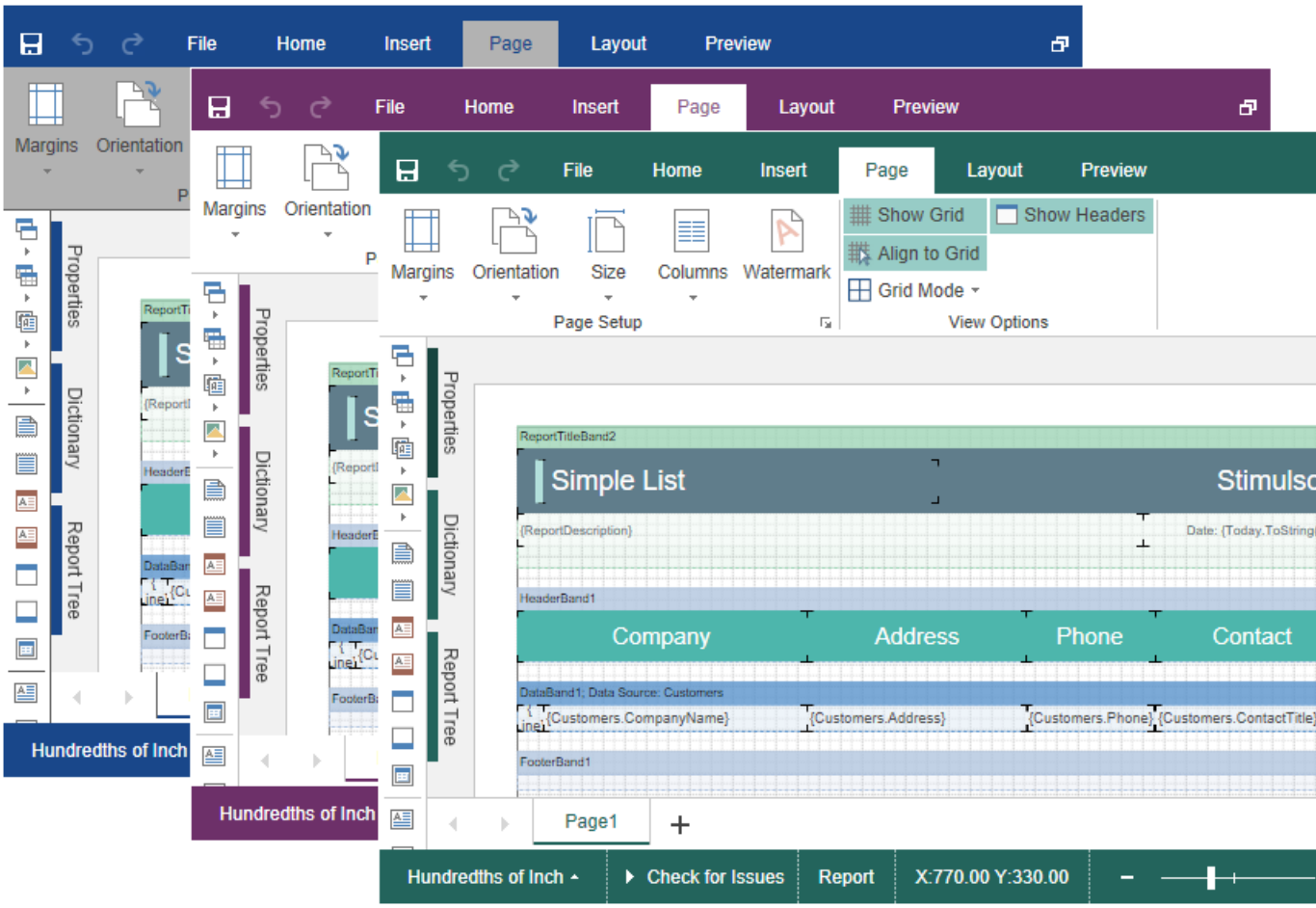
Default.aspx

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Theme = StiDesignerTheme.Office2022WhiteTeal
```

© 2003-2024 Stimulsoft

```
})
...
```

There are currently **2 themes** available with different color accents. As a result, **more than 50** variants of the appearance are available. This allows you to customize the appearance of the designer for almost any design of the Web project.



6.2.10 Caching

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

HTM5 Designer uses the server cache to store the editable report template. It is necessary because the client part of the designer contains only a visual representation of components of a report template. The report object itself with all the parameters and properties is stored on the server side.

You can manage caching with the following properties.

The CacheMode property

This property of the designer enables caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** - for caching, the current session is used, in which the report is saved as a packed string.

The CacheItemPriority property

This property sets the priority of the report stored in the cache of the server. It affects the automatic clearing of the server memory in case of memory shortage. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes you want to store the report in the server cache. If, when using caching, the requested report is not found in the cache (time of storing this report expired), then it will be requested again using the special **GetReport** action. In this case the unsaved changes may be lost.

© 2003-2024 Stimulsoft

The **HTML5 Designer** component provides the ability to specify its own methods for working with report caching. For this purpose, a special **StiCacheHelper** class is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods which respectively have the names - **GetObject, SaveObject** and **RemoveObject**.

Index.cshtml.cs

```
...
public class IndexModel : PageModel
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override object GetObject(string guid)
      {
         var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
         guid);
         if (File.Exists(path))
         {
            byte[] cacheData = System.IO.File.ReadAllBytes(path);
            return StiCacheHelper.GetObjectFromCacheData(cacheData);
         }
         return null;
      }

      public override void SaveObject(object obj, string guid)
      {
         byte[] cacheData = StiCacheHelper.GetCacheDataFromObject(obj);
         var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
         guid);
         File.WriteAllBytes(path, cacheData);
      }

      public override void RemoveObject(string guid)
      {
         var path = Path.Combine(HttpContext.Server.MapPath("CacheFiles"),
         guid);
         if (File.Exists(path))
            File.Delete(path);
      }
   }

   static IndexModel()
   {
      StiNetCoreDesigner.CacheHelper = new StiMyCacheHelper();
   }
}
...
```

© 2003-2024 Stimulsoft

To initialize the work with report caching using the created class, it is enough to set it as the value of the **StiNetCoreDesigner.CacheHelper** static property in the constructor of the controller.

6.2.11 Additional Methods

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

For **HTML5 Designer**, there are several additional methods that are used to get the object of the currently edited report, parameters of the current state of the designer and other useful data. These methods can be used in any actions of the designer.

The GetReportObject() Method

Returns the report object with which the designer is currently working. It is possible to perform the necessary actions with it - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the designer, specifying it as a parameter in the resulting action method.

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
{
   StiReport report = StiNetCoreDesigner.GetReportObject(this);
   report.ReportName = "MyReportName";

   return StiNetCoreDesigner.ExportReportResult(this, report);
}
...
```

The GetActionReportObject() method

Returns the report object that will be used for the particular action. For example, for the **OpenReport** action, this method returns a report loaded from the local disk of

© 2003-2024 Stimulsoft

the computer. For the **PreviewReport** action, the method returns a prepared copy of the report for preview.

Index.cshtml.cs

```
...
public IActionResult OnPostOpenReport()
{
   StiReport report = StiNetCoreDesigner.GetActionReportObject(this);

   // Register data for the opened report, if necessary
   DataSet data = new DataSet("Demo");
   data.ReadXml(StiNetCoreHelper.MapPath(this, "Data/Demo.xml"));
   report.RegData(data);
   report.Dictionary.Synchronize();

   return StiNetCoreDesigner.GetReportResult(this, report);
}
...
```

The GetFormValues() method

Returns the values of the form that initiated (opened by the POST request) a page of the designer. Thus, it is possible to get a collection of form parameters in any action of the designer.

Index.cshtml.cs

```
...
public IActionResult OnPostDesignerInteraction()
{
   NameValueCollection formValues = StiNetCoreDesigner.GetFormValues(this);

   return StiNetCoreDesigner.InteractionResult(this);
}
...
```

By default, this feature is disabled to optimize requests of the client-side of the designer to the server. To enable it, set the **PassFormValues** property to **true**.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Server =
   {
      PassFormValues = true
```

© 2003-2024 Stimulsoft

```
    }
})
...
```

The GetRequestParams() method

Returns all parameters of the current state of the designer passed to the server side. They can be useful for determining the type of action that the designer is currently executing - for example, to determine the type of export, as well as all action parameters.

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
{
   StiRequestParams requestParams =
   StiNetCoreDesigner.GetRequestParams(this);
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiNetCoreDesigner.GetReportObject(this);

      // Some action with report for the PDF export
      // ...

      return StiNetCoreDesigner.ExportReportResult(this, report);
   }

   return StiNetCoreDesigner.ExportReportResult(this);
}
...
```

The GetExportSettings() method

Returns all parameters of the current report export. The type of the parameter object will correspond to the type of export selected in the report preview menu. Any export parameters can be changed and passed to the input of the resulting method. In this case, the report will be exported with the parameters transferred.

Index.cshtml.cs

```
...
public IActionResult OnPostExportReport()
{
   StiExportSettings settings = StiNetCoreDesigner.GetExportSettings(this);
   if (settings.GetExportFormat() == StiExportFormat.Pdf)
```

© 2003-2024 Stimulsoft

```
    {
        StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
        pdfSettings.EmbeddedFonts = true;
        pdfSettings.AllowEditable = StiPdfAllowEditable.No;
        return StiNetCoreDesigner.ExportReportResult(this, settings);
    }

    return StiNetCoreDesigner.ExportReportResult(this);
}
...
```

The MapPath() and MapWebRootPath() methods

Returns the absolute path, respectively, to the application or wwwroot directory. You can use this to upload report templates files, data files, etc. These methods are located in the **StiNetCoreHelper** static class.

Index.cshtml.cs

```
...
public IActionResult OnPostGetReport()
{
    StiReport report = new StiReport();
    report.Load(StiNetCoreHelper.MapPath(this, "Reports/SimpleList.mrt"));

    return StiNetCoreDesigner.GetReportResult(this, report);
}
...
```

6.2.12 Timeout

When working with the **StiNetCoreDesigner** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
    Server =
    {
```

© 2003-2024 Stimulsoft

```
    CacheTimeout = 10
  }
})
...
```

Using cache will increase the speed of the report designer. See the chapter Caching for more information

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
  Server =
  {
    RequestTimeout = 30
  }
})
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection, and data sources in the report.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
  Actions =
  {
    GetReport = "GetReport",
    DesignerEvent = "DesignerEvent"
  }
})
...
```

© 2003-2024 Stimulsoft

Index.cshtml.cs

```
...
public IActionResult OnGetDesignerEvent()
{
   return StiNetCoreDesigner.DesignerEventResult(this);
}

public IActionResult OnPostDesignerEvent()
{
   return StiNetCoreDesigner.DesignerEventResult(this);
}
...
```

6.2.13 Add custom functions

Information

See on GitHub example of adding a custom function in the ASP.NET MVC HTML5 Designer component.

You can add a custom function to the Dictionary in the report designer when you integrate it into your application. After adding the custom function, you can use this in creating reports and dashboards. Below is the example of adding a function for calculating the sum total.

Index.cshtml.cs

```
...
public static decimal MySum(object value)
{
   if (!ListExt.IsList(value))
   return Stimulsoft.Base.Helpers.StiValueHelper.TryToDecimal(value);

   return Stimulsoft.Data.Functions.Funcs.SkipNulls(ListExt.ToList(value))
   .TryCastToDecimal()
   .Sum();
}
...
static IndexModel()
{
   StiFunctions.AddFunction("MyCategory", "MySum",
   "description", typeof(DesignerController),
   typeof(decimal), "Calculates a sum of the specified set of values.",
   new[] { typeof(object) },
   new[] { "values" },
   new[] { "A set of values" }).UseFullPath = false;
```

© 2003-2024 Stimulsoft

```
}
...
```

6.2.14 Settings

The **HTML5 Designer** configuration is done using properties that are located in the **StiNetCoreDesignerOptions** class. All properties are divided into groups, some of the groups contain subgroups of properties for ease of use. Below is an example of setting some properties of the designer.

Index.cshtml

```
...
@Html.StiNetCoreDesigner(new StiNetCoreDesignerOptions() {
   Theme = Stimulsoft.Report.Web.StiDesignerTheme.Office2022WhiteTeal,
   Localization = "Localization/en.xml",
   Actions =
   {
      GetReport = "GetReport",
      PreviewReport = "PreviewReport",
      SaveReport = "SaveReport",
      DesignerEvent = "DesignerEvent"
   },
   Appearance =
   {
      InterfaceType = StiInterfaceType.Auto,
      ShowTooltipsHelp = false,
      ShowDialogsHelp = false,
      DefaultUnit = Stimulsoft.Report.StiReportUnitType.Centimeters
   },
   Dictionary =
   {
      PermissionBusinessObjects =
      Stimulsoft.Report.Web.StiDesignerPermissions.None,
      PermissionDataConnections =
      Stimulsoft.Report.Web.StiDesignerPermissions.View
   },
   Bands =
   {
      ShowChildBand = false,
      ShowEmptyBand = false,
      ShowOverlayBand = false
   }
})
...
```

Basic settings (without groups)

Name	Description
Theme	Specifies the theme of the report designer. The

© 2003-2024 Stimulsoft

	list of available themes is located in the **StiDesignerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Specifies the path to the XML localization file. The path can be absolute or relative. By default, English localization is used. It is built into the designer and does not require additional XML files.
LocalizationDirectory	Specifies the path to the directory with XML localization files. The localization files located in the specified folder will be loaded to the localization list in the designer panel.
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the component is expanded to the entire area of the browser window.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the component is expanded to the entire area of the browser window.

Actions

Name	Description
GetReport	Specifies the name of the action method to prepare the report template when loading the designer.
OpenReport	Specifies the name of the action method to open the report template from the designer menu.

© 2003-2024 Stimulsoft

CreateReport	Specifies the name of the action method to prepare the report template when creating the new report in the designer.
SaveReport	Specifies the name of the action method to save the report template on the server side..
SaveReportAs	Specifies the name of the action method to store the report template on the server side when using the **Save As** menu item. If no action is specified, the built-in method of saving the report template to the local disk will be used.
PreviewReport	Specifies the name of the action method to prepare the rendered report in the preview window.
GetPreviewReport	Specifies the name of the action method just before a report is displayed in the preview window.
ExportReport	Specifies the name of the action method to export reports to the specified format.
Exit	Specifies the name of the action method to go to the desired view by clicking the Exit button in the main menu of the report designer.
DesignerEvent	Specifies the name of the action method of the report designer to handle additional designer actions, such as working with data, report components, and others. Also, this action is used to load scripts and designer styles.

Server

Name	Description
RouteTemplate	Sets the route template that is returned when the report designer actions are executed. If the property is not set, then the Razor project template will be used instead. The default value of the property is null.

© 2003-2024 Stimulsoft

ShowServerErrorPage	Enables the display of an HTML page with error details that occurred on the server side. The error details will be displayed in the preview window if the property is enabled. If the property is disabled, the numeric error code and a short error description will be displayed in the dialog window. By default, the property is set to **true**.
AllowAutoUpdateCookies	Allows the designer to update the cookies automatically on every request to the server. By default, cookies are set when creating the designer, if they are not specified in the report. By default, the property is set to **false**.
AllowAntiforgeryToken	Allow the designer to automatically request and send the antiforgery token. By default, the property is set to **true**.
RequestTimeout	Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled in **HTML5 Designer**; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed

© 2003-2024 Stimulsoft

	string; › **StringSession** – the session is used as a repository, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or server session will be automatically re-saved after a certain period of time if the designer is idle (about every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the designer mode in which relative URLs are used for requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value which specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the designer. If **false**, only the necessary parameters are used to request the resources of the designer. This corresponds to the more correct operation of the browser cache. By default, the property is set to **true**.
PassFormValues	Enables transferring the POST form values to the client side, if these values are to be used in the actions of the designer. If you enable this feature, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
UseCompression	Enables compression of designer requests in the GZip stream. This allows you to reduce the

© 2003-2024 Stimulsoft

	amount of Internet traffic, but slows down the designer. By default, the property is **false**.
UseCacheForResources	Enables caching of the component resources on the server side. The following resources are supported: scripts, styles and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Specifies the path to the CSS file of styles for the report designer. If this property is set, the standard styles of the selected theme will not be loaded. The default value is an empty value.
DefaultUnit	Sets the units for the size of the report and all its components. By default, centimeters are used.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the **StiZoomMode** enumeration: › **PageWidth** – when the designer runs, the zoom, necessary to display the report by the page width, will be set; › **PageHeight** – when the designer runs, the zoom, required to display the page height of the report, will be set.
ShowAnimation	Enables animation for various elements of the

© 2003-2024 Stimulsoft

	designer interface. By default, the property is set to **true**.
ShowOpenDialog	Allows to display the open dialog, or to open with the open event. By default, the property is set to **true**.
ShowTooltips	Enables displaying tooltips for designer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enable displaying links to online documentation in tooltips for designer controls. By default, the property is set to **true**.
ShowDialogsHelp	Enables displaying links to online documentation on the titles of dialog forms of the designer. By default, the property is set to **true**.
InterfaceType	Sets the type of interface used for the designer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the interface type of the designer will be selected automatically depending on the device used (default value); › **Mouse** – forced use of the interface to control the designer with the mouse; › **Touch** – forced use of the Touch interface to control the designer via the touch screen (mobile devices), also in this mode, the interface elements are increased.
DatePickerFirstDayOfWeek	Sets the first day of the week for the select date item. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Auto** – automatic detection of the first day of the week from the browser settings (default value); › **Monday** – the first day of the week is Monday;

© 2003-2024 Stimulsoft

	› **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
ShowReportTree	Enables displaying the tree of report components. By default, the property is set to **true**.
ChartRenderType	Gets or sets the type of the chart in the preview. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages in the preview tab. Can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
CloseDesignerWithoutAsking	Sets a value which indicates that the designer

© 2003-2024 Stimulsoft

	will be closed without asking. By default, the property is set to **true**.
ShowSystemFonts	Sets a visibility of the system fonts in the fonts list. By default, the property is set to **true**.
WizardTypeRunningAfterLoad	Calls the Report wizard after starting the report designer. It may have one of the following **StiWizardType** enumeration values: › **None** - runs the report designer without running the report wizard; › **StandardReport** - runs the Standard wizard; › **MasterDetailReport** - runs the Master-Detail wizard; › **LabelReport** - runs the Label wizard; › **InvoicesReport** - runs the Invoice wizard; › **OrdersReport** - runs the Order wizard; › **QuotationReport** - runs the Quote wizard.
AllowWordWrapTextEditors	Allows word wrap in the text editors. By default, the property is set to **true**.

Behavior

Name	Description
ShowSaveDialog	Enables displaying the dialog to insert a report name when it is saved. The name of the report will be transferred in the parameters of the report designer. By default, the property is set to **true**.
UndoMaxLevel	Sets the maximum number to cancel actions with the report (the Undo/Redo function). A big value of this property will consume memory on the server side to store the undo parameters. The default value is **6**.
AllowChangeWindowTitle	Allows using the title of the browser window to display the file name of the edited report. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

SaveReportMode	Sets the mode to save the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
SaveReportAsMode	Sets the mode for saving the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
CheckReportBeforePreview	Sets the value that allows running the report checker before preview.

FileMenu

Name	Description
Visible	Enables displaying the main menu of the report designer. By default, the property is set to **true**.
ShowNew	Enables showing the main menu item - **New**. By default, the property is set to **true**.
ShowFileMenuNewReport	Sets a visibility of the new report button in the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowFileMenuNewDashboard	Sets a visibility of the new dashboard button in the designer. By default, the property is set to **true**.
ShowOpen	Enables showing the main menu item - **Open**. By default, the property is set to **true**.
ShowSave	Enables showing the main menu item - **Save**. By default, the property is set to **true**.
ShowSaveAs	Enables showing the main menu item - **Save As**. By default, the property is set to **true**.
ShowClose	Enables showing the main menu item - **Close**. By default, the property is set to **true**.
ShowExit	Enables showing the main menu item - **Exit**. By default, the property is set to **true**.
ShowReportSetup	Enables showing the main menu item - **Report Setup**. By default, the property is set to **true**.
ShowOptions	Enables showing the main menu item - **Options**. By default, the property is set to **true**.
ShowInfo	Enables showing the main menu item - **Info**. By default, the property is set to **true**.
ShowAbout	Enables showing the main menu item - **About**. By default, the property is set to **true**.
ShowHelp	Enables showing the main menu item - **Help**. By default, the property is set to **true**.

Dictionary

Name	Description
Visible	Enables showing the data dictionary of the report. By default, the property is set to **true**.
UseAliases	Allows you to use aliases in the data dictionary. It has the three values of the **StiUseAliases** enumeration:

© 2003-2024 Stimulsoft

	› **Auto** - defines the mode of using aliases from a saved value in cookies (default value); › **True** - sets the mode of using aliases in the data dictionary; › **False** - disables the mode of using aliases in the data dictionary.
NewReportDictionary	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It has the three values of the **StiNewReportDictionary** enumeration: › **Auto** - defines the mode to create or join the data dictionary from a saved value in cookies (default value); › **DictionaryNew** - sets the mode to create a new data dictionary when creating a new report; › **DictionaryMerge** - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
ShowDictionaryContextMenuProperties	Sets a visibility of the **Properties** item in the dictionary context menu. By default, the property is set to **true**.
ShowDictionaryActions	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.
PermissionDataConnections	Sets the available actions to connect data to the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataSources	Sets available actions on report data sources. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataColumns	Sets the available actions on data columns in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionBusinessObjects	Sets the available actions on the business objects in the report. It can take one or more values from the **StiDesignerPermissions**

© 2003-2024 Stimulsoft

	enumeration.
PermissionDataRelations	Sets the available actions to linking data in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionVariables	Sets available actions on report variables. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionResources	Sets the available actions for the resources in the Report Dictionary. Takes one or several values from the **StiDesignerPermissions** enumeration.
PermissionSqlParameters	Sets the available actions for the parameters of the SQL queries for the Report DataSources. Takes one or several values from **StiDesignerPermissions** enumeration.
DataTransformationsPermissions	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

The table below shows all available values for the **StiDesignerPermissions** enumeration, which can be set for the dictionary elements of the report.

Value	Description
None	Disables any action on the item of the data dictionary.
All	Allows any action on the item of the data dictionary.
Create	Allows creating a specific data dictionary item.
Delete	Allows deleting a specific data dictionary item.
Modify	Allows modifying a specific data dictionary item.
View	Allows viewing a specific data dictionary item.
ModifyView	Allows modifying and viewing a specific data dictionary item.

© 2003-2024 Stimulsoft

Toolbar

Name	Description
ShowToolbar	Enables displaying the toolbar of the designer. By default, the property is set to **true**.
ShowSetupToolboxButton	Enables displaying the button to call the dialog box with settings for the side toolbar. By default, the property is set to **true**.
ShowInsertButton	Enables displaying the **Insert** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowLayoutButton	Enables displaying the tab **Layout** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPageButton	Enables displaying the tab **Page** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPreviewButton	Enables displaying the tab **Preview** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the designer. By default, the property is set to **true**.
ShowAboutButton	Enables displaying the **About** on the toolbar of the designer. By default, the property is set to **false**.

PropertiesGrid

Name	Description
Visible	Enables displaying the property panel. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

Width	Sets the width of the property panel. By default, the width is set to **370 px**.
LabelWidth	Specifies the width of the labels on the properties panel. By default, the width is set to **160 px**.
PropertiesGridPosition	Sets **Left** or **Right** position of the properties grid in the designer. It has the three values of the **StiPropertiesGridPosition** enumeration: › **Left**; › **Right**.
ShowPropertiesWhichUsedFromStyles	Sets a visibility of the properties which used from styles in the designer. By default, the property is set to **false**.

Components

Name	Description
ShowText	Enables displaying the **Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowTextInCells	Enables displaying the **Text in Cells** component in the insert menu for report components. By default, the property is set to **true**.
ShowRichText	Enables displaying the **Rich Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowImage	Enables displaying the **Image** component in the insert menu for report components. By default, the property is set to **true**.
ShowBarCode	Enables displaying the **Bar Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowShape	Enables displaying the **Shape** component in the insert menu for report components. By default,

© 2003-2024 Stimulsoft

	the property is set to **true**.
ShowHorizontalLinePrimitive	Enables displaying the **Horizontal Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowVerticalLinePrimitive	Enables displaying the **Vertical Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowRectanglePrimitive	Enables displaying the **Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowRoundedRectanglePrimitive	Enables displaying the **Rounded Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowPanel	Enables displaying the **Panel** component in the insert menu for report components. By default, the property is set to **true**.
ShowClone	Enables displaying the **Clone** component in the insert menu for report components. By default, the property is set to **true**.
ShowCheckBox	Enables displaying the **Check Box** component in the insert menu for report components. By default, the property is set to **true**.
ShowSubReport	Enables displaying the **Sub Report** component in the insert menu for report components. By default, the property is set to **true**.
ShowZipCode	Enables displaying the **Zip Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowTable	Enables displaying the **Table** component in the insert menu for report components. By default, the property is set to **true**.
ShowCrossTab	Enables displaying the **Cross-Tab** component in the insert menu for report components. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.
ShowChart	Enables displaying the **Chart** component in the insert menu for report components. It affects on all chart types. By default, the property is set to **true**.
ShowMap	Enables displaying the **Map** component in the insert menu for report components. By default, the property is set to **false**.
ShowGauge	Enables displaying the **Gauge** component in the insert menu for report components. By default, the property is set to **false**.
ShowSparkline	Enables displaying the **Sparkline** component in the insert menu for report components. By default, the property is set to **true**.
ShowMathFormula	Enables displaying the **Math Formula** component in the insert menu for report components. By default, the property is set to **true**.
ShowElectronicSignature	Enables displaying the **Electronic Signature** component in the insert menu for report components. By default, the property is set to **true**.
ShowPdfDigitalSignature	Enables displaying the **PDF Digital Signature** component in the insert menu for report components. By default, the property is set to **true**.

Bands

Name	Description
ShowReportTitleBands	Enables displaying the **Report Title** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowReportSummaryBand	Enables displaying the **Report Summary** item in

© 2003-2024 Stimulsoft

	the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageHeaderBand	Enables displaying the **Page Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageFooterBand	Enables displaying the **Page Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupHeaderBand	Enables displaying the **Group Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupFooterBand	Enables displaying the **Group Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHeaderBand	Enables displaying the **Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowFooterBand	Enables displaying the **Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnHeaderBand	Enables displaying the **Column Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnFooterBand	Enables displaying the **Column Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowDataBand	Enables displaying the **Data** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHierarchicalBand	Enables displaying the **Hierarchical** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowChildBand	Enables displaying the **Child** item in the **Bands** menu of the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowEmptyBand	Enables displaying the **Empty** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowOverlayBand	Enables displaying the **Overlay** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowTableOfContents	Enables displaying the **Table of Contents** item in the **Bands** menu of the designer. By default, the property is set to **true**.

DashboardElements

Name	Description
ShowTableElement	Enables displaying the **Table** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowCardsElement	Enables displaying the **Cards** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowChartElement	Enables displaying the **Chart** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowGaugeElement	Enables displaying the **Gauge** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPivotTableElement	Enables displaying the **Pivot** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowIndicatorElement	Enables displaying the **Indicator** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowProgressElement	Enables displaying the **Progress** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowRegionMapElement	Enables displaying the **Region Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowOnlineMapElement	Enables displaying the **Online Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowImageElement	Enables displaying the **Image** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTextElement	Enables displaying the **Text** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPanelElement	Enables displaying the **Panel** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowShapeElement	Enables displaying the **Shape** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowButtonElement	Enables displaying the **Button** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowListBoxElement	Enables displaying the **List Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowComboBoxElement	Enables displaying the **Combo Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewElement	Enables displaying the **Tree View** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewBoxElement	Enables displaying the **Tree View Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowDatePickerElement	Enables displaying the **Date Picker** element in the Dashboard Elements menu of the designer.

© 2003-2024 Stimulsoft

	By default, the property is set to **true**.

CrossBands

Name	Description
ShowCrossGroupHeaderBand	Enables displaying the **Cross Group Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossGroupFooterBand	Enables displaying the **Cross Group Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossHeaderBand	Enables displaying the **Cross Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossFooterBand	Enables displaying the **Cross Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossDataBand	Enables displaying the **Cross Data** section in the insert menu for report components. By default, the property is set to **true**.

Dashboards

Name	Description
ShowNewDashboardButton	Sets a visibility of the **New Dashboard** button in the designer. By default, the property is set to **true**.

Pages

Name	Description

© 2003-2024 Stimulsoft

ShowNewPageButton	Sets a visibility of the **New Page** button in the designer. By default, the property is set to **true**.

When designing a report or dashboard in the report designer, you can also define **ExportOptions**, **EmailOptions**, and **PreviewToolbarOptions** on the **Preview** tab. These options are similar to the report viewer options.

7 Reports and Dashboards for Blazor

Blazor is a cross-platform technology for creating Web applications for Windows, Linux, and macOS. We offer tools for creating, displaying, converting reports using this technology. Two options of this technology are supported - **Blazor Server**, a distributed system, where all logic is executed on the server-side, and the visual part is displayed on the client; **Blazor WebAssembly**, where all product modules are loaded and run directly in the Web browser window.

Tools for creating and editing reports:	Tools for viewing and converting reports:
Blazor Designer	Blazor Viewer

7.1 Viewer

Samples

Get acquainted with the examples of working with the **Blazor Viewer** component. The Blazor Server and the Blazor WebAssembly technologies are available on GitHub. All the samples are separate projects, grouped into one solution for Visual Studio.

The **Blazor Viewer** (**StiBlazorViewer**) component is intended for report viewing in the browser window. At the same time, you don't need to install components or some special plugins on the client. All you need is a modern Web browser. With the help of the **Blazor Viewer**, you can view, print, export reports on any computer with any installed operating system.

© 2003-2024 Stimulsoft

The **Blazor Viewer** component is developed as a universal component, and it can work both with the use of the **Blazor Server** and the use of the **WebAssembly** technology. When making all interactive actions on reports, the component requests only the necessary data; it allows to get rid of reloading the entire page, save Web traffic, and increase work speed.

The **Blazor Viewer** supports many design themes, animated interface, bookmarks, interactive reports, editing report elements on the page, full-screen mode, search, and other necessary report viewing features.

To use the **Blazor Viewer** in Web-project, you should install the Nuget package Stimulsoft.Reports.Blazor or Stimulsoft.Dashboards.Blazor:

- Select the «Manage Nuget Packages...» in the context project menu.
- Specify the Stimulsoft.Reports.Blazor, in the search line on the Browse tab;
- Select the element, define the version of a package, and click on the **Install**. When updating a package, you should click on the **Update**.

If for some reason it is not possible, you should add the following assemblies to the project:
Stimulsoft.Base.dll
Stimulsoft.Blockly.dll
Stimulsoft.Data.dll
Stimulsoft.Drawing.dll
Stimulsoft.Map.dll
Stimulsoft.Report.dll
Stimulsoft.Report.Blazor.dll
Stimulsoft.Report.Check.dll
Stimulsoft.Report.Helper.dll
Stimulsoft.Report.Web.dll
Stimulsoft.Report.WebDesign.dll
Stimulsoft.System.dll
Stimulsoft.System.Web.dll

You should additionally add the following assemblies to the project for using dashboards:
Stimulsoft.Dashboard.dll
Stimulsoft.Dashboard.Drawing.dll
Stimulsoft.Dashboard.Export.dll

© 2003-2024 Stimulsoft

- Activation
- Dynamic Collapsing, Sorting, Drill-Down
- Showing Report
- Work with Bookmarks
- Connecting Data
- Editing Report
- Localization
- Sending Report by Email
- Printing Reports
- Export and Printing from Code
- Exporting Reports
- Basic Features
- Vieweing Modes
- Viewer Events
- Calling Designer from Viewer
- Settings
- Work with parameters

7.1.1 Activation

After acquiring a Stimulsoft product, you should activate the license for used components. You can do it by specifying your license key or loading a file with your license key. Below is an example of the **StiBlazorViewer** component activation.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer />

@code
{
   protected override void OnInitialized()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
      Stimulsoft.Base.StiLicense.LoadFromFile("Content/license.key");

      base.OnInitialized();
   }
}
```

© 2003-2024 Stimulsoft

You can get a license key or download a file with a license key in the user's account. To authorize in the user's personal account, you should use a username and password specified when buying a product.

7.1.2 Showing Reports

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

To display a report, you should add the **StiBlazorViewer** component on the Razor page, add the **StiReport** object, and assign it to the viewer using the Report property. After a report is loaded from a file, it will be displayed in the viewer automatically.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
  //Report object to use in viewer
  private StiReport Report;

  protected override void OnInitialized()
  {
    base.OnInitialized();

    //Create empty report object
    var report = new StiReport();

    //Load report template
    report.Load("Reports/TwoSimpleLists.mrt");

    //Assing report object to viewer
    Report = report;
  }
```

© 2003-2024 Stimulsoft

```
}
```

If a report is not rendered before display, the **BlazorViewer** component will render it automatically. This way, to display a report, you can use various types of reports - report templates and already generated reports.

Loading fonts

The Blazor does not have access to the fonts installed on a computer, so to render a report correctly, you should specify the fonts you used in your report. You can do it with the help of the statistic **StiFontCollection** class by either specifying a file or the Base64 string, which has a necessary font.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
   //Report object to use in viewer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init base font as a file
      Stimulsoft.Base.StiFontCollection.AddFontFile("Fonts/Microsoft Sans
      Serif.ttf", "Segoe UI");

      //Init base font as a Base64 string
      var fontBase64 = "AAEAAAAVAQAABABQRFNJR/W/YxAAA814A...";
      Stimulsoft.Base.StiFontCollection.AddFontBase64(fontBase64, "Segoe
      UI");

      //Create empty report object
      report = new StiReport();

      //Load report template
      report.Load("Reports/TwoSimpleLists.mrt");

      //Assing report object to viewer
      Report = report;
   }
}
```

© 2003-2024 Stimulsoft

7.1.3 Connecting Data

Data for report rendering can be connected in various ways. The easiest way is to keep connection settings in a report template. In addition, data can be connected from a code. You can do it before a report is assigned to the viewer.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
   //Report object to use in viewer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Load new data from XML file
      var data = new System.Data.DataSet();
      data.ReadXml("Data/Demo.xml");

      //Create and load report template
      var report = new StiReport();
      report.Load("Reports/TwoSimpleLists.mrt");
      report.Dictionary.Databases.Clear();
      report.RegData("Demo", data);

      //Assing report object to viewer
      Report = report;
   }
}
```

Information

At this moment, the SQL data resources are supported only for Blazor Server components.

SQL data sources

The connection parameters to the SQL data source and any other ones can be stored in the report template. Suppose you want to set the connection parameters

© 2003-2024 Stimulsoft

from the code before rendering the report (for example, for security reasons or depending on the authorized user). In that case, you can use the example below.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
   //Report object to use in viewer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      OracleConnection connection = new OracleConnection("Data
      Source=Oracle8i;Integrated Security=yes");
      connection.Open();

      OracleDataAdapter adapter = new OracleDataAdapter();
      adapter.SelectCommand = new OracleCommand("SELECT * FROM Products",
      connection);

      var dataSet = new DataSet("productsDataSet");
      adapter.Fill(dataSet, "Products");

      var report = new StiReport();
      report.Load("Reports/TwoSimpleLists.mrt");
      report.Dictionary.Databases.Clear();
      report.RegData("Products", dataSet);

      //Assing report object to viewer
      Report = report;
   }
}
```

Also, for SQL data sources used in the report, you can specify the Query Timeout in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection and data sources in the report.

Index.razor

© 2003-2024 Stimulsoft

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
   //Report object to use in viewer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      var report = new StiReport();
      report.Load("Reports/Report.mrt");
      ((StiSqlDatabase)
      report.Dictionary.Databases["Connection"]).ConnectionString = @"Data
      Source=server;Integrated Security=True;Initial Catalog=DataBase";
      ((StiSqlSource)
      report.Dictionary.DataSources["DataSourceName"]).SqlCommand = "select
      * from Table where Column = 100";
      ((StiSqlSource)
      report.Dictionary.DataSources["DataSourceName"]).CommandTimeout =
      1000;

      //Assing report object to viewer
      Report = report;
   }
}
```

Information

For SQL data sources of other types, the connection is created similarly, and an adapter corresponding to the data source type is connected. For example, for the MS SQL data source, you should connect SqlDataAdapter. For OracleDataAdapter is required for Oracle. Also, you should specify a connection string that matches the connection type.

You can also use data for designing reports and dashboards obtained from OData storage. In this case, you can do the authorization using a username, user password, or using a token. Authorization parameters are specified in the connection string to the OData storage using the ";" separator.

Index.razor

© 2003-2024 Stimulsoft

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
   //Report object to use in viewer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      var report = new StiReport();

      //Authorization using a user account
      var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
      services.odata.org/V4/Northwind/
      Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Passwor
      d;Client_Id=Your Client ID", false, null);

      //Authorization using a user token
      var oDataDatabase = new StiODataDatabase("OData", "OData", @"https://
      services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your
      token", false, null);

      report.Dictionary.Databases.Add(oDataDatabase);
      oDataDatabase.Synchronize(report);

      //Query with data filter
      ((StiSqlSource)report.Dictionary.DataSources["Products"]).SqlCommand
      = "Products?$filter=ProductID eq 2";

      //Assing report object to viewer
      Report = report;
   }
}
```

The table below shows the connection string templates for different types of data sources.

Data Source	Connection String Template
MS SQL	Integrated Security=False; Data Source=myServerAddress;Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
MySQL	Server=myServerAddress;

© 2003-2024 Stimulsoft

	Database=myDataBase;UserId=myUsername; Pwd=myPassword;
ODBC	Driver={SQL Server}; Server=myServerAddress;Database=myDataBase ; Uid=myUsername; Pwd=myPassword;
OLE DB	Provider=SQLOLEDB.1; Integrated Security=SSPI;Persist Security Info=False; Initial Catalog=myDataBase;Data Source=myServerAddress
Oracle	Data Source=TORCL;User Id=myUsername;Password=myPassword;
MS Access	Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Password=pass;Data Source=C:\ \myAccessFile.accdb;
PostgreSQL	Server=myServerAddress; Port=5432; Database=myDataBase;User Id=myUsername; Password=myPassword;
Firebird	User=SYSDBA; Password=masterkey; Database=SampleDatabase.fdb;DataSource=my ServerAddress; Port=3050; Dialect=3; Charset=NONE;Role=; Connection lifetime=15; Pooling=true; MinPoolSize=0;MaxPoolSize=50; Packet Size=8192; ServerType=0;
SQL CE	Data Source=c:\MyData.sdf; Persist Security Info=False;
SQLite	Data Source=c:\mydb.db; Version=3;
DB2	Server=myAddress:myPortNumber;Database=m yDataBase;UID=myUsername;PWD=myPassword; Max Pool Size=100;Min Pool Size=10;
Infomix	Database=myDataBase;Host=192.168.10.10;Serv er=db_engine_tcp;Service=1492;Protocol=onsoc tcp;UID=myUsername;Password=myPassword;
Sybase	Data Source=myASEserver;Port=5000;Database=myD ataBase;Uid=myUsername;Pwd=myPassword;

© 2003-2024 Stimulsoft

Teradata	Data Source=myServerAddress;User ID=myUsername;Password=myPassword;
VistaDB	Data Source=D:\folder \myVistaDatabaseFile.vdb4;Open Mode=ExclusiveReadWrite;
Universal(dotConnect)	Provider=Oracle;direct=true;data source=192.168.0.1;port=1521;sid=sid;user=user; password=pass
MongoDB	mongodb://<user>:<password>@localhost/test
OData	http://services.odata.org/v3/odata/OData.svc/
Other...	The table shows the most commonly used templates for the connection string. You can view various connection string options at the special website.

Data from XML, JSON, Excel files

You can keep connections to the XML and the JSON data resources in a report template. If you need to specify data files from a code, you can use the following example.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Load new data from XML file
      var dataSet = new System.Data.DataSet();
      dataSet.ReadXml("Data/Demo.xml");

      var report = new StiReport();
      report.Load("Reports/SimpleList.mrt");
```

© 2003-2024 Stimulsoft

```
        //Register data for the report
        report.RegData("Demo", dataSet);

        //Assing report object to viewer
        Report = report;
    }
}
```

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Report="@Report" />

@code
{
    private StiReport Report;

    protected override void OnInitialized()
    {
        base.OnInitialized();

        //Load new data from JSON file
        var dataSet = StiJsonToDataSetConverterV2.GetDataSetFromFile("Data/
        Demo.json");

        var report = new StiReport();
        report.Load("Reports/SimpleList.mrt");

        //Register data for the report
        report.RegData(dataSet);

        //Assing report object to viewer
        Report = report;
    }
}
```

Information

There is an option to retrieve data from an Excel file in the viewer. To do this you can use the following method:

```
var dataSet = StiExcelConnector.Get().GetDataSet(new StiExcelOptions(a
```

© 2003-2024 Stimulsoft

7.1.4 Localization

The Blazor Viewer component supports the localization of its interface. To localize the report viewer interface to the language you need, use the special **Localization** property. You should specify the path to the localization XML file (relative or absolute) as the value of this property (relative or absolute).

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Localization="Localization/en.xml" />
```

When loading the report viewer, the localization file will be loaded automatically.

7.1.5 Using Themes

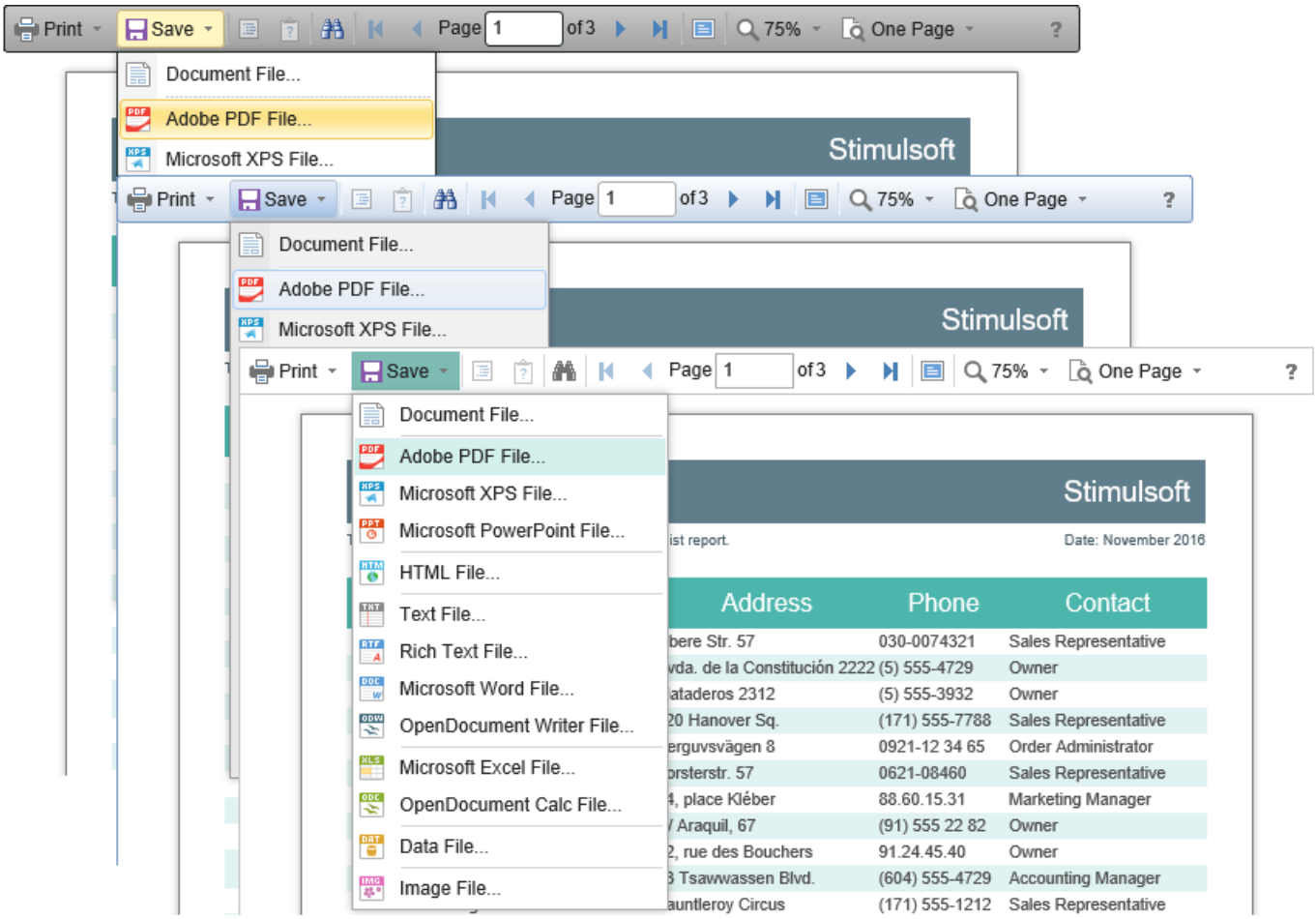
The **Blazor Viewer** component has an option to change the design themes of controls. The **Theme** property in the component options is used to change a theme. It may have one of the values of the **StiViewerTheme** enumeration.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Theme="StiViewerTheme.Office2022WhiteCarmine" />
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.

© 2003-2024 Stimulsoft



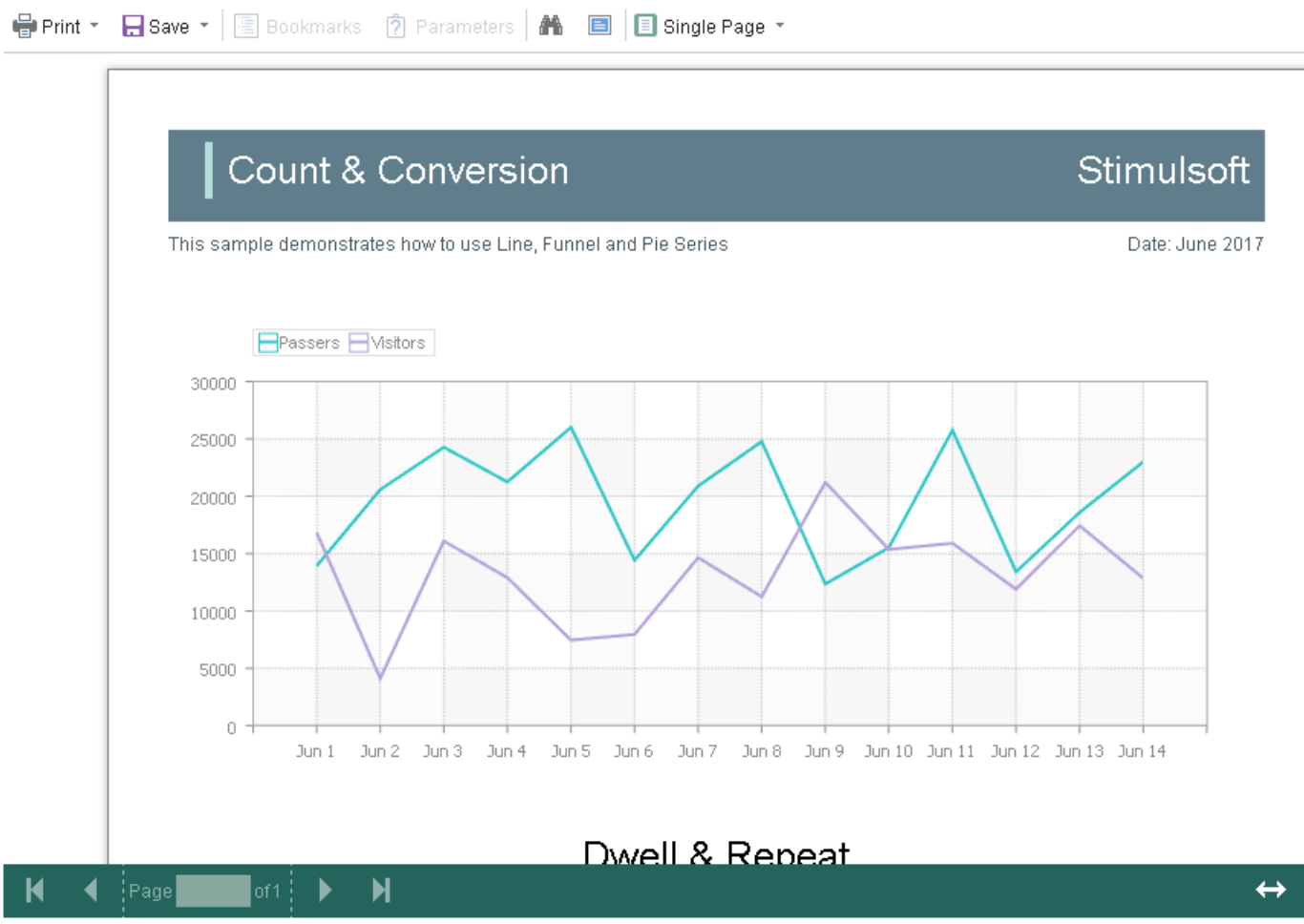
By default, the Viewer displays only the top toolbar, where there are all controls. If it's necessary, you can divide the toolbar into the top and the lower. The menu of printing and exporting reports will be on the top panel, also the buttons to work with parameters and bookmarks. The lower toolbar will contain elements to switch between report pages and the zoom control menu. The **DisplayMode** property is intended for enabling the specified mode, which can have the **Simple** value mode and the **Separated** mode.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
```

© 2003-2024 Stimulsoft

```
    //Options object
    private StiBlazorViewerOptions Options;

    protected override void OnInitialized()
    {
        base.OnInitialized();

        //Init options object
        Options = new StiBlazorViewerOptions();
        Options.Appearance.ScrollbarsMode = true;
        Options.Toolbar.DisplayMode = StiToolbarDisplayMode.Separated;
    }
}
```



Additionally, you may set the parameters of the main viewer elements. For example, you can change the font and color of the Viewer toolbar titles, set the background of the Viewer, color, etc. Below is the list of available properties, which change the Viewer design and their value by default.

© 2003-2024 Stimulsoft

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web
@using System.Drawing

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Appearance.BackgroundColor = Color.White;
      Options.Appearance.PageBorderColor = Color.Blue;
      Options.Appearance.ShowPageShadow = true;
      Options.Toolbar.BackgroundColor = Color.White;
      Options.Toolbar.BorderColor = Color.Gray;
      Options.Toolbar.FontColor = Color.Black;
      Options.Toolbar.FontFamily = "Arial";
   }
}
```

7.1.6 Basic Features

The main options of the **Blazor Viewer** contain the following operations: report display, switching between report pages, zoom changing and report display mode. All specified operations are performed without reloading browser page. You don`t need to setting any special options or events to perform them.

The report viewer has a special event the **OnViewerEvent**, which will be invoked after any action of the viewer. At this event you can find out the type of action, which the viewer is performing at the moment and get all parameters of the viewer, transferred to the server side.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnViewerEvent="@OnViewerEvent" />

@code
{
   private void OnViewerEvent(StiReportDataEventArgs args)
```

© 2003-2024 Stimulsoft

```
    {
        var action = args.Action;
        var report = args.Report;
        var parameters = args.RequestParams;
    }
}
```

Information

The event won't be invoked for events that have their own processors - printing, export, interactive actions on a report etc. These events are described separately in the relevant sections of the documentation.

7.1.7 Printing Reports

Several options of report printing are envisaged in the **Blazor Viewer** component. Each of them has its features, advantages, and disadvantages.

Print to PDF
Printing will be performed with the help of report exporting to **PDF format**. Advantages include great accuracy of location and report elements printing in comparison with other printing options. Among the disadvantages, we can mention the obligatory presence of a plug-in installed in the browser for viewing PDF files (modern Web browsers have an embedded tool for viewing and printing PDF files).

Print with Preview
Report printing will be performed in the separate pop-up window of a browser in the **Blazor Viewer**. A report can be previewed and then sent to the printer or copied to another place as text or an HTML code. Its advantages contain cross-browser compatibility when printing, the absence of need for special plug-ins installation. However, there is one disadvantage; it is relatively low accuracy of report elements location, tied features of the implementation of HTML formatting.

Print without Preview
Report printing will be performed directly in the printer without preview. After selection of this, system print dialog displays. So as printing in this mode is done in HTML, print quality is analogous to report printing quality with the Preview.

Information

© 2003-2024 Stimulsoft

When printing to HTML, you should make sure a report page parameters correspond to the printer page parameters (the size of paper, orientation, fields, indents) and check such browser print settings as indents, headers, and footers, the printing of background images, color printing.

You don't need additional settings of the Viewer for print functions to work. If you need to make some actions before a report print, you can specify the special **OnPrintReport** event.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnPrintReport="@OnPrintReport" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   private void OnPrintReport(StiPrintReportEventArgs args)
   {
      // Some code before print
      // ...
   }
}
```

Report printing setting

The menu with print options is displayed when selecting a report print in the Viewer panel. The **Blazor Viewer** component has a feature to set the requested print mode forcibly. To use this option, you should set the **PrintDestination** property to one of the **StiPrintDestination** enum values specified below.

- **Default** – when selecting a printing, the menu (property value by default) will be displayed;
- **Pdf** – printing in PDF format;
- **Direct** – printing in an HTML format directly to the Printer, the systemic print dialog will be displayed;
- **WithPreview** – printing in HTML format with the Preview in a pop-up window.

© 2003-2024 Stimulsoft

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;


   protected override void OnInitialized()
   {
      base.OnInitialized();

      Options = new StiBlazorViewerOptions();
      Options.Toolbar.PrintDestination = StiPrintDestination.Default;
   }
}
```

The HTML5 component has a feature, which allows you to disable report printing. To do this, you should set the **ShowPrintButton** property to **false**.

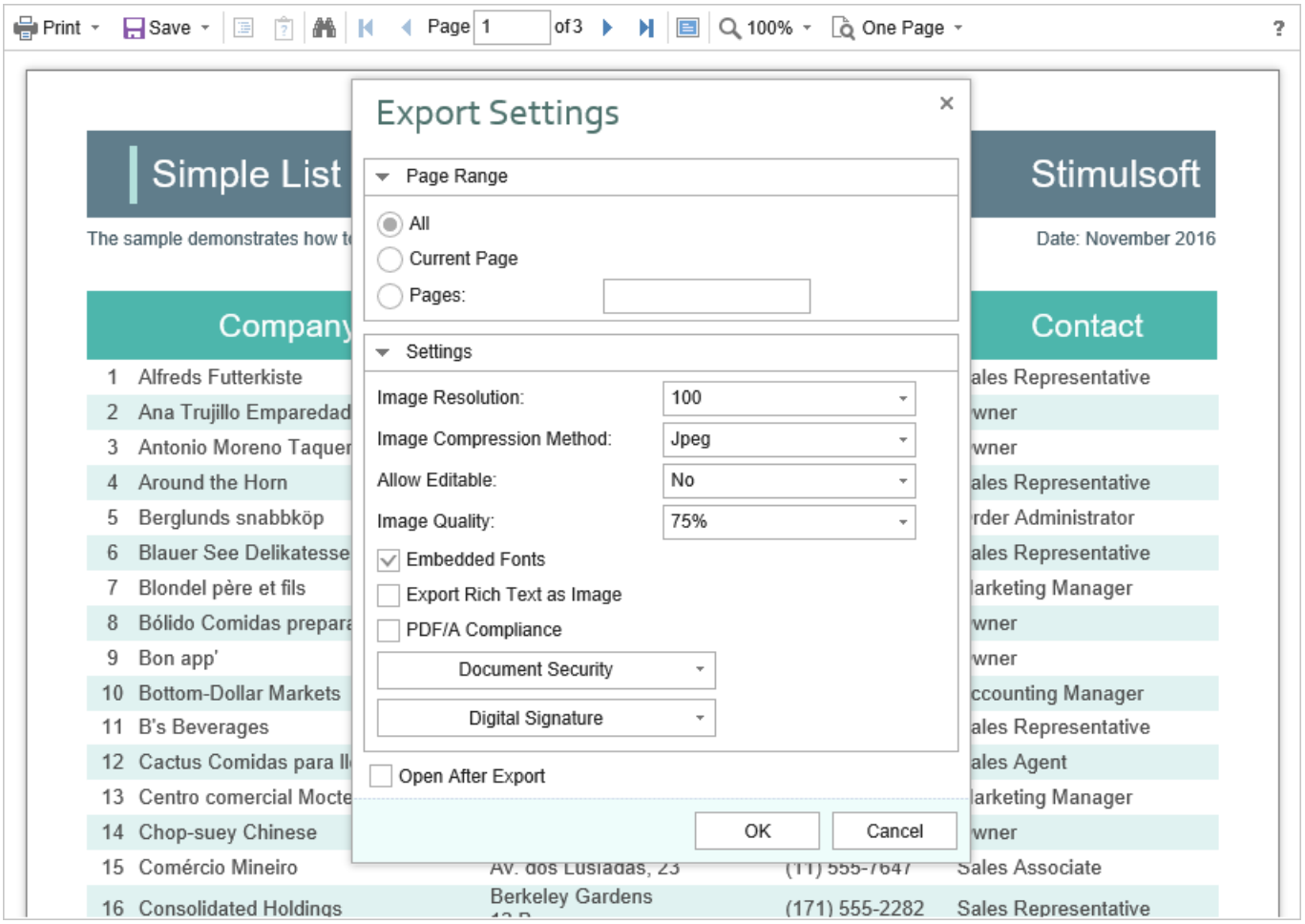
Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options"></StiBlazorViewer>

@code
{
   //Options object
   private StiBlazorViewerOptions Options;


   protected override void OnInitialized()
   {
      base.OnInitialized();

      Options = new StiBlazorViewerOptions();
      Options.Toolbar.ShowPrintButton = true;
   }
}
```

7.1.8 Exporting Reports

The **Blazor Viewer** component allows you to export a displayed report to various formats such as **PDF**, **HTML**, **Word**, **Excel**, text, etc. The export function doesn`t require additional settings in the viewer.

© 2003-2024 Stimulsoft



If you need to make any actions before exporting a report, you can set the special **OneExportReport** event.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnExportReport="@OnExportReport" />

@code
{
   private void OnExportReport(StiExportReportEventArgs args)
   {
      // Some code before export
      // ...
   }
}
```

© 2003-2024 Stimulsoft

Export Settings

Each report export format of the Blazor Viewer has a lot of settings, and each setting has its values by default. Sometimes you need other values by default. The special **DefaultSettings** property of the Viewer is used for this. You can find it in the export options. This property is the container of all export settings used by default.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();

      //PDF default settings
      Options.Exports.DefaultSettings.ExportToPdf.ImageQuality = 0.75f;
      Options.Exports.DefaultSettings.ExportToPdf.ImageFormat =
      Stimulsoft.Report.Export.StiImageFormat.Color;

      //HTML default settings
      Options.Exports.DefaultSettings.ExportToHtml.UseEmbeddedImages =
      true;
      Options.Exports.DefaultSettings.ExportToHtml.ExportMode =
      Stimulsoft.Report.Export.StiHtmlExportMode.Div;
   }
}
```

If required, you can completely hide the display of the export dialog windows. The exporting will always be done with the settings by default. To do this, you just need to set the false value for the **ShowExportDialog** property.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
```

© 2003-2024 Stimulsoft

```
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Exports.ShowExportDialog = false;
   }
}
```

The **Blazor Viewer** component contains about 20 various export formats, and sometimes you need to disable some of them. It allows you to load the interface and simplify the use of the Viewer. To disable not used export formats, just set the **false** value for corresponding viewer properties, given in the list below.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Exports.ShowExportDialog = true;
      Options.Exports.ShowExportToDocument = true;
      Options.Exports.ShowExportToPdf = true;
      Options.Exports.ShowExportToXps = true;
      Options.Exports.ShowExportToPowerPoint = true;
      Options.Exports.ShowExportToHtml = true;
      Options.Exports.ShowExportToHtml5 = true;
      Options.Exports.ShowExportToMht = true;
      Options.Exports.ShowExportToText = true;
      Options.Exports.ShowExportToRtf = true;
```

© 2003-2024 Stimulsoft

```
        Options.Exports.ShowExportToWord2007 = true;
        Options.Exports.ShowExportToOpenDocumentWriter = true;
        Options.Exports.ShowExportToExcel = true;
        Options.Exports.ShowExportToExcelXml = true;
        Options.Exports.ShowExportToExcel2007 = true;
        Options.Exports.ShowExportToOpenDocumentCalc = true;
        Options.Exports.ShowExportToCsv = true;
        Options.Exports.ShowExportToDbf = true;
        Options.Exports.ShowExportToXml = true;
        Options.Exports.ShowExportToDif = true;
        Options.Exports.ShowExportToSylk = true;
    }
}
```

The **Blazor Viewer** component has a feature, which allows you to disable the report export menu. To do this, you should set the **ShowSaveButton** property to **false**.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
    //Options object
    private StiBlazorViewerOptions Options;

    protected override void OnInitialized()
    {
        base.OnInitialized();

        //Init options object
        Options = new StiBlazorViewerOptions();
        Options.Toolbar.ShowSaveButton = false;
    }
}
```

7.1.9 Vieweing Modes

There are two modes of report display - with scroll bars and without them. The mode to view reports without scroll bars is set by default. To disable the viewing mode with scroll bars, you should set the **ScrollbarsMode** property to **true**.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web
```

© 2003-2024 Stimulsoft

```
<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Appearance.ScrollbarsMode = false;
   }
}
```

The Viewer displays a page or a report entirely, automatically stretching the viewer in the first mode (without scroll bars). If the sizes in width and heights are specified, the Viewer will crop the page that has gone beyond the margins. In the second mode, unlike the first one, the cropping won`t be made when a page goes beyond the Viewer size. Instead, scroll bars appear, with the help of them, you can view the entire page or a report.

Information

You should set the height of the Viewer in the report viewing mode; otherwise, the height is **650 pixels** will be set by default.

The mode of a report or a dashboard full-screen display is envisaged in the **Blazor Viewer** component. The standard viewing mode is enabled by default. The Viewer has specified sizes in settings. To enable the full-screen mode of viewing, just set the **true** value for the **FullScreenMode** property.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
```

© 2003-2024 Stimulsoft

```
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Appearance.FullScreenMode = true;
   }
}
```

Also, to enable or disable the full-screen mode, you can use the corresponding button in the Viewer toolbar.

There are three report display modes in the **Blazor Viewer** - page display, full report as a ribbon, and tabular display of report pages. The **View Mode** property is used for this. This property accordingly takes one of the following values - **SinglePage**, **Continuous**, **MultiplePages**.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Toolbar.ViewMode = StiWebViewMode.SinglePage;
   }
}
```

The support of work with a simple computer as well as with touchscreens and mobile devices is realized in the **Blazor Viewer** component. The Interface type property is intended for managing modes. This property takes one of the following

© 2003-2024 Stimulsoft

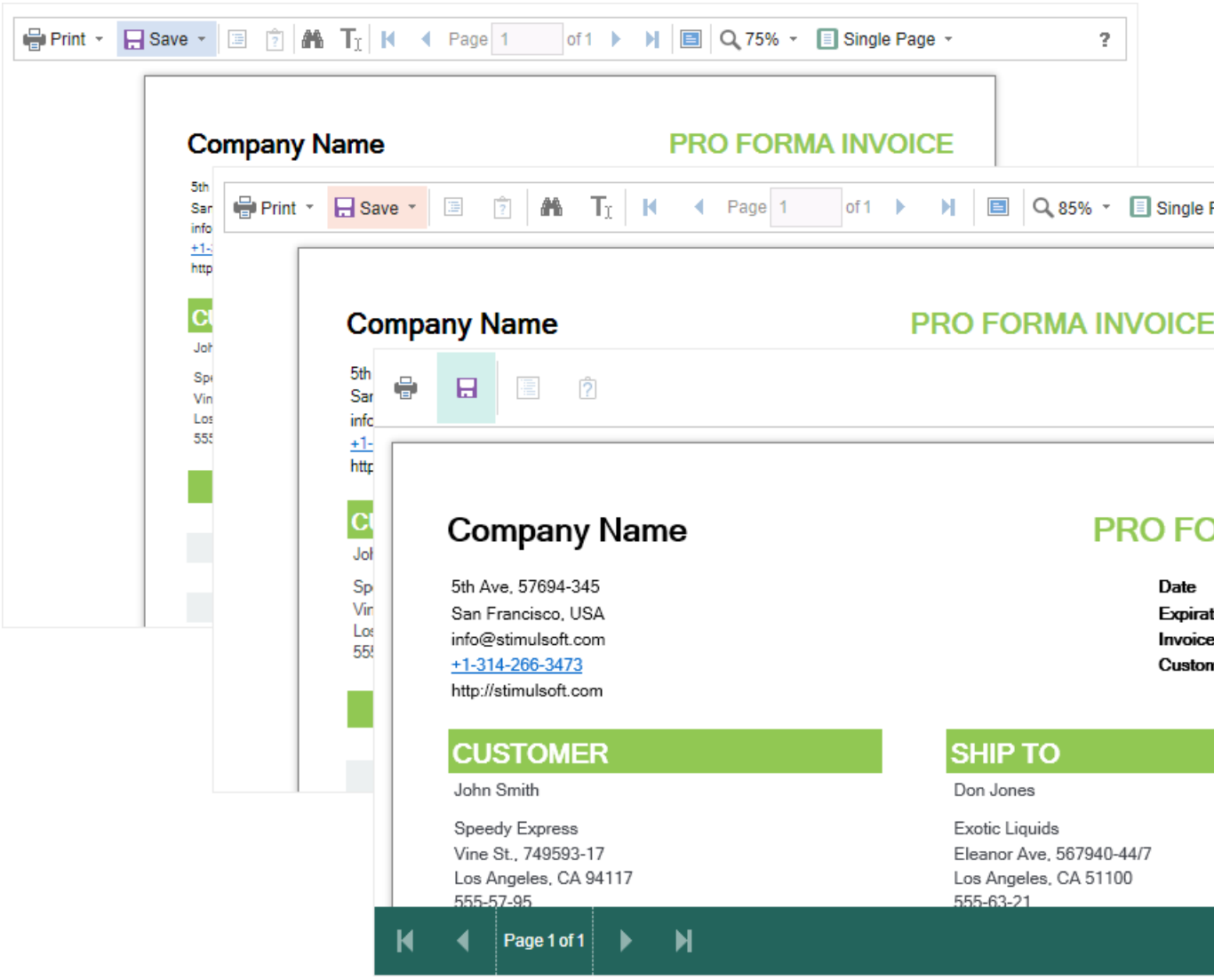
values:

- **Auto** - the Viewer interface type will be selected automatically depending on the device used (value by default).
- **Mouse** - forced using the standard interface for managing the viewer.
- **Touch** - forced using the Touch interface for managing the viewer with the help of the touchscreen monitor. In this mode, the viewer interface elements have increased sizes for comfortable managing.
- **Mobile** - forced use of Mobile interface for managing viewer with the help smartphone screen, in this mode the viewer interface has a simplified view and adapted for managing with the help of a mobile device.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Appearance.InterfaceType = StiInterfaceType.Auto;
   }
}
```

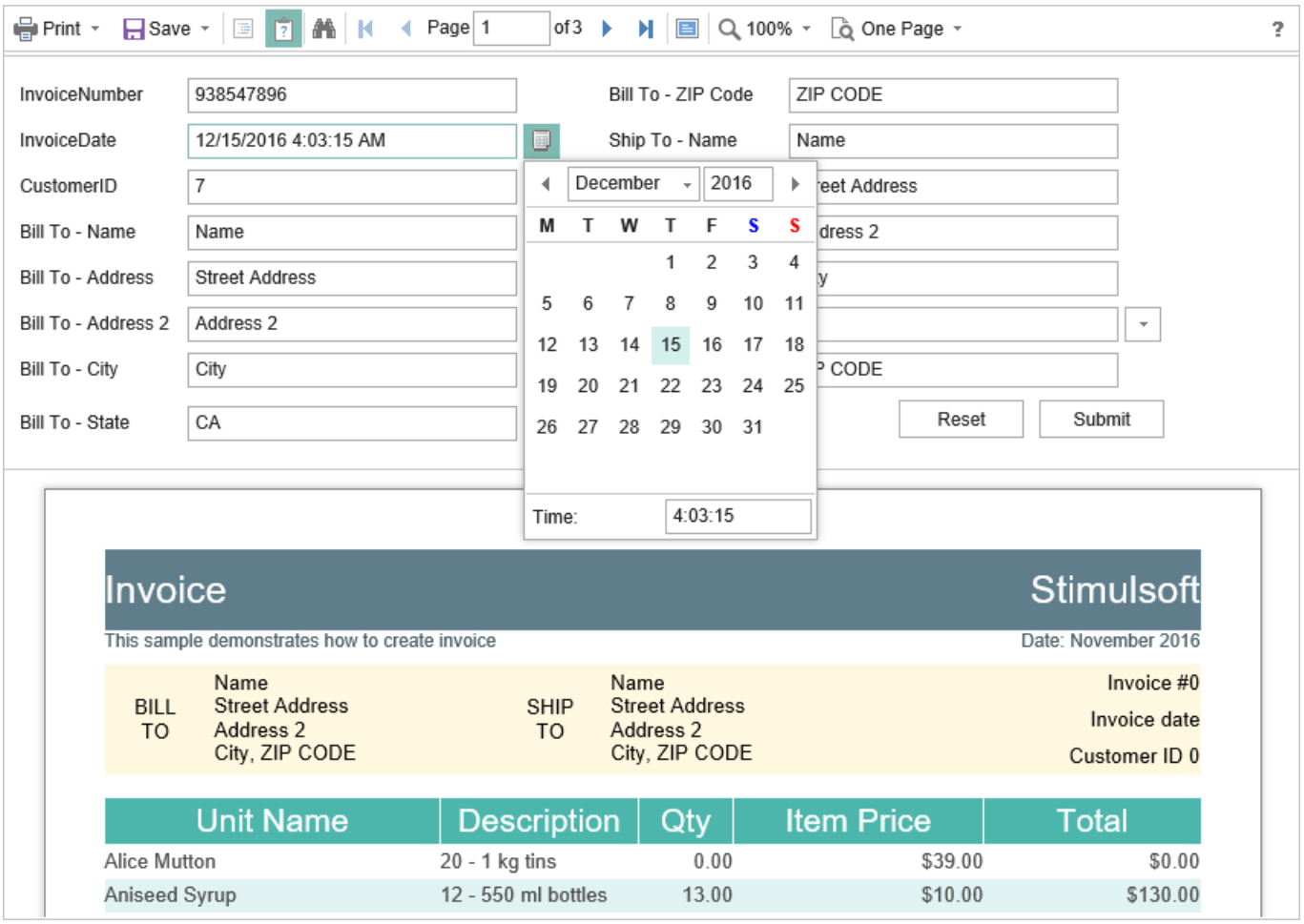
© 2003-2024 Stimulsoft



7.1.10 Work with Parameters

The support of a particular parameter panel for work with report parameters in the **Blazor Viewer** is realized. To add a parameter to the panel, you should define the variable requested from a user in a report. When viewing a report in the viewer, this variable will be added automatically to the parameter panel — all types of report variables (simple variables, date and time, borders, lists, etc.).

© 2003-2024 Stimulsoft



Additional viewer settings are not required for working reports with parameters. If you need to take some actions before using parameters, you can define the special **OnInteraction** event.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnInteraction="@OnInteraction" />

@code
{
   private void OnInteraction(StiReportDataEventArgs args)
   {
      // Some code before any interaction
      // ...
   }
}
```

© 2003-2024 Stimulsoft

This event is triggered for any interactive viewer actions. If you need to make some actions only when applying report parameters, you can use value parameters in argument events. The arguments contain all necessary data and conditions of the viewer client side. To define a type of viewer actions, you should use the **Action** property.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnInteraction="@OnInteraction" />

@code
{
   private void OnInteraction(StiReportDataEventArgs args)
   {
      if (args.Action == StiAction.Variables)
      {
         // Some code before apply parameters
      }
   }
}
```

If the work with parameters is requested, you can disable this feature. The **ShowParametersButton** property is intended for this. This property is located in the **Toolbar** properties section, and the **false** value must be set for it.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
```

© 2003-2024 Stimulsoft

```
        Options = new StiBlazorViewerOptions();
        Options.Toolbar.ShowParametersButton = false;
    }
}
```

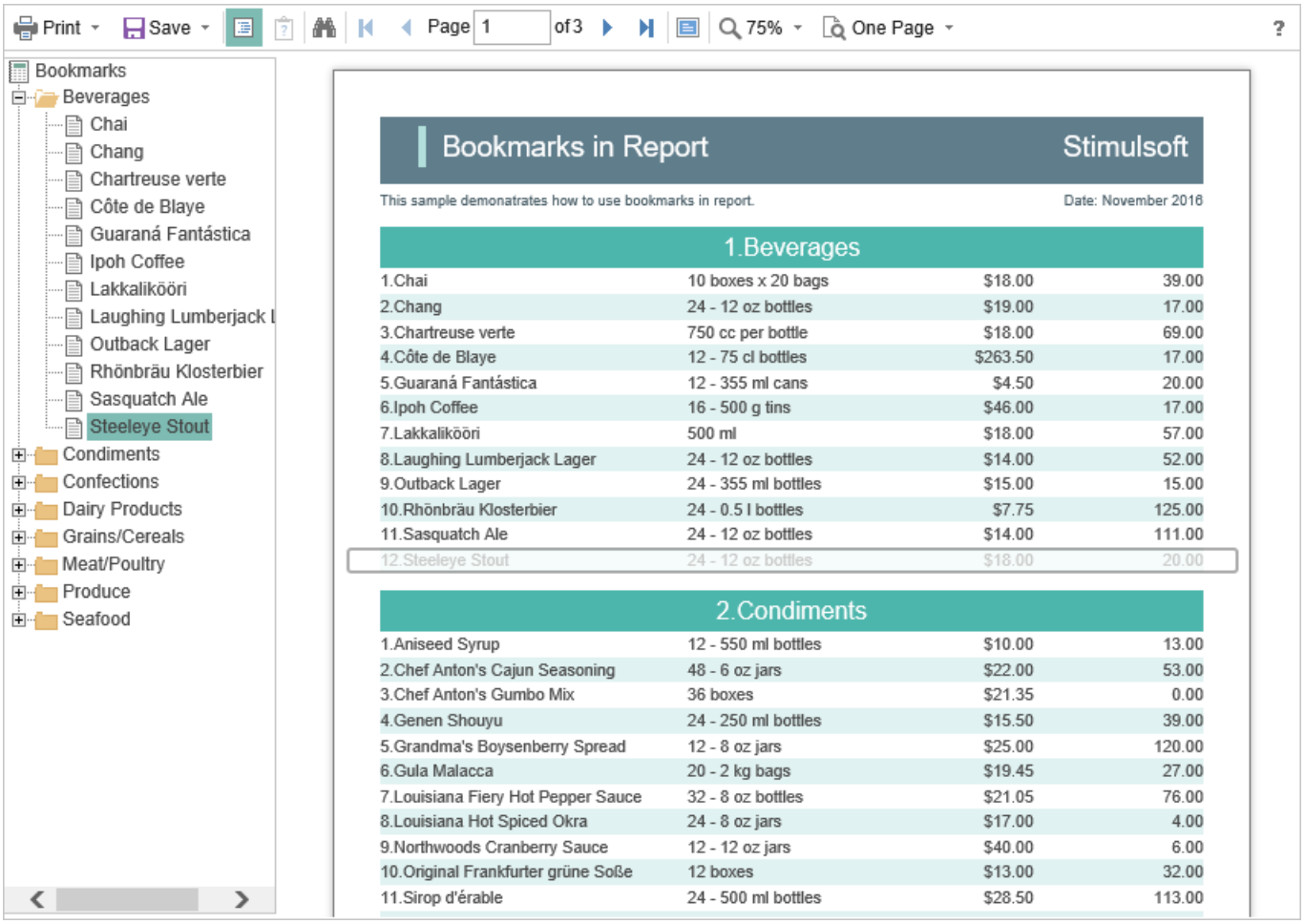
Information

Due to this viewer configuration, the parameter panel won't be shown, even if the parameters are present in a displayed report.

7.1.11 Work with Bookmarks

The support of report bookmarks is implemented in the **Blazor Viewer** component. When displaying a report, the panel with bookmarks will be displayed to the left of the page. When selecting a bookmark for a report, the viewer will automatically transit to the page you need, and the report element with a bookmark will be highlighted.

© 2003-2024 Stimulsoft



By default, the width of the bookmark panel is 180 pixels; the **Blazor Viewer** component allows you to change this value. The **BookmarksTreeWidth** is intended for this. Its value is specified in pixels.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();
```

© 2003-2024 Stimulsoft

```
    //Init options object
    Options = new StiBlazorViewerOptions();
    Options.Appearance.BookmarksTreeWidth = 200;
  }
}
```

If the work with report bookmarks is not requested, you can completely disable this feature. The **ShowBookmarksButton** property is used for this, and it should be set to **false**.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
{
  //Options object
  private StiBlazorViewerOptions Options;

  protected override void OnInitialized()
  {
    base.OnInitialized();

    //Init options object
    Options = new StiBlazorViewerOptions();
    Options.Toolbar.ShowBookmarksButton = false;
  }
}
```

Information

In this case, report bookmarks won't be shown, even if they are present in a displayed report. This feature does not exert influence over printing and exporting a report.

When printing a report with bookmarks, the tree of bookmarks will be hidden. If apart from a report you need to print and bookmarks too, you should set the **BookmarksPrint** property to **true**.

© 2003-2024 Stimulsoft

Index.razor

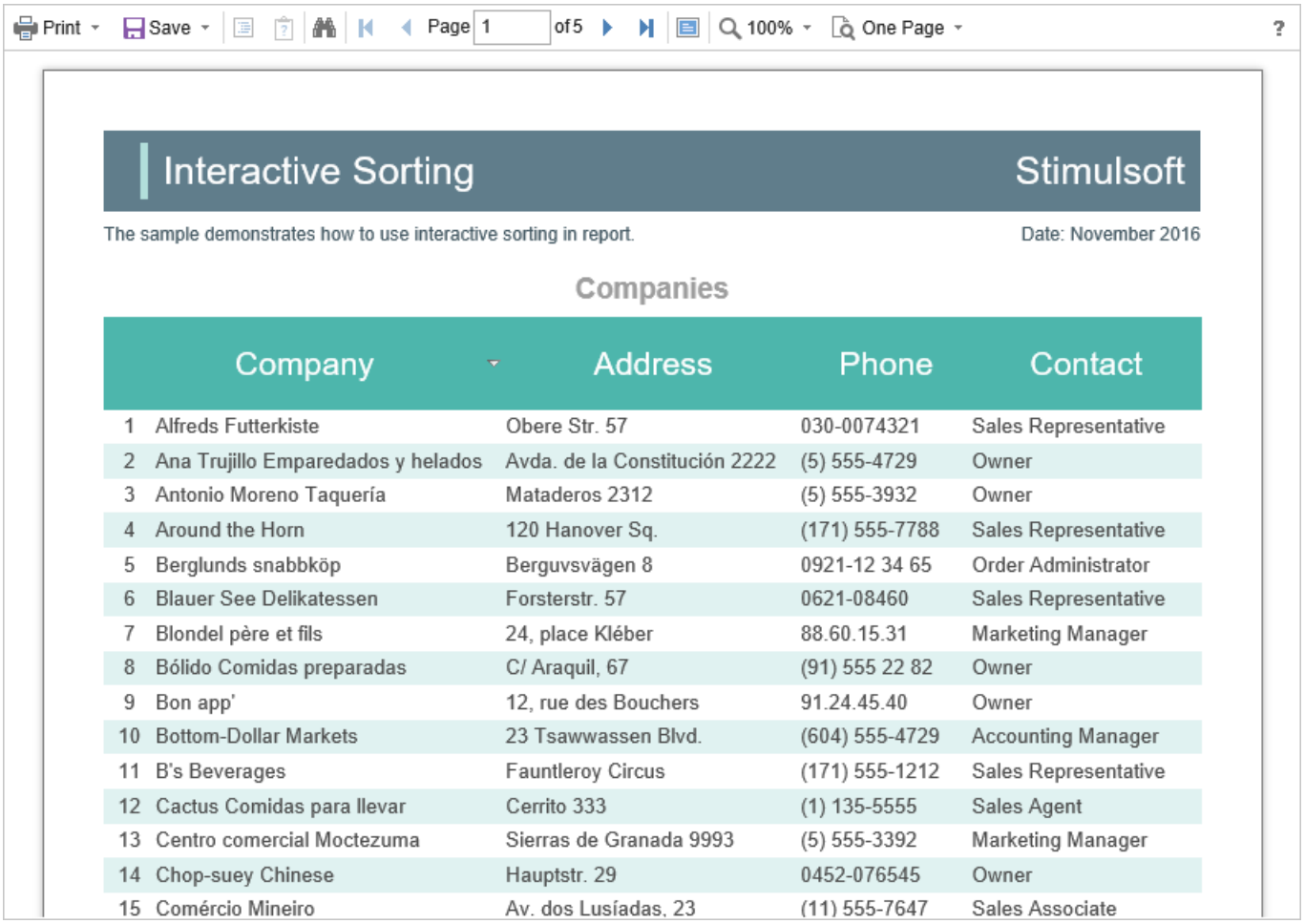
```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@options" />

@code
{
   //Options object
   private StiBlazorViewerOptions options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      Stimulsoft.Base.StiFontCollection.AddFontFile("Fonts/Microsoft Sans
      Serif.ttf", "Segoe UI");

      //Init options object
      options = new StiBlazorViewerOptions();
      options.Appearance.BookmarksPrint = true;
   }
}
```

7.1.12 Dynamic Sorting, Collapsing, and Drill-Down

The **Blazor Viewer** supports dynamic sorting, collapsing, and drill down in reports. Dynamic sorting allows changing the sort direction in a rendered report. To do it, you should click on the component in which dynamic sorting was set. Dynamic sorting is carried out in the following direction: **Ascending** and **Descending**. Each time when you click on some component, the direction is reversed.

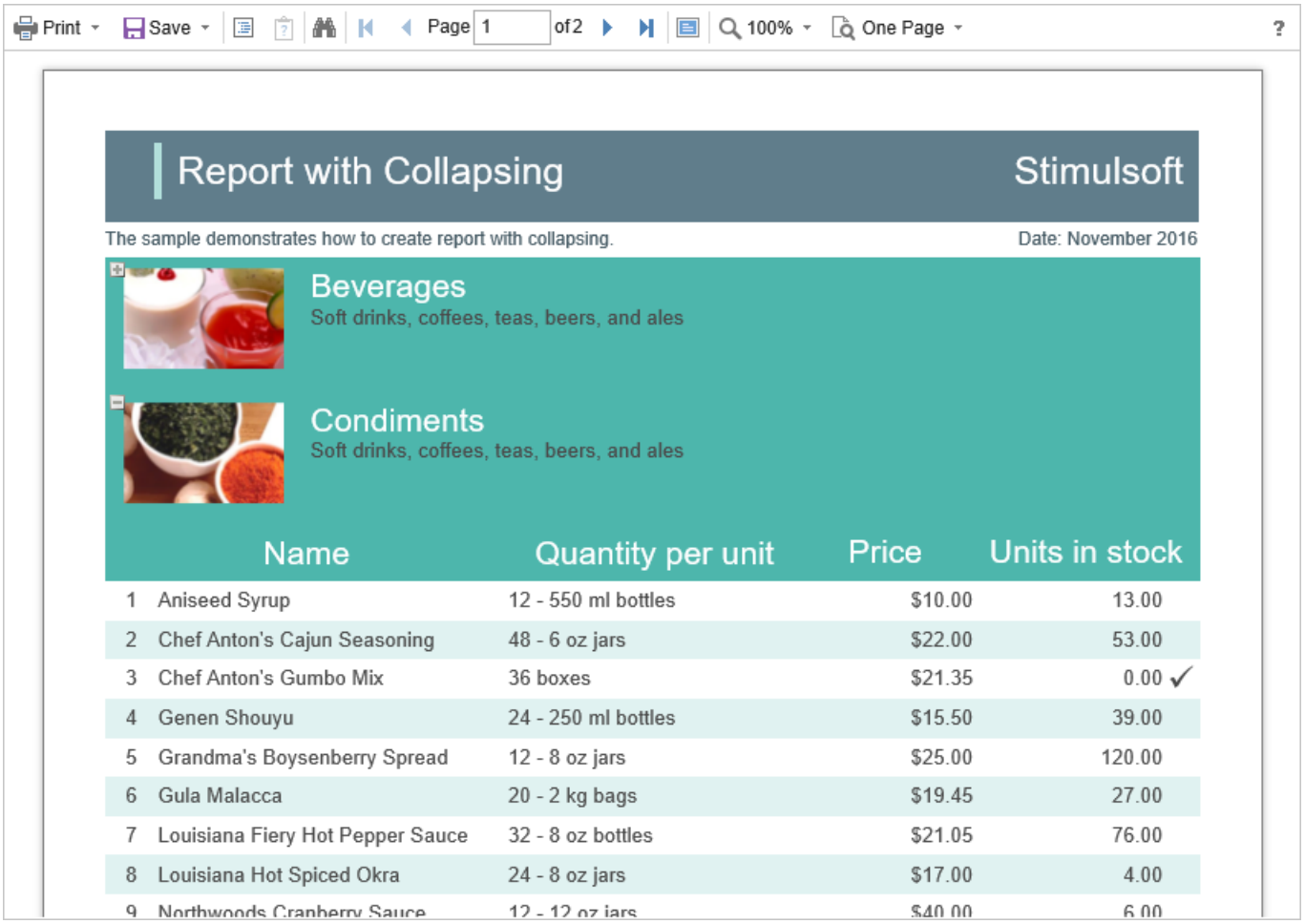
Multilevel sorting is allowed in a report. To do it, you should hold down the **Ctrl** and, step-by-step, click on sorted report components. To reset the sort, you can click on any sorted report without holding down the **Ctrl**.

© 2003-2024 Stimulsoft



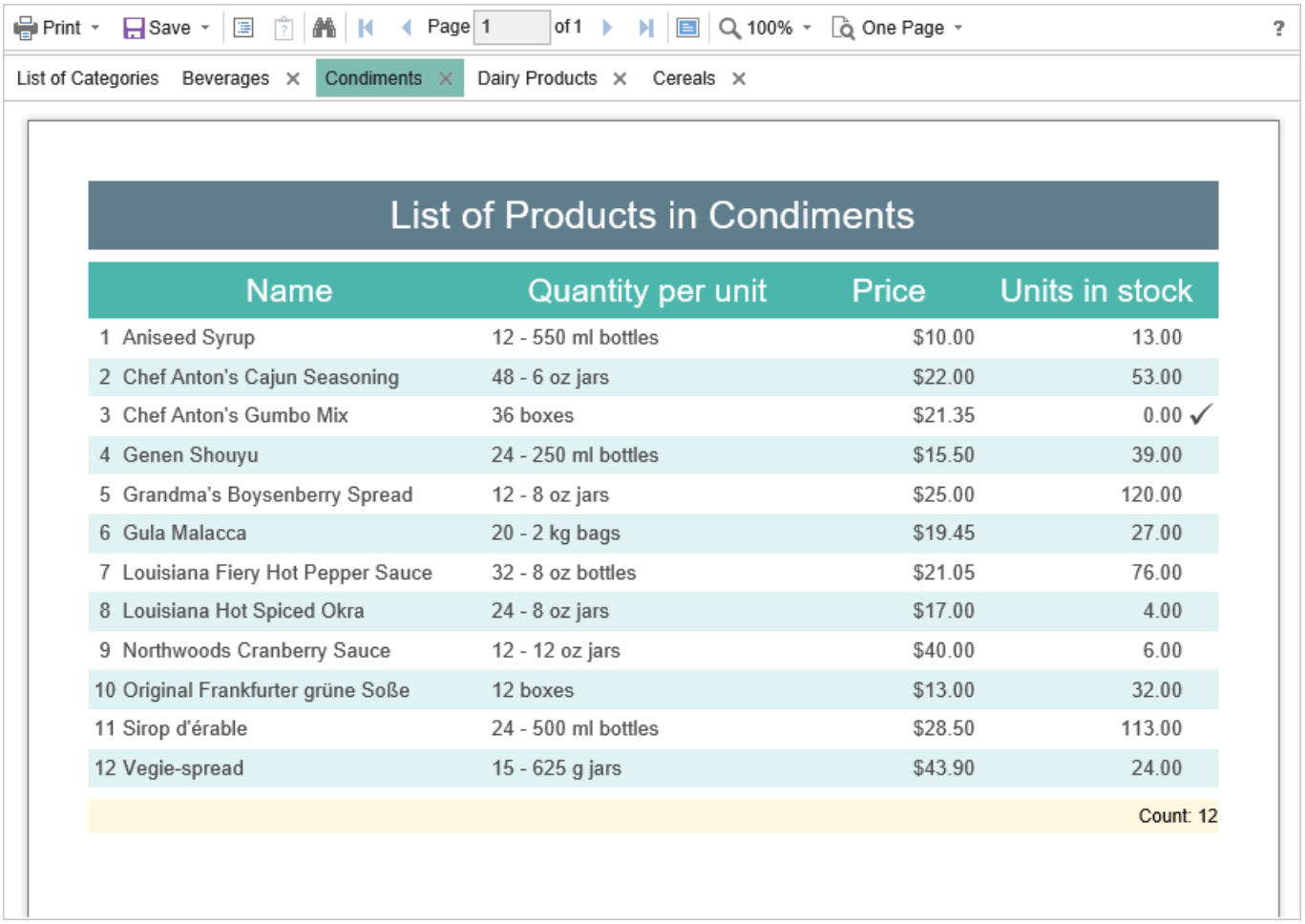
A report with dynamic collapsing is an interactive report where collapsing blocks can collapse/expand their content when clicking on the header of the block. The report elements which you can collapse/expand are highlighted with a special icon with **[-]** or **[+]**.

© 2003-2024 Stimulsoft



When data is drilled down under the main panel, the drill down panel with the bookmarks of drill down reports will be displayed. A displayed report now will be highlighted.

© 2003-2024 Stimulsoft



Additional viewer settings are not required for the work with dynamic sorting, collapsing, and drilling down. If you need to make some actions before sorting or drilling down a report, you can define the special **OnInteraction** event.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnInteraction="@OnInteraction" />

@code
{
   private void OnInteraction(StiReportDataEventArgs args)
   {
      // Some code before any interaction
      // ...
   }
}
```

© 2003-2024 Stimulsoft

To get an action type, you can use an event argument. A definite type of action is envisaged for each kind of viewer interaction:

- The **Sorting** - when using sorting and columns.
- The **DrillDown** - when using a report drill down.
- The **Collapsing** - when using a report blocks collapsing.

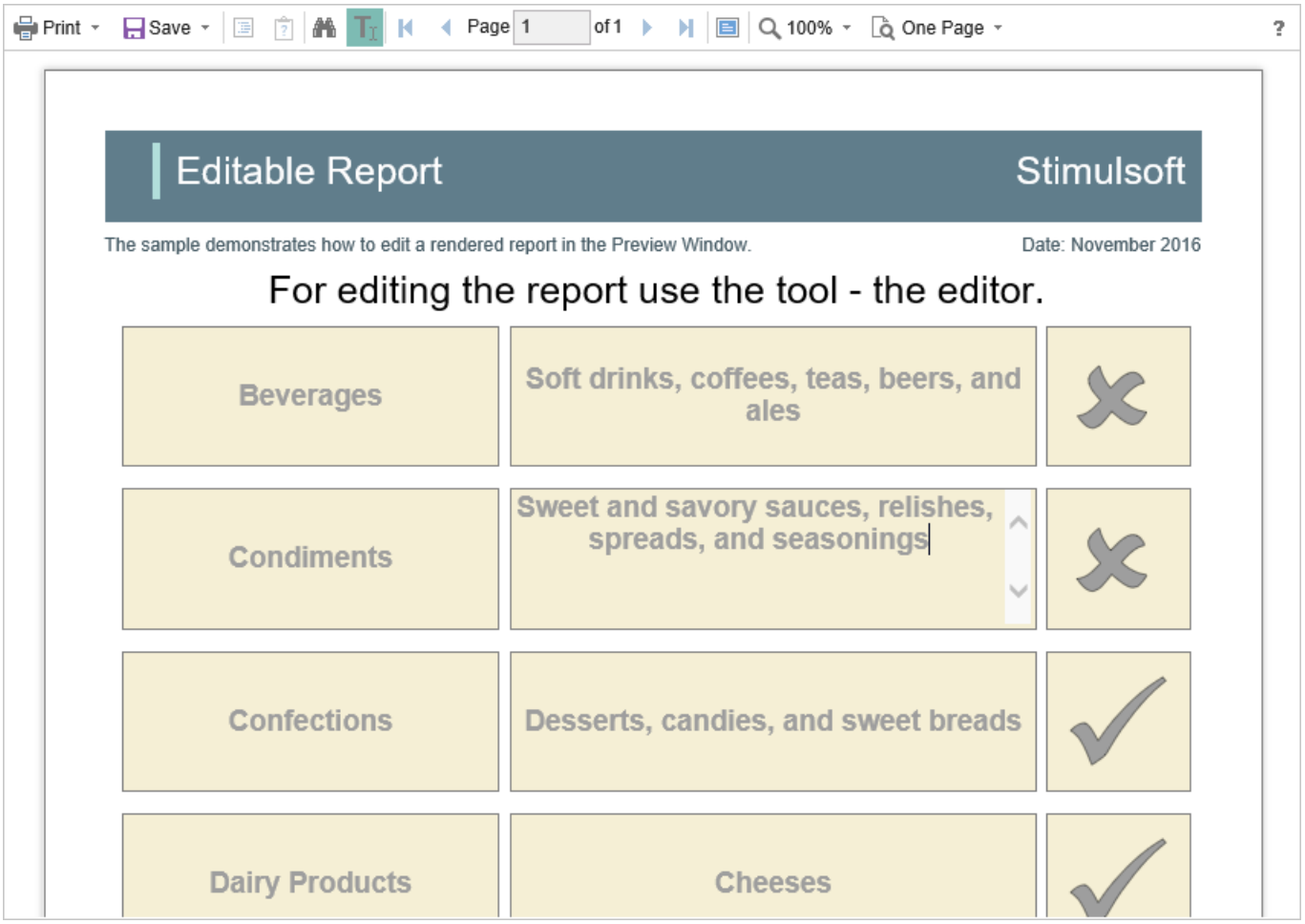
Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnInteraction="@OnInteraction" />

@code
{
   private void OnInteraction(StiReportDataEventArgs args)
   {
      switch (args.Action)
      {
         case StiAction.Sorting:
            break;

         case StiAction.DrillDown:
            break;

         case StiAction.Collapsing:
            break;
      }
   }
}
```

7.1.13 Editing Report

The **Blazor Viewer** component has the ability to edit report items, such as text boxes and check boxes. In order the editing be possible, in the report template, you should mark the required components as editable. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be applied to the report.

© 2003-2024 Stimulsoft



For the report edit mode, no special settings of the viewer required.

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

7.1.14 Sending Report by Email

The option to send a report by email is implemented in the **Blazor Viewer** component. To activate it, you should set the **ShowSendEmailButton** viewer property to true and define the **OnEmailReport** event.

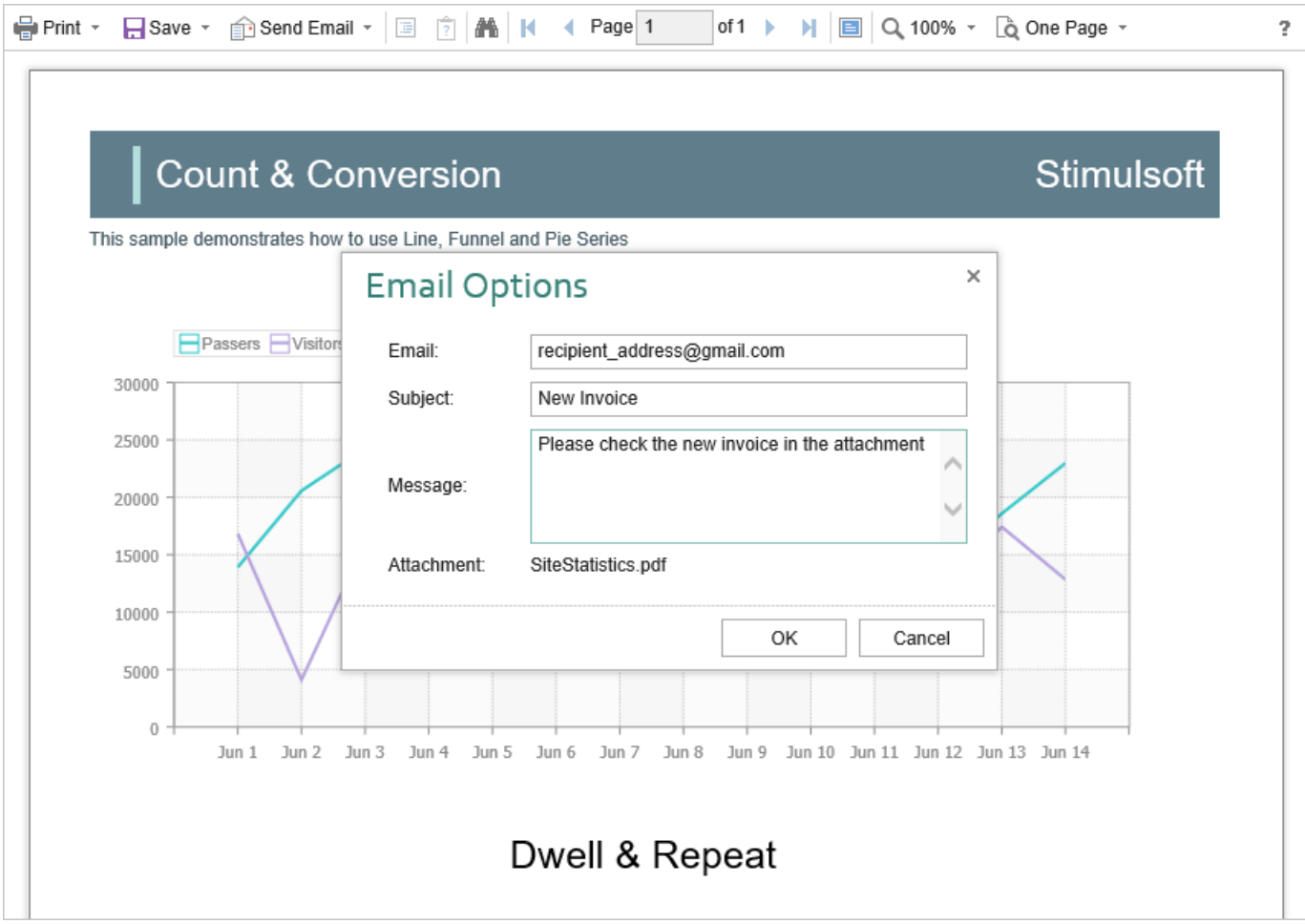
© 2003-2024 Stimulsoft

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" OnEmailReport="@OnEmailReport" />

@code
{
   //Options object
   private StiBlazorViewerOptions Options;

   private void OnEmailReport(StiEmailReportEventArgs args)
   {
      //args.Options.AddressTo = "";
      //args.Options.Subject = "";
      //args.Options.Body = "";

      // Should be filled here
      args.Options.AddressFrom = "admin_address@test.com";
      args.Options.Host = "smtp.test.com";
      args.Options.Port = 465;
      args.Options.UserName = "admin_address@test.com";
      args.Options.Password = "admin_password";

      //args.Options.CC.Add("email@test.com");
      //args.Options.BCC.Add("email@test.com");
      //args.Options.EnableSsl = true;
   }
}
```

The menu of attachment format selection displays when sending a report by email. This menu corresponds to the Export format menu. After a format is selected, the parameters dialog, such as recipient`s email, theme, and text of the letter, will display.

© 2003-2024 Stimulsoft



After confirming the submission, the **OnEmailReport** event described above will be triggered, where you can check and correct data typed in this form. The exported report file will be attached to an email automatically.

The **Blazor Viewer** component allows setting values by default for the Send Email form. The **DefaultEmailAddress**, the **DefaultEmailSubject**, and **DefaultEmailMessage** properties are used for this. These properties are empty by default.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer Options="@Options" />

@code
```

© 2003-2024 Stimulsoft

```
{
   //Options object
   private StiBlazorViewerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorViewerOptions();
      Options.Email.DefaultEmailAddress = "recipient_address@gmail.com";
      Options.Email.DefaultEmailSubject = "New Invoice";
      Options.Email.DefaultEmailMessage = "Please check the new invoice in
      the attachment";
   }
}
```

7.1.15 Calling Designer from Viewer

The **Blazor Viewer** component has the ability to call the report designer. To use this feature, you should define the **OnDesignReport** event handler.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web
@inject NavigationManager NavigationManager

<StiBlazorViewer Report="@Report" OnDesignReport="@OnDesignReport" />

@code
{
   //Report object to use in viewer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      var report = new StiReport();
      report.Load("Reports/TwoSimpleLists.mrt");
      Report = report;
   }

   protected void OnDesignReport(StiReportDataEventArgs args)
   {
      //Redirect to the Designer page
      NavigationManager.NavigateTo("Designer?report=" +
      args.Report.ReportName);
   }
}
```

Information

© 2003-2024 Stimulsoft

The viewer doesn't run the designer. It just calls the specified event in which you can get all necessary parameters. Later in the activity, you can redirect to another page that contains the report designer.

7.1.16 Export and Printing from Code

The **Blazor Viewer** provides the ability to print reports in various ways and export reports to various formats. These actions are performed using the viewer menu. If you want to print or export a report by using the code, for example, in the event of pressing the button, you can use the special **StiReportResponse** class. This class contains a set of static methods that allow you to print or export a report from the code, and the report viewer is not required.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<button @onclick="@OnClickPrintButton">Print PDF</button>
<button @onclick="@OnClickExportButton">Export PDF</button>

@code
{
   private StiReport LoadSimpleList()
   {
      var dataSet = new System.Data.DataSet();
      dataSet.ReadXml("Data/Demo.xml");

      var report = new StiReport();
      report.Load("Reports/SimpleList.mrt");
      report.RegData(dataSet);

      return report;
   }

   protected void OnClickPrintButton()
   {
      var report = LoadSimpleList();

      StiReportResponse.PrintAsPdf(report);
      //StiReportResponse.PrintAsHtml(report);
   }

   protected void OnClickExportButton()
   {
      var report = LoadSimpleList();

      StiReportResponse.ResponseAsPdf(report);
      //StiReportResponse.ResponseAsExcel2007(report);
```

© 2003-2024 Stimulsoft

```
        //StiReportResponse.ResponseAsPng(report);
        //StiNetCoreReportResponse.ResponseAsJson(report);
    }
}
```

If the **Razor** page does not have components for working with reports (viewer or designer), it is necessary to pre-initialize the reporting tool. You can do this by overriding the standard **OnInitializedAsync** event of a report by adding a special static method, **StiBlazorHelper.Initialize()**, to which you need to pass a **JSRuntime** object as input. After that, the actions for exporting and printing the report will work correctly.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@inject IJSRuntime JSRuntime;

protected override Task OnInitializedAsync()
{
    StiBlazorHelper.Initialize(JSRuntime);

    return base.OnInitializedAsync();
}
```

7.1.17 Viewer Events

The **Blazor Viewer** component supports events which allows you to execute necessary operations before certain actions, such as printing and exporting, sending reports by email, interactivity etc. Below is a sample of processing viewer events.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer OnViewerReport="@OnViewerEvent" />

@code
{
    private void OnViewerEvent(StiReportDataEventArgs args)
    {
        var action = args.Action;
        var report = args.Report;
        var parameters = args.RequestParams;
    }
}
```

© 2003-2024 Stimulsoft

Events list

Name	Description
OnOpenReport	The event occurs when opening a report.
OnPrintReport	The event occurs when printing a report.
OnExportReport	The event occurs when exporting a report.
OnEmailReport	The event occurs when sending a report by email.
OnInteraction	The event occurs when the viewer works with interactive operations, such as using parameters, dynamic sorting, collapsing, and drilling a report.
OnViewerEvents	The event occurs for any action in the report viewer.
OnDesignReport	The event occurs when pressing the Design button on the toolbar of the viewer.
OnViewerAfterRender	The event occurs when the HTML5 code of the viewer and all its controls have been completed.

7.1.18 Viewer Settings

The **Blazor Viewer** setting is configured with the help of component properties. The main settings are set in the Razor component; all additional settings are set using the **StiBlazorViewerOptions** class. Below is an example of the viewer properties setting.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorViewer ID="Viewer1" Width="500px" Height="500px"
Localization="Lozalization/en.xml" Options="@Options"
Theme="StiViewerTheme.Office2022WhiteCarmine" />

@code
{
   //Options object
```

© 2003-2024 Stimulsoft

```
    private StiBlazorViewerOptions Options;

    protected override void OnInitialized()
    {
        base.OnInitialized();

        //Init options object
        Options = new StiBlazorViewerOptions();
        Options.Appearance.ScrollbarsMode = true;
        Options.Appearance.ShowTooltips = false;
        Options.Toolbar.DisplayMode = StiToolbarDisplayMode.Separated;
        Options.Appearance.ReportDisplayMode =
        StiReportDisplayMode.FromReport;
        Options.Exports.ShowExportToDbf = false;
        Options.Exports.ShowExportToDif = false;
        Options.Exports.DefaultSettings.ExportToPdf.CreatorString = "Company
        Name";
        Options.Exports.DefaultSettings.ExportToPdf.ImageQuality = 0.75f;
    }
}
```

The Viewer properties

Name	Description
ID	Allows you to set a string unique identifier of a component, by default **StiBlazorViewer**.
Report	Allows you to set report object that will be displayed in the viewer.
Localization	Specifies the path to the XML localization file. The path can be absolute or relative. By default, the English localization is used, which is built into the viewer and does not require additional XML files.
Width	Sets the width of the component in the required units - pixels (default value), percentage, points. For example, “500”, “500px”, “100%”, “300pt”.
Height	Sets the height of the component in the required units - pixels (default value), percentage, points. For example, “500”, “500px”, “100%”, “300pt”. The automatic height is set by default, depending on the size of a report page or 650 pixels in the mode of the viewer display

© 2003-2024 Stimulsoft

	with scroll bars.
Options	It allows specifying the object of the **StiBlazorViewerOptions**, which contains the set of the component options.
Theme	Specifies the theme of the viewers layout. The list of available themes can be found in the **StiTheme** enumeration. The default value is **Office2022WhiteCarmine**.

Server

Name	Desription
UseRelativeUrls	Sets the viewer mode in which relative URLs are used for AJAX requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value which specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersToReport	Enables using all the URL parameters of the request as the variable values. The variable names must match the parameters. The default value of the property is **false**.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the viewer. If **false**, only the necessary parameters are used to request the resources of the viewer. By default, the property is set to **true**.
PassFormValues	Enables passing the values of the POST form to the client side, if these values are required to be used in the actions of the viewer. If you enable this property, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
AllowLoadingCustomFontsToClie	Allows you to pass custom fonts to the client

© 2003-2024 Stimulsoft

ntSide	side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Sets the path to the CSS file of the viewer's styles. The standard styles of the chosen theme will not be loaded if this property has got a value. The default value of the property is an empty string.
BackgroundColor	Sets the background color of the viewer. By default it is set to **White**.
PageBorderColor	Sets the border color of the viewer. By default it is set to **Gray**.
RightToLeft	Sets the **Right to Left** mode for viewer controls. By default the property is set to **false**. By default, the property is set to **false**.
FullScreenMode	Sets the full-screen display mode of the viewer. By default, the property is set to **false**.
ScrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to **false**.
OpenLinksWindow	Sets the target window for opening links contained in the report. By default, the property is set to **Blank** (new window).
OpenExportedReportWindow	Sets the target window for opening the export file from the viewer. By default, the property is set to **Blank** (new window).
DesignWindow	Sets the destination window for launching the report designer. The default value of the property is **Self** (whish is the current window).
ShowTooltips	Enables showing tips for the viewer controls when the mouse hovers over. By default, the

© 2003-2024 Stimulsoft

	property is set to **true**.
ShowTooltipsHelp	Enables showing links to online documentation for the viewer controls. By default, the property is set to **true**.
ShowDialogsHelp	Sets a value which indicates that show or hide the help button in dialogs. By default, the property is set to **true**.
PageAlignment	Sets the position of the report page in the viewer window. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – the page will be aligned left; › **Center** – the page will be centered (default value); › **Right** – the page will be aligned right.
ShowPageShadow	Enables displaying shadow for report pages. By default the property is set to **true**.
BookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default the property is set to **false**.
BookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
ParametersPanelPosition	Specifies the position of the report parameters panel. It can take one of the following **StiParametersPanelPosition** enumeration values: › **Top** - the panel will be docked to the top margin (default value); › **Left** - the panel will be docked to the left margin.
ParametersPanelMaxHeight	Sets the maximum height of the parameters bar in pixels. By default, the maximum height is 300 pixels.
ParametersPanelColumnsCount	Sets the number of columns to display report

© 2003-2024 Stimulsoft

	parameters. By default, there are 2 columns.
ParametersPanelSortDataItems	Gets or sets a value which indicates that variable items will be sorted. By default the property is set to **true**.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
InterfaceType	Sets the type of interface used for the viewer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the viewer's interface is determined automatically depending of the device that is report is displayed on. That is the default value. › **Mouse** – the standard interface with a mouse control will be used for all the screen types. › **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability. › **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.
AllowMobileMode	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to **false**, then the mobile view will not be used. If the option is set to **true**, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option is set to **true**.
ChartRenderType	Sets the displaying mode of charts on the report

© 2003-2024 Stimulsoft

	page. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages. It can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
DatePickerFirstDayOfWeek	Sets the first day of the week for the date picker. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Auto** – automatic detection of the first day of the week from the browser settings (default value); › **Monday** – the first day of the week is Monday; › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default the property is set to **false**.
AllowTouchZoom	Sets ability to change the scale of the report page by using the two-fingers gesture (Pinch to

© 2003-2024 Stimulsoft

	Zoom) for the touch-screens. The default value of the property is **true**.
ShowReportIsNotSpecifiedMessage	Sets a value which indicates that 'The report is not specified' message will be shown. The default value of the property is **true**.
PrintToPdfMode	Sets the Print to PDF mode. It has the following values: › **StiPrintToPdfMode.Hidden** - hidden print mode (default value); › **StiPrintToPdfMode.Popup** - the PDF document will be displayed before printing in a pop-up window.
ImagesQuality	Gets or sets the image quality that will be used on the viewer page. It has the following values: › **StiImagesQuality.Low** - low quality, used to speed up loading reports and saves memory; › **StiImagesQuality.Normal** - normal quality, suitable for most cases (default value); › **StiImagesQuality.High** - high quality, used for ultra high-definition displays, but may slow down the loading of pages.
CombineReportPages	Sets a value which indicates that if a report contains several pages, then they will be combined in preview. By default the property is set to **false**.

Toolbar

Name	Description
Visible	Enables displaying the viewer toolbar. By default, the property is set to **true**.
DisplayMode	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the **StiToolbarDisplayMode** enumeration: › **Simple** - all controls are located on the same

© 2003-2024 Stimulsoft

	control panel (default value); › **Separated** - the control panel is split into top and bottom panels.
BackgroundColor	Specifies the background color of the viewer toolbar. The default color of the selected theme is used.
BorderColor	Specifies the border color of the viewer toolbar. The default color of the selected theme is used.
FontColor	Specifies the text color for the toolbar and the viewer menu. The default color of the selected theme is used.
FontFamily	Specifies the font for the toolbar and the viewer menu. The default font of the selected theme is used.
Alignment	Sets the alignment mode for the controls on the viewer toolbar. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – elements will be aligned left; › **Center** – elements will be centered; › **Right** – elements will be aligned right; › **Default** – the alignment depends on the RightToLeft property (default value).
ShowButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default the property is set to **true**.
ShowPrintButton	Enables showing the button - **Print** - on the viewer toolbar. By default, the property is set to **true**.
ShowOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports. By default, the property is set to true.

© 2003-2024 Stimulsoft

ShowSendEmailButton	Enables showing the button - **Send Email** - on the viewer toolbar. By default, the property is set to **false**. Also, you should add the EmailReport action.
ShowFindButton	Enables showing the button - **Find** - on the viewer toolbar. By default, the property is set to **true**.
ShowBookmarksButton	Enables showing the button - **Bookmarks** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
ShowParametersButton	Enables showing the button - **Parameters** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.
ShowResourcesButton	Enables showing the button - **Resources** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.
ShowEditorButton	Enables showing the button - **Editor** - on the viewer toolbar. By default, the property is set to **true**.
ShowFullScreenButton	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports. By default, the property is set to **true**.
ShowFirstPageButton	Enables showing the button - **First Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowPreviousPageButton	Enables showing the button - **Previous Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowCurrentPageControl	Enables showing the current report page indicator. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowNextPageButton	Enables showing the button - **Next Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowLastPageButton	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowZoomButton	Enables showing the button to select the report zoom. By default, the property is set to **true**.
ShowViewModeButton	Enables showing the button to select the view mode of the report page. By default, the property is set to **true**.
ShowDesignButton	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports. By default, the property is set to **false**.
ShowAboutButton	Enables showing the button - **About** - on the viewer toolbar. By default, the property is set to **true**.
ShowRefreshButton	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to **true**.
ShowPinToolbarButton	Enables displaying of the **Pin Toolbar** button on the viewer's toolbar. The button is available only in the Mobile mode of the viewer's interface. The default value of the property is **true**.
PrintDestination	Sets the report printing mode. It can take one of the following values of the **StiPrintDestination** enumeration: › **Default** – a menu with a choice of printing modes will be displayed (default value); › **Pdf** – printing will be done in the PDF format; › **Direct** – printing will be done to the HTML format directly to the printer, the system print dialog will be displayed; › **PopupWindow** – printing will be done in the

© 2003-2024 Stimulsoft

	HTML format via the preview window of the report.
ViewMode	Sets the mode for displaying report pages. It can take one of the following **StiWebViewMode** enumeration values: › **SinglePage** - displays one page of the report selected in the toolbar of the viewer (default value); › **Continuous** - displays all pages of the report; › **MultiplePages** - displays all report pages as a table.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of the following values: › **StiZoomMode.PageWidth** – when the viewer runs, the zoom, necessary to display the report by the page width, will be set; › **StiZoomMode.PageHeight** – when the viewer runs, the zoom, necessary to display the report by the page height, will be set.
MenuAnimation	Enables animation when the viewer menu shows/hides. By default the property is set to **true**.
ShowMenuMode	Sets the display mode of the viewer menu. It can take one of the following values of the **StiShowMenuMode** enumeration: › **Click** – shows menu by mouse click (default value); › **Hover** – shows menu by hovering the mouse cursor.
AutoHide	Enables auto-hiding of the viewer's toolbar. The property will work only for the Mobile mode of the viewer's interface. The default value of the

© 2003-2024 Stimulsoft

	property is **false**.

Export

Name	Description
DefaultSettings	This group of properties provides the ability to specify the default export settings for each export type. These settings will be applied to the export dialogs when the viewer runs or to the report, if export dialogs are disabled.
StoreExportSettings	Enables saving selected settings in the export dialogs. Settings will be stored in browser cookies. By default the property is set to **true**.
ShowExportDialog	Enables showing the export options dialog box. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
ShowExportToDocument	Enables the export menu item - **Document File**. By default, the property is set to **true**.
ShowExportToPdf	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToXps	Enables the export menu item - **Microsoft XPS File**. By default, the property is set to **false**.
ShowExportToPowerPoint	Enables the export menu item - **Microsoft PowerPoint 2007/2010 File**. By default, the property is set to **true**.
ShowExportToHtml	Enables the export menu item - **HTML File**. By default, the property is set to **true**.
ShowExportToHtml5	Enables the export menu item - **HTML5 File**. By default, the property is set to **true**.
ShowExportToMht	Enables the export menu item - **MHT Web Archive**. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToText	Enables the export menu item - **Text File**. By default, the property is set to **true**.
ShowExportToRtf	Enables the export menu item - **Rich Text File**. By default, the property is set to **true**.
ShowExportToWord2007	Enables the export menu item - **Microsoft Word 2007/2010 File**. By default, the property is set to **true**.
ShowExportToOpenDocumentWriter	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to **true**.
ShowExportToExcel	Enables the export menu item - **Microsoft Excel File**. By default, the property is set to **true**.
ShowExportToExcelXml	Enables the export menu item - **Microsoft Excel Xml File**. By default, the property is set to **true**.
ShowExportToExcel2007	Enables displaying the **Microsoft Excel 2007/2010 File** export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToOpenDocumentCalc	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to **true**.
ShowExportToCsv	Enables the export menu item - **CSV File**. By default, the property is set to **true**.
ShowExportToDbf	Enables the export menu item - **DBF File**. By default, the property is set to **true**.
ShowExportToXml	Enables the export menu item - **XML File**. By default, the property is set to **true**.
ShowExportToDif	Enables the export menu item - **Data Interchange Format (DIF) File**. By default, the property is set to **true**.
ShowExportToSylk	Enables the export menu item - **Symbolic Link (SYLK) File**. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToJson	Enables the export menu item - **JSON File**. By default, the property is set to **true**.
ShowExportToImageBmp	Enables displaying the **BMP Image** export menu item when viewing reports, and the **BMP Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageGif	Enables displaying the **GIF Image** export menu item when viewing reports, and the **GIF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageJpeg	Enables displaying the **JPEG Image** export menu item when viewing reports, and the **JPEG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePcx	Enables displaying the **PCX Image** export menu item when viewing reports, and the **PCX Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePng	Enables displaying the **PNG Image** export menu item when viewing reports, and the **PNG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageTiff	Enables displaying the **TIFF Image** export menu item when viewing reports, and the **TIFF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvg	Enables displaying the **Scalable Vector Graphics (SVG) File** export menu item when viewing reports, and the **Scalable Vector Graphics (SVG) File** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvgz	Enables displaying the **Compressed SVG (SVGZ) File** export menu item when viewing reports, and the **Compressed SVG (SVGZ) File** item when viewing dashboards. By default, the

© 2003-2024 Stimulsoft

	property is set to **true**.
ShowOpenAfterExport	Enables displaying the **Open After Export** parameter in export settings menu. By default, the property is set to **true**.

Email

Name	Description
ShowEmailDialog	Enables displaying settings for sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server side in the **EmailReport** action. By default the property is set to **true**.
ShowExportDialog	Enables displaying export options dialog box when sending email. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
DefaultEmailAddress	Sets the default recipient email, i.e. the address to which the email with the attached report will be sent.
DefaultEmailSubject	Sets the default email subject (header).
DefaultEmailMessage	Sets the default email message (text).
DefaultEmailReplyTo	Sets the default text of the replyTo of the message created in the viewer.

7.2 Designer

Samples

On GitHub, you may find the samples which show how to work with the **Blazor Designer** component. We support Blazor Server and Blazor WebAssembly. All the examples are separate projects, grouped into one solution for the Visual Studio.

© 2003-2024 Stimulsoft

The **Blazor Designer** component is designed to create reports in the web browser. And you don't need to set components or any special plugins on the client. All you need is a modern web browser. With the help of the **Blazor Designer**, you can create, edit, save, view reports, and print them on any computer with any installed operating system.

The **Blazor Designer** component is developed as a unique component, and it can work both with the Blazor Server and WebAssembly. When performing all actions on reports, the component requests only necessary data, which allows you to get rid of reloading the entire page and save Web traffic and increase work speed.

To use the **Blazor Designer** in a Web project, you should use the NuGet package Stimulsoft.Reports.Blazor or Stimulsoft.Dashboards.Blazor:

- Select the Manage NuGet Packages item in the context menu of the project;
- Specify the Stimulsoft.Reports.Blazor on the Browse in the search line;
- Select the element, define the package version, and click **Install**. You should click the **Update** when updating the package.

If for some reason it is not possible, you should add the following assemblies to the project:
Stimulsoft.Base.dll
Stimulsoft.Blockly.dll
Stimulsoft.Data.dll
Stimulsoft.Drawing.dll
Stimulsoft.Map.dll
Stimulsoft.Report.dll
Stimulsoft.Report.Blazor.dll
Stimulsoft.Report.Check.dll
Stimulsoft.Report.Helper.dll
Stimulsoft.Report.Web.dll
Stimulsoft.Report.WebDesign.dll
Stimulsoft.System.dll
Stimulsoft.System.Web.dll

You should additionally add the following assemblies to the project for using dashboards:
Stimulsoft.Dashboard.dll
Stimulsoft.Dashboard.Drawing.dll
Stimulsoft.Dashboard.Export.dll

© 2003-2024 Stimulsoft

- Activation
- Editing Reports
- Creating New Reports
- Saving Reports
- Settings
- Additional Features of Preview
- Preview
- Localization
- Using Themes
- Designer Events

7.2.1 Activation

After acquiring a Stimulsoft product, you should activate the license. You can do it by specifying a license key or loading a file with a license key. Below is an example of the **Blazor Designer** component activation.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner />

@code
{
   protected override void OnInitialized()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
      Stimulsoft.Base.StiLicense.LoadFromFile("Content/license.key");

      base.OnInitialized();
   }
}
```

You can get a license key or download a file with a license key in a personal account.

7.2.2 Editing Reports

To edit a report, you should add the **StiBlazorDesigner** component to the **Razor** page, add the **StiReport** object, and assign it to the designer, using the **Report** property. After loading a report from a file, it will be automatically transferred to the

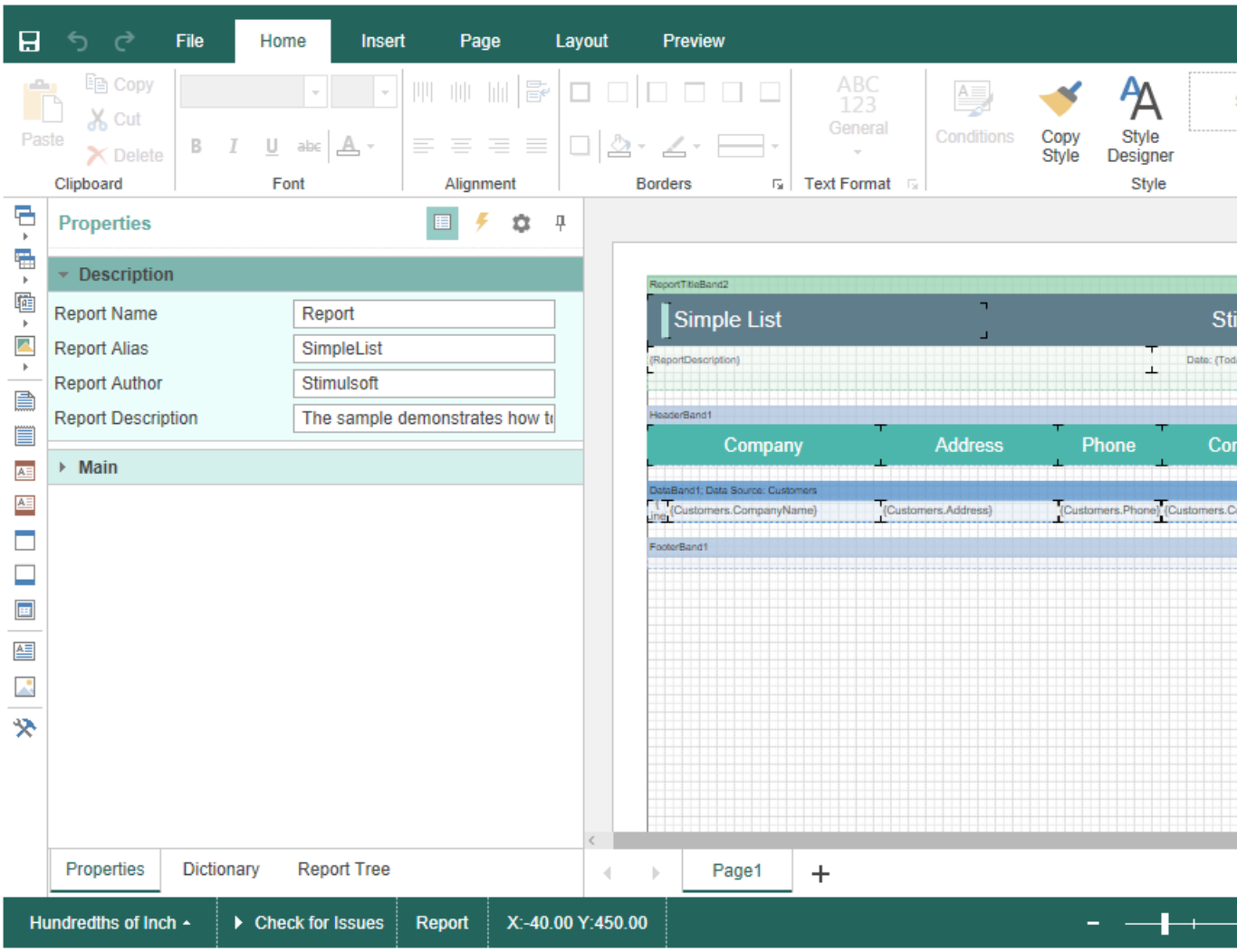
© 2003-2024 Stimulsoft

designer.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" />

@code
{
   //Report object to use in designer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Create empty report object
      var report = new StiReport();

      //Load report template
      report.Load("Reports/TwoSimpleLists.mrt");

      //Assing report object to viewer
      Report = report;
   }
}
```

© 2003-2024 Stimulsoft



7.2.3 Creating New Reports

It's enough to create a new object in the **OnInitialized** event to run the designer with a new report. If required, you can preload data for your report or perform some other necessary actions.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" />

@code
{
   //Report object to use in designer
   private StiReport Report;
```

© 2003-2024 Stimulsoft

```
    protected override void OnInitialized()
    {
        base.OnInitialized();

        //Create empty report object
        var report = new StiReport();

        //Assing report object to designer
        Report = report;
    }
}
```

Also, you can create a new report using the main menu of the designer. You can use the **OnCreateReport** event to preload data for a new report or perform other necessary actions. This event will be processed when creating a new blank report from the main menu or when creating a report using the wizard.

Index.razor

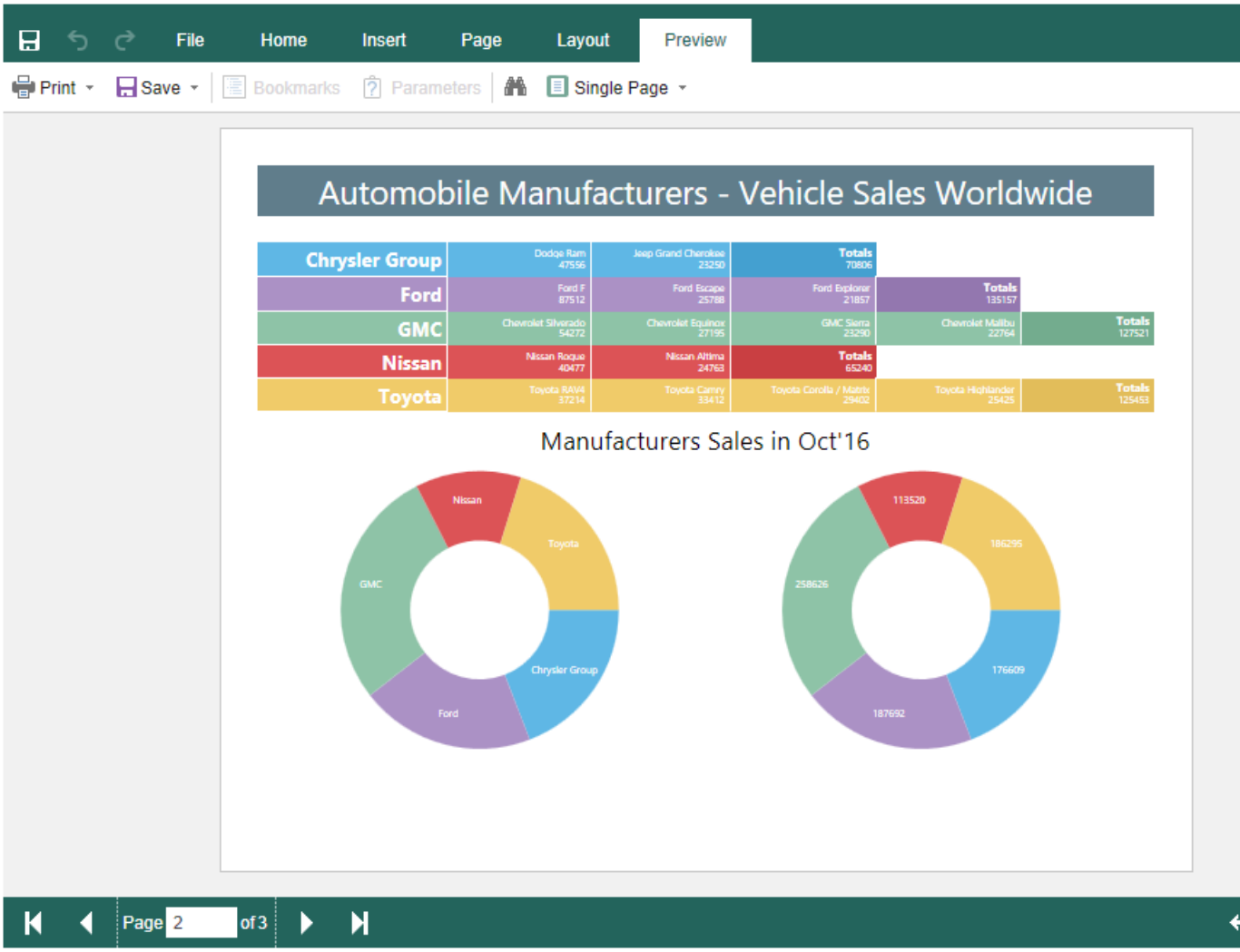
```
@using Stimulsoft.Report
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" OnCreateReport="@OnCreateReport"/>

@code
{
    //Report object to use in designer
    private StiReport Report;

    protected override void OnInitialized()
    {
        base.OnInitialized();

        //Create empty report object
        var report = new StiReport();

        //Load report template
        report.Load("Reports/Simple List.mrt");

        //Assing report object to designer
        Report = report;
    }

    private void OnCreateReport(StiReportDataEventArgs args)
    {
        //Delete connections in the report template
        args.Report.Dictionary.Databases.Clear();

        //Load new data from XML file
        var data = new System.Data.DataSet();
        data.ReadXml("Data/Demo1.xml");

        args.Report.RegData(data);
```

© 2003-2024 Stimulsoft

```
        args.Report.Dictionary.Synchronize();
    }
}
```

7.2.4 Preview

The **Blazor Designer** component has the mode of an editable report preview. To view a report, you should go to an appropriate bookmark in the designer window. The template will be rendered and displayed in the rendered viewer.



There is the ability to perform some necessary actions before previewing a report, for example: connect data for a report. To do it, you should define the **OnPreviewReport** event, which will be invoked before viewing a report.

Index.razor

© 2003-2024 Stimulsoft

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" OnPreviewReport="@OnPreviewReport"/>

@code
{
   //Report object to use in designer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Create empty report object
      var report = new StiReport();

      //Load report template
      report.Load("Reports/Simple List.mrt");

      //Assing report object to designer
      Report = report;
   }

   private void OnPreviewReport(StiReportDataEventArgs args)
   {
      //Load new data from XML file
      var data = new System.Data.DataSet();
      data.ReadXml("Data/Demo1.xml");

      args.Report.RegData(data);
   }
}
```

7.2.5 Additional Features of Preview

The window of report viewing of the **Blazor Designer** is a full-fledged interactive viewer, which can print and export a report. Besides, it supports the work with their parameters. Also, interactive actions are supported among them - dynamic sorting, drill-down, collapsing. To use these options, you don't need any additional settings of the report designer.

You can make manipulations with a report template in any of the above actions. For example, you can change properties and parameters, connect new data for rendering. When exporting a report, you can get the export format, read or change its settings.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
```

© 2003-2024 Stimulsoft

```
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" OnExportReport="@OnExportReport"/>

@code
{
   //Report object to use in designer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Create empty report object
      var report = new StiReport();

      //Load report template
      report.Load("Reports/Simple List.mrt");

      //Assing report object to designer
      Report = report;
   }

   private void OnExportReport(StiExportReportEventArgs args)
   {
      //Current export format
      var exportFormat = args.Format;

      //Current export settings
      var exportSettings = args.Settings;

      //Load new data from XML file
      var data = new System.Data.DataSet();
      data.ReadXml("Data/Demo1.xml");

      args.Report.RegData(data);
   }
}
```

Information

Suppose some of the specified additional options of the Report Preview are not required (for example, exporting or report printing). In that case, you can disable them using the appropriate options of the **Blazor Designer**.

7.2.6 Saving Reports

There are two options of saving reports in the **Blazor Designer** component, which are available in the main menu and the main panel of the designer - **Save** and **Save as**. At the same time, each of these options of saving has its modes and settings.

© 2003-2024 Stimulsoft

Saving reports on the server-side

To save an edited report on the server-side, you should define the **OnSaveReport** event, which will be invoked when selecting the **Save** item in the main menu or clicking the **Save** in the main panel of the designer.

Index.razor

```
@using Stimulsoft.Base
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" OnSaveReport="@OnSaveReport"/>

@code
{
   //Report object to use in designer
   private StiReport Report;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Create empty report object
      var report = new StiReport();

      //Load report template
      report.Load("Reports/TwoSimpleLists.mrt");

      //Assing report object to designer
      Report = report;
   }

   private void OnSaveReport(StiSaveReportEventArgs args)
   {
      args.Report.Save("Reports/TwoSimpleLists.mrt");
   }
}
```

The **OnSaveReport** event will be invoked when clicking **Save**. The edited report will be passed in the event arguments. If required, you can use the option of displaying the dialog window with an error or a text message after saving a report. The **ErrorCode** and **ErrorString** properties in the arguments of the event are used for this.

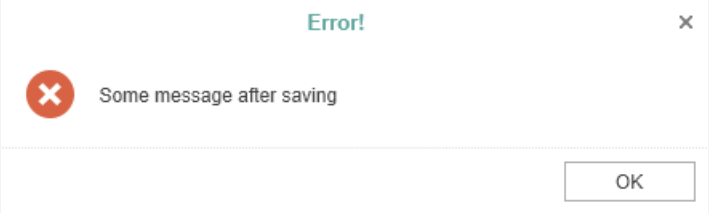
Index.razor

```
...
private void OnSaveReport(StiSaveReportEventArgs args)
{
```

© 2003-2024 Stimulsoft

```
    args.Report.Save("Reports/TwoSimpleLists.mrt");

    args.ErrorString = "Some message after saving";
}
...
```

In this case, the dialog window with a specified text will be displayed. The text can contain either a save error or a warning message or any other message.



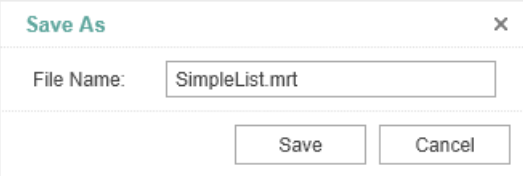
If required, you can access the original report name or report name from the save dialog.

Index.razor

```
...
private void OnSaveReport(StiSaveReportEventArgs args)
{
    //Report name from designer save dialog
    var savingReportName = args.FileName;

    //Original report name from properties
    var originalReportName = args.Report.ReportName;
}
...
```

Saving on the client-side

To save an edited report on the client-side as a file, you don't need additional designer settings. It's enough to select **Save as**, when clicking on it, the file saving dialog will be displayed. In this dialog, you can change the name of your report file, and after that, the file will be saved on the local disk of the computer.



© 2003-2024 Stimulsoft

The **Blazor Designer** component changes the behavior of the specified saving variant; there is the **OnSaveReportAs** event for this. When specifying this action, a report will be saved on the server-side; the work of this action will be analogous to the **OnSaveReportAs** event.

Index.razor

```
...
private void OnSaveReportAs(StiSaveReportAsEventArgs args)
{
   args.Report.Save("Reports/TwoSimpleLists.mrt");
}
...
```

Saving settings

A report is saved in the background mode, i.e., without reloading a page in the browser window. Suppose it is required to control the process of report saving somehow visually. In that case, you should change the **SaveReportMode** option (or the **SaveReportAsMode** option) to one of three values - **Hidden** (set by default), the **Visible**, or the **NewWindow**.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" Options="@Options" />

@code
{
   //Report object to use in designer
   private StiReport Report;

   private StiBlazorDesignerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorDesignerOptions();
      Options.Behavior.SaveReportMode = StiSaveMode.Visible;
      Options.Behavior.SaveReportAsMode = StiSaveMode.Visible;
   }
```

© 2003-2024 Stimulsoft

```
}
```

If the **SaveReportMode** option is set to **Visible**, report saving will be invoked in the current window of the browser in a simple (visible) mode with the help of the POST request. If the **SaveReportMode** option is set to the **NewWindow** value, the save report event will be invoked in a new browser window. This option is set to the **Hidden** value by default, i.e., the report saving is invoked in the background with the help of the AJAX request, and it is not displayed in the browser window. The same values and behavior are available for the **SaveReportAsMode** option.

7.2.7 Localization

The **Blazor Designer** component supports the full localization of its interface. The **Localization** property is used to localize the interface of the report designer. You should specify the path to the XML localization file (relative or absolute).

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Localization="Localization/en.xml" />
```

7.2.8 Using Themes

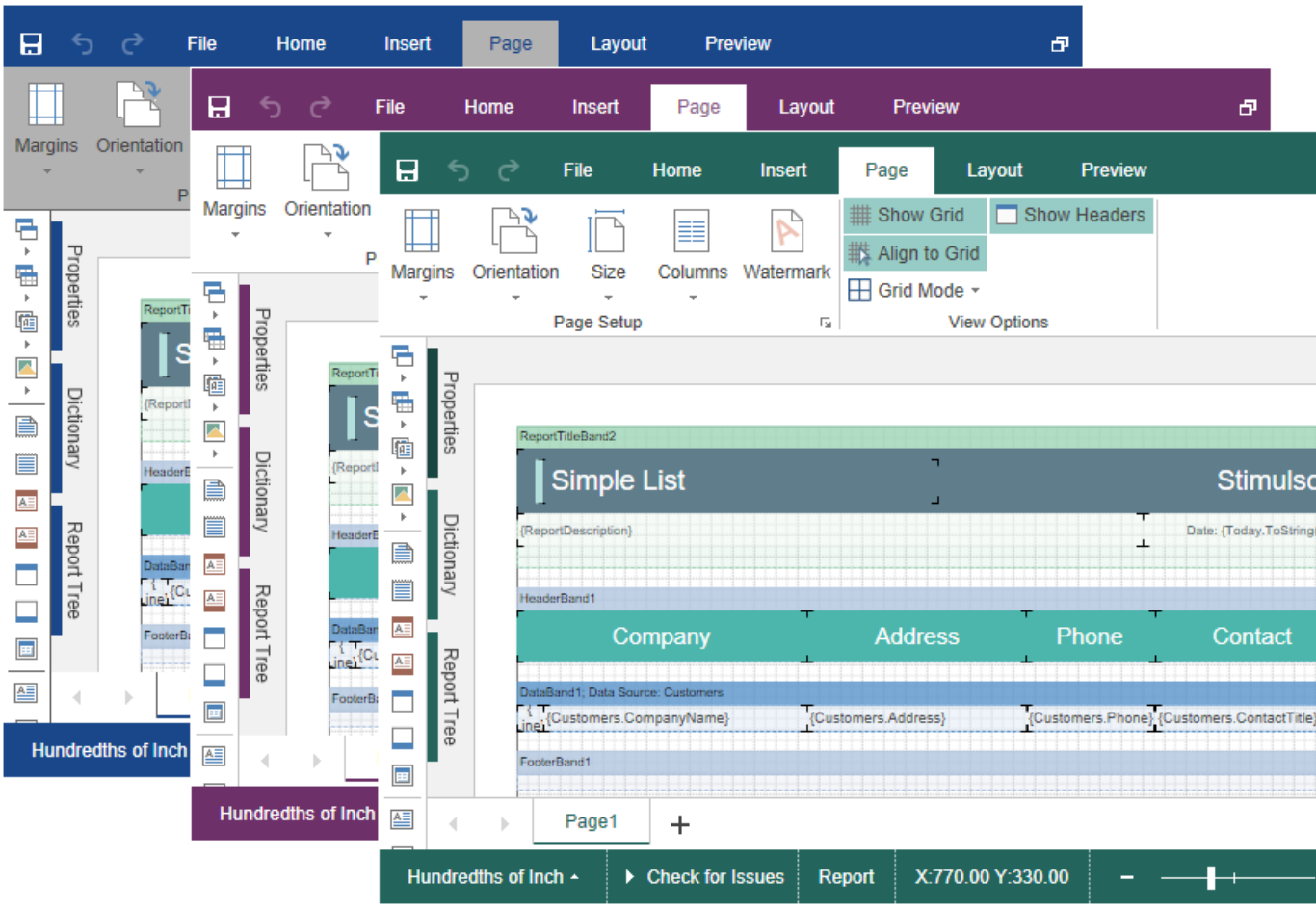
The **Blazor Designer** component has an option to change themes of visual controls. You should use the Theme property to change a theme.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Theme="StiDesignerTheme.Office2022WhiteCarmine" />
```

There are currently **2 themes** available with different color accents. As a result, **more than 50** variants of the appearance are available. This allows you to customize the appearance of the designer for almost any design of the Web project.

© 2003-2024 Stimulsoft



7.2.9 Designer Events

The **Blazor Designer** component supports events which allows you to execute necessary operations before certain actions, such as creating and editing report templates, previewing, printing and exporting, interactivity and etc. Below is a sample for processing designer events.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner Report="@Report" OnPreviewReport="@OnPreviewReport"/>

@code
{
   //Report object to use in designer
   private StiReport Report;

   protected override void OnInitialized()
   {
```

© 2003-2024 Stimulsoft

```
        base.OnInitialized();

        //Create empty report object
        var report = new StiReport();

        //Load report template
        report.Load("Reports/TwoSimpleLists.mrt");

        //Assing report object to designer
        Report = report;
    }

    private void OnPreviewReport(StiReportDataEventArgs args)
    {
        //Load new data from XML file
        var data = new System.Data.DataSet();
        data.ReadXml("Data/Demo1.xml");

        args.Report.RegData(data);
    }
}
```

Events list

Name	Description
OnCreateReport	Occurs when creating new reports from the designer menu.
OnOpenReport	The event occurs when you open a report from the designer menu. In the arguments of the event, the downloaded report will be sent.
OnPreviewReport	Occurs when going to the preview tab, as well as when interactive activities such as using report variables, dynamic collapsing, drill-down, and sorting a report when previewing it.
OnSaveReport	Occurs when clicking the Save button on the panel or from the main menu of the designer.
OnSaveReportAs	Occurs when clicking the Save As button from the main menu of the designer. If the event is not specified, the report will be saved to the local

© 2003-2024 Stimulsoft

	disk.
OnExportReport	Occurs when expoting reports.
OnDesignerEvent	The event occurs for any action in the report designer.
OnPrintReport	The event occurs when printing a report from the preview.
OnExit	The event occurs when clicking the **Exit** button in the main menu of the designer.
OnDesignerAfterRender	The event occurs when the HTML5 code of the designer and all its controls have been completed.

7.2.10 Settings

The **Blazor Designer** is configured with the help of the component properties. Basic settings are set at the **Razor** component; all additional settings are set with the help of the special **StiBlazorDesignerOptions** options class. Below is an example of the designer properties setting.

Index.razor

```
@using Stimulsoft.Report
@using Stimulsoft.Report.Blazor
@using Stimulsoft.Report.Web

<StiBlazorDesigner ID="Viewer1" Width="500px" Height="500px"
Localization="Localization/en.xml" Options="@Options"
Theme="StiDesignerTheme.Office2022WhiteCarmine" />

@code
{
   //Options object
   private StiBlazorDesignerOptions Options;

   protected override void OnInitialized()
   {
      base.OnInitialized();

      //Init options object
      Options = new StiBlazorDesignerOptions();
      Options.FileMenu.Visible = false;
      Options.Dictionary.Visible = false;
      Options.Toolbar.ShowToolbar = false;
      Options.Components.ShowText = false;
      Options.Bands.ShowEmptyBand = false;
      Options.DashboardElements.ShowTableElement = false;
```

© 2003-2024 Stimulsoft

```
        Options.Behavior.SaveReportMode = StiSaveMode.Hidden;
    }
}
```

The Designer properties

Name	Description
ID	Allows you to set a string unique identifier of a component, by default **StiBlazorDesigner**.
Options	Allows you to set an object of the **StiBlazorDesignerOptions** class which contains a set of options (settings) for a component.
Report	Allows you to set report object that will be displayed in the designer.
Theme	It specifies the theme of the designer. The list of available themes is in the **StiDesignerTheme** enumeration. The **Office2022WhiteBlue** value is set by default.
Localization	It specifies the path to the XML localization file. The path can be absolute or relative. English localization is used by default.
Width	It specifies the width of a component in required units – pixels (by default), percent, and points. For example, «500», «500px», «100%», «300pt». The component expands to the entire area of the browser window by default.
Height	It specifies the height of a component in required units – pixels (by default), percent, and points. For example, «500, » «500px, » «100%, » «300pt. » The component expands to the entire area of the browser window by default.

Server

Name	Description

© 2003-2024 Stimulsoft

UseRelativeUrls	It sets the designer mode, where relative links are used for requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value that specifies the port number to use in a URL. The 0 value defines automatic detection (set by default). The -1 value removes the port number.
PassFormValues	Enables the passing of the POST-form values to the client side, if these values are required to use in the designer actions. When enabling the feature, the **GetFormValues()** helper method will return a collection of form parameters. By default, the property is set to **false**.
PassQueryParametersForResources	It enables the transferring of all parameters of the URL request when generating links to the designer sources. If it is set to **false**, only necessary parameters are used to request the designer sources. It helps the browser work more correctly with the cache. By default, the property is set to **true**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Specifies the path to the CSS file of styles for the report designer. If this property is set, the standard styles of the selected theme will not be loaded. The default value is an empty value.
DefaultUnit	Sets the units for the size of the report and all its components. By default, centimeters are used.

© 2003-2024 Stimulsoft

Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the **StiZoomMode** enumeration: › **PageWidth** – when the designer runs, the zoom, necessary to display the report by the page width, will be set; › **PageHeight** – when the designer runs, the zoom, required to display the page height of the report, will be set.
ShowAnimation	Enables animation for various elements of the designer interface. By default, the property is set to **true**.
ShowOpenDialog	Allows to display the open dialog, or to open with the open event. By default, the property is set to **true**.
ShowTooltips	Enables displaying tooltips for designer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enable displaying links to online documentation in tooltips for designer controls. By default, the property is set to **true**.
ShowDialogsHelp	Enables displaying links to online documentation on the titles of dialog forms of the designer. By default, the property is set to **true**.
InterfaceType	Sets the type of interface used for the designer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the interface type of the designer will be selected automatically depending on the device used (default value); › **Mouse** – forced use of the interface to control the designer with the mouse; › **Touch** – forced use of the Touch interface to

© 2003-2024 Stimulsoft

	control the designer via the touch screen (mobile devices), also in this mode, the interface elements are increased.
DatePickerFirstDayOfWeek	Sets the first day of the week for the select date item. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Auto** – automatic detection of the first day of the week from the browser settings (default value); › **Monday** – the first day of the week is Monday; › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default, the property is set to **false**.
ShowReportTree	Enables displaying the tree of report components. By default, the property is set to **true**.
ReportDisplayMode	Sets the export mode for displaying report pages in the preview tab. Can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table; › **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
ChartRenderType	Gets or sets the type of the chart in the preview. It can take one of the following **StiChartRenderType** enumeration values:

© 2003-2024 Stimulsoft

	› **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
CloseDesignerWithoutAsking	Sets a value which indicates that the designer will be closed without asking. By default, the property is set to **true**.
ShowSystemFonts	Sets a visibility of the system fonts in the fonts list. By default, the property is set to **true**.
WizardTypeRunningAfterLoad	Calls the Report wizard after starting the report designer. It may have one of the following **StiWizardType** enumeration values: › **None** - runs the report designer without running the report wizard; › **StandardReport** - runs the Standard wizard; › **MasterDetailReport** - runs the Master-Detail wizard; › **LabelReport** - runs the Label wizard; › **InvoicesReport** - runs the Invoice wizard; › **OrdersReport** - runs the Order wizard; › **QuotationReport** - runs the Quote wizard.
AllowWordWrapTextEditors	Allows word wrap in the text editors. By default, the property is set to **true**.

Behavior

Name	Description
ShowSaveDialog	Enables displaying the dialog to insert a report

© 2003-2024 Stimulsoft

	name when it is saved. The name of the report will be transferred in the parameters of the report designer. By default, the property is set to **false**.
UndoMaxLevel	Sets the maximum number to cancel actions with the report (the Undo/Redo function). A big value of this property will consume memory on the server side to store the undo parameters. The default value is **6**.
AllowChangeWindowTitle	Allows using the title of the browser window to display the file name of the edited report. By default, the property is set to **true**.
SaveReportMode	Sets the mode for saving the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
SaveReportAsMode	Sets the mode for saving the report. It has the three values of the **StiSaveMode** enumeration. › **Hidden** - saving of the report is called in the background mode using the AJAX request and is not shown in the browser window (default value); › **Visible** - saving of the report is called in the current web browser window in the visible mode using the POST request; › **NewWindow** - saving of the report is called in a new window (tab) of the web browser.
CheckReportBeforePreview	Sets the value that allows running the report

© 2003-2024 Stimulsoft

	checker before preview.

FileMenu

Name	Description
Visible	Enables displaying the main menu of the report designer. By default, the property is set to **true**.
ShowNew	Enables showing the main menu item - **New**. By default, the property is set to **true**.
ShowFileMenuNewReport	Sets a visibility of the new report button in the designer. By default, the property is set to **true**.
ShowFileMenuNewDashboard	Sets a visibility of the new dashboard button in the designer. By default, the property is set to **true**.
ShowOpen	Enables showing the main menu item - **Open**. By default, the property is set to **true**.
ShowSave	Enables showing the main menu item - **Save**. By default, the property is set to **true**.
ShowSaveAs	Enables showing the main menu item - **Save As**. By default, the property is set to **true**.
ShowClose	Enables showing the main menu item - **Close**. By default, the property is set to **true**.
ShowExit	Enables showing the main menu item - **Exit**. By default, the property is set to **true**.
ShowReportSetup	Enables showing the main menu item - **Report Setup**. By default, the property is set to **true**.
ShowOptions	Enables showing the main menu item - **Options**. By default, the property is set to **true**.
ShowInfo	Enables showing the main menu item - **Info**. By default, the property is set to **true**.
ShowAbout	Enables showing the main menu item - **About**. By default, the property is set to **true**.
ShowHelp	Enables showing the main menu item - **Help**. By

© 2003-2024 Stimulsoft

	default, the property is set to **true**.

Dictionary

Name	Description
Visible	Enables showing the data dictionary of the report. By default, the property is set to **true**.
UseAliases	Allows you to use aliases in the data dictionary. It has the three values of the **StiUseAliases** enumeration: › **Auto** - defines the mode of using aliases from a saved value in cookies (default value); › **True** - sets the mode of using aliases in the data dictionary; › **False** - disables the mode of using aliases in the data dictionary.
NewReportDictionary	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It has the three values of the **StiNewReportDictionary** enumeration: › **Auto** - defines the mode to create or join the data dictionary from a saved value in cookies (default value); › **DictionaryNew** - sets the mode to create a new data dictionary when creating a new report; › **DictionaryMerge** - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
ShowDictionaryContextMenuProperties	Sets a visibility of the **Properties** item in the dictionary context menu. By default, the property is set to **true**.
ShowDictionaryActions	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

PermissionDataConnections	Sets the available actions to connect data to the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataSources	Sets available actions on report data sources. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataColumns	Sets the available actions on data columns in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionBusinessObjects	Sets the available actions on the business objects in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionDataRelations	Sets the available actions to linking data in the report. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionVariables	Sets available actions on report variables. It can take one or more values from the **StiDesignerPermissions** enumeration.
PermissionResources	Sets the available actions for the resources in the Report Dictionary. Takes one or several values from the **StiDesignerPermissions** enumeration.
PermissionSqlParameters	Sets the available actions for the parameters of the SQL queries for the Report DataSources. Takes one or several values from **StiDesignerPermissions** enumeration.
DataTransformationsPermissions	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

The table below shows all available values for the **StiDesignerPermissions** enumeration, which can be set for the dictionary elements of the report.

Value	Description

© 2003-2024 Stimulsoft

None	Disables any action on the item of the data dictionary.
All	Allows any action on the item of the data dictionary.
Create	Allows creating a specific data dictionary item.
Delete	Allows deleting a specific data dictionary item.
Modify	Allows modifying a specific data dictionary item.
View	Allows viewing a specific data dictionary item.
ModifyView	Allows modifying and viewing a specific data dictionary item.

Toolbar

Name	Description
ShowToolbar	Enables displaying the toolbar of the designer. By default, the property is set to **true**.
ShowSetupToolboxButton	Enables displaying the button to call the dialog box with settings for the side toolbar. By default, the property is set to **true**.
ShowInsertButton	Enables displaying the **Insert** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowLayoutButton	Enables displaying the tab **Layout** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPageButton	Enables displaying the tab **Page** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowPreviewButton	Enables displaying the tab **Preview** tab on the toolbar of the designer. By default, the property is set to **true**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the designer. By default, the property

© 2003-2024 Stimulsoft

	is set to **true**.
ShowAboutButton	Enables displaying the **About** on the toolbar of the designer. By default, the property is set to **false**.

PropertiesGrid

Name	Description
Visible	Enables displaying the property panel. By default, the property is set to **true**.
Width	Sets the width of the property panel. By default, the width is set to **370 px**.
LabelWidth	Specifies the width of the labels on the properties panel. By default, the width is set to **160 px**.
PropertiesGridPosition	Sets **Left** or **Right** position of the properties grid in the designer. It has the three values of the **StiPropertiesGridPosition** enumeration: › **Left**; › **Right**.
ShowPropertiesWhichUsedFromStyles	Sets a visibility of the properties which used from styles in the designer. By default, the property is set to **false**.

Components

Name	Description
ShowText	Enables displaying the **Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowTextInCells	Enables displaying the **Text in Cells** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowRichText	Enables displaying the **Rich Text** component in the insert menu for report components. By default, the property is set to **true**.
ShowImage	Enables displaying the **Image** component in the insert menu for report components. By default, the property is set to **true**.
ShowBarCode	Enables displaying the **Bar Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowShape	Enables displaying the **Shape** component in the insert menu for report components. By default, the property is set to **true**.
ShowHorizontalLinePrimitive	Enables displaying the **Horizontal Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowVerticalLinePrimitive	Enables displaying the **Vertical Line** component in the insert menu for report components. By default, the property is set to **true**.
ShowRectanglePrimitive	Enables displaying the **Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowRoundedRectanglePrimitive	Enables displaying the **Rounded Rectangle** component in the insert menu for report components. By default, the property is set to **true**.
ShowPanel	Enables displaying the **Panel** component in the insert menu for report components. By default, the property is set to **true**.
ShowClone	Enables displaying the **Clone** component in the insert menu for report components. By default, the property is set to **true**.
ShowCheckBox	Enables displaying the **Check Box** component in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowSubReport	Enables displaying the **Sub Report** component in the insert menu for report components. By default, the property is set to **true**.
ShowZipCode	Enables displaying the **Zip Code** component in the insert menu for report components. By default, the property is set to **true**.
ShowTable	Enables displaying the **Table** component in the insert menu for report components. By default, the property is set to **true**.
ShowCrossTab	Enables displaying the **Cross-Tab** component in the insert menu for report components. By default, the property is set to **true**.
ShowChart	Enables displaying the **Chart** component in the insert menu for report components. It affects on all chart types. By default, the property is set to **true**.
ShowMap	Enables displaying the **Map** component in the insert menu for report components. By default, the property is set to **false**.
ShowGauge	Enables displaying the **Gauge** component in the insert menu for report components. By default, the property is set to **false**.
ShowSparkline	Enables displaying the **Sparkline** component in the insert menu for report components. By default, the property is set to **true**.
ShowMathFormula	Enables displaying the **Math Formula** component in the insert menu for report components. By default, the property is set to **true**.
ShowElectronicSignature	Enables displaying the **Electronic Signature** component in the insert menu for report components. By default, the property is set to **true**.
ShowPdfDigitalSignature	Enables displaying the **PDF Digital Signature** component in the insert menu for report

© 2003-2024 Stimulsoft

	components. By default, the property is set to **true**.

Bands

Name	Description
ShowReportTitleBands	Enables displaying the **Report Title** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowReportSummaryBand	Enables displaying the **Report Summary** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageHeaderBand	Enables displaying the **Page Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowPageFooterBand	Enables displaying the **Page Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupHeaderBand	Enables displaying the **Group Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowGroupFooterBand	Enables displaying the **Group Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHeaderBand	Enables displaying the **Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowFooterBand	Enables displaying the **Footer** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnHeaderBand	Enables displaying the **Column Header** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowColumnFooterBand	Enables displaying the **Column Footer** item in

© 2003-2024 Stimulsoft

	the **Bands** menu of the designer. By default, the property is set to **true**.
ShowDataBand	Enables displaying the **Data** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowHierarchicalBand	Enables displaying the **Hierarchical** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowChildBand	Enables displaying the **Child** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowEmptyBand	Enables displaying the **Empty** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowOverlayBand	Enables displaying the **Overlay** item in the **Bands** menu of the designer. By default, the property is set to **true**.
ShowTableOfContents	Enables displaying the **Table of Contents** item in the **Bands** menu of the designer. By default, the property is set to **true**.

DashboardElements

Name	Description
ShowTableElement	Enables displaying the **Table** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowCardsElement	Enables displaying the **Cards** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowChartElement	Enables displaying the **Chart** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowGaugeElement	Enables displaying the **Gauge** element in the

© 2003-2024 Stimulsoft

	Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPivotTableElement	Enables displaying the **Pivot** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowIndicatorElement	Enables displaying the **Indicator** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowProgressElement	Enables displaying the **Progress** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowRegionMapElement	Enables displaying the **Region Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowOnlineMapElement	Enables displaying the **Online Map** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowImageElement	Enables displaying the **Image** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTextElement	Enables displaying the **Text** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowPanelElement	Enables displaying the **Panel** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowShapeElement	Enables displaying the **Shape** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowButtonElement	Enables displaying the **Button** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowListBoxElement	Enables displaying the **List Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowComboBoxElement	Enables displaying the **Combo Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewElement	Enables displaying the **Tree View** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowTreeViewBoxElement	Enables displaying the **Tree View Box** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.
ShowDatePickerElement	Enables displaying the **Date Picker** element in the Dashboard Elements menu of the designer. By default, the property is set to **true**.

CrossBands

Name	Description
ShowCrossGroupHeaderBand	Enables displaying the **Cross Group Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossGroupFooterBand	Enables displaying the **Cross Group Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossHeaderBand	Enables displaying the **Cross Header** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossFooterBand	Enables displaying the **Cross Footer** section in the insert menu for report components. By default, the property is set to **true**.
ShowCrossDataBand	Enables displaying the **Cross Data** section in the insert menu for report components. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

Dashboards

Name	Description
ShowNewDashboardButton	Sets a visibility of the **New Dashboard** button in the designer. By default, the property is set to **true**.

Pages

Name	Description
ShowNewPageButton	Sets a visibility of the **New Page** button in the designer. By default, the property is set to **true**.

When designing a report or dashboard in the report designer, you can also define **ExportOptions**, **EmailOptions**, and **PreviewToolbarOptions** on the **Preview** tab. These options are similar to the report viewer options.

8 Reports for Angular

.NET Core Angular is a cross-platform technology for creating Web applications for Windows, Linux and macOS. Stimulsoft provides tools for displaying reports using this technology.

8.1 Get Started

Step 1: Open Visual Studio;

Step 2: Click **File** menu, select **New** item, and choose **Project** item;

Step 3: Select **ASP.NET Core Web Application** command;

Step 4: Click **Next** button;

Step 5: Specify the name of this project, location and solution name. For example, **AngularViewer**;

Step 6: Click **Create** button;

© 2003-2024 Stimulsoft

Step 7: Select **Angular** type, set **.NET Core** as a framework and select **ASP.NET Core 3.1** and higher version;

Step 8: Uncheck **Configure for HTTPS** parameter;

Step 9: Click **Create** button;

Step 10: Install Stimulsoft.Reports.Angular.NetCore NuGet package:
- Select **Manage NuGet Packages ...** command in the context menu of the project;
- Specify **Stimulsoft.Reports.Angular.NetCore** in the search bar on the **Browse** tab;
- Select the item, define the version of the package, and click Install. When updating the package, click the **Update** button.

Step 11: Add **ViewerController** to Conntrollers folder:
- Call context menu of **Controllers** folder;
- Select **Add** item;
- Choose **Controller...** command;
- Set **MVC Controller - Empty** as controller type;
- Click **Add** button;
- Specify controller name. For example, **ViewerController**;
- Click the **Add** button.

Step 12: Create the **Reports** folder in the project, and add a report file to it. For example, add **MasterDetail.mrt** report.

Step 13: Add below code in **ViewerController.cs**.

ViewerController.cs

```
...
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Stimulsoft.Report;
using Stimulsoft.Report.Angular;
using Stimulsoft.Report.Web;

namespace AngularViewer.Controllers
   {
   [Controller]
```

© 2003-2024 Stimulsoft

```
  public class ViewerController : Controller
  {

    //Specify viewer options
    [HttpPost]
    public IActionResult InitViewer()
    {
      var requestParams = StiAngularViewer.GetRequestParams(this);
      var options = new StiAngularViewerOptions();
      options.Actions.ViewerEvent = "ViewerEvent";
      return StiAngularViewer.ViewerDataResult(requestParams, options);
    }

    //ViewerEvent() that will process viewer requests.
    [HttpPost]
    public IActionResult ViewerEvent()
    {
      var requestParams = StiAngularViewer.GetRequestParams(this);

      if (requestParams.Action == StiAction.GetReport)
      {
        var report = StiReport.CreateNewReport();
        var path = StiAngularHelper.MapPath(this, $"Reports/
        MasterDetail.mrt");
        report.Load(path);
        return StiAngularViewer.GetReportResult(this, report);
      }

      return StiAngularViewer.ProcessRequestResult(this);
    }
  }
}
...
```

Step 14: Open project folder in explorer;

Step 15: Install Angular Client Components from npm.

```
npm install stimulsoft-viewer-angular
```

Step 16: Close console;

Step 17: Delete the **ClientApp** folder;

Step 18: Go to the **AngularViewer** folder and open console;

© 2003-2024 Stimulsoft

Step 19: Enter **ng new ClientApp** command in console:

```
ng new ClientApp
```

Step 20: Enter '**N**' to not use routing;

Step 21: Select the **CSS** format and click Enter button;

Step 22: Close console and go to **ClientApp** folder;

Step 23: Open console and download **stimulsoft-viewer-angular**.

```
npm install stimulsoft-viewer-angular
```

Step 24: Close console;

Step 25: Open **app.module.ts** file in the editor from "**...ClientApp\src\app**" directory, and add **StimulsoftViewerModule**. Specify below code in **app.module.ts**:

app.module.ts

```
...
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { StimulsoftViewerModule } from 'stimulsoft-viewer-angular';
import { HttpClientModule } from '@angular/common/http';
import { BrowserAnimationsModule } from '@angular/platform-browser/
animations';
import { FormsModule } from '@angular/forms';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
```

© 2003-2024 Stimulsoft

```
    AppComponent
  ],
  imports: [
    BrowserModule,
    StimulsoftViewerModule
    HttpClientModule,
    BrowserAnimationsModule,
    FormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
...
```

Step 26: Open **app.component.html** file in the editor from "**...ClientApp\src\app**" directory, and add **AppComponent**. Specify below code in **app.component.html**:

app.component.html

```
...
<stimulsoft-viewer-angular
  [requestUrl]="'http://localhost:60801/Viewer/{action}'"
  [action]="'InitViewer'"
  [height]="'600px'"
></stimulsoft-viewer-angular>
...
```

Where:

- **[requestUrl]** is URL to ViewerController with {action} template. You can find out the URL to **ViewerController** in launchSettings.json file from the "**...\Properties**" directory.
- **[action]** is an action name to initialize an angular viewer.
- **[height]** is viewer height.

Step 27: Go to the visual studio and run the project. After the project run, you can see the viewer with a specific report.

Information

If your project does not start, try to delete node_modules folder and package-lock.json file from the "**...ClientApp**" directory.

© 2003-2024 Stimulsoft

8.2 Activation

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiAngularViewer** component.

ViewerController.cs

```
...
//Activation with using license code
public class ViewerController : Controller
{
   public ViewerController()
   {
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";
   }
}

//Activation with using license file
public class ViewerController : Controller
{
   public ViewerController(IWebHostEnvironment hostEnvironment)
   {
      var path = Path.Combine(hostEnvironment.ContentRootPath, "Content\
      \license.key");
      Stimulsoft.Base.StiLicense.LoadFromFile(path);
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

8.3 Angular Viewer

YouTube

Watch videos .NET Core Angular Viewer. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

Samples

© 2003-2024 Stimulsoft

See on GitHub examples for working with the Angular components. All examples are separate projects, grouped into one solution for Visual Studio.

The **Angular Viewer (StiAngularViewer)** component is designed to view reports in the web browser. You do not need to install the .NET Framework, ActiveX components or any special plug-ins on the client side. All that is needed is any modern Web browser.

With help of **Angular Viewer**, you can view, print and export reports on any computer with any operating system installed. Since the viewer only uses HTML and JavaScript technologies, it can be run on devices where there is no Flash or Silverlight support - tablets, smartphones. Also, the viewer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **Angular Viewer** component uses the AJAX technology to perform all actions (uploading a report, paging, scaling, interactivity in reports, etc.), which allows you to get rid of reloading the entire page, and save Web traffic and speed up work. The report engine built using the .NET Core technology is used to render reports. This is a cross-platform technology. It allows you to deploy the application on servers that use the operating systems like Windows, macOS, and Linux.

Angular Viewer supports many themes, animated interface, bookmarks, interactive reports, editing of report elements on the page, full screen mode, search panel, and other necessary features for viewing reports.

To use the **Angular Viewer** in a your projects, you need to install:

NetCore
To use the **Angular Viewer** in a Web project, you need to install the NuGet package of **Stimulsoft.Reports.Angular.NetCore**:
- Select "Manage NuGet Packages ..." in the context menu of the project;
- Specify Stimulsoft.Reports.Angular.NetCore in the search bar on the Browse tab;
- Select the item, define the version of the package, and click **Install**. When updating the package, click the **Update** button.

© 2003-2024 Stimulsoft

Angular

The easiest way is to install through npm:

```
npm install stimulsoft-viewer-angular
```


- How this works?
- Timeout
- Connecting data
- Localization
- Printing Reports
- Exporting Reports
- Viewing Modes
- Work with Parameters
- Work with Bookmarks
- Viewer Settings
- Showing Reports
- Editing Rendered Reports
- Sending Reports by Email
- Calling Designer from Viewer
- Caching
- Using Themes
- Basic Features
- Additional Methods
- Interactive Reports
- Creating your Angular application

8.3.1 How this Works

To run the viewer, you need to place the **stimulsoft-viewer-angular** component on the page, set the necessary settings to it, and set the necessary actions in the view controller. When the report viewer runs, the following actions occur:

› When the component is output, the JavaScript method is launched. It requests the first page of the report on the server side or the entire report (depending on the selected mode) and the required report parameters;

› Each action in the viewer (for example, paging, printing or exporting a report, etc.) calls a certain action on the server side, in which you can perform the necessary manipulations with the report;

› To speed up the work, the viewer saves the report in cache or server session, which makes it impossible to re-build the report.

8.3.2 Showing Reports

Notice

© 2003-2024 Stimulsoft

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `report.render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer.

To show the report, you need to add the **stimulsoft-viewer-angular** component to the template and set it to the minimum required properties: request URL template & action name.

component.ts

```
...
<stimulsoft-viewer-angular [requestUrl]="'http://server.url:51528/Viewer/{action}'" [action]="'Post'" ></stimulsoft-viewer-angular>
...
```

And in the view controller, specify the necessary actions.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult ViewerEvent()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);

   if (requestParams.Action == StiAction.GetReport)
   {
      return GetReport(reportName);
   }
   return StiAngularViewer.ProcessRequestResult(this);
}

public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiAngularHelper.MapPath(this, "Reports/SimpleList.mrt"));

   return StiAngularViewer.GetReportResult(this, report);
}
```

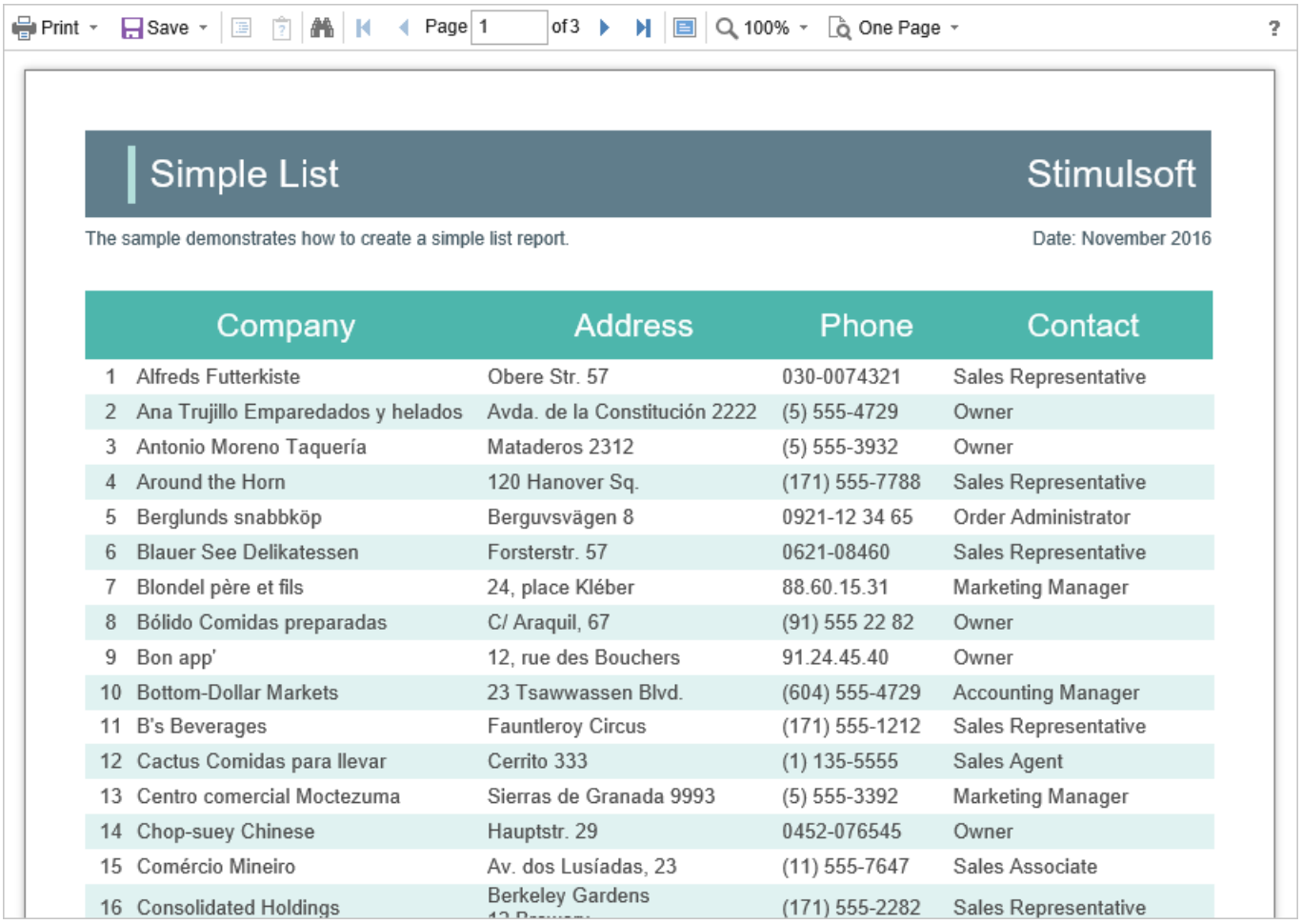
© 2003-2024 Stimulsoft

```
...
```

In the above example, processing of actions of the viewer is added. Angular viewer initialized by action configured in `[requestUrl] & [action]` here you need to specify **StiAngularViewerOptions** and **ViewerEvent** action. The **ViewerEvent** action handles the viewer events and if action is **GetReport** returns the report prepared for preview.

Information

The **ViewerEvent** action is mandatory. Without it, the correct operation of the viewer is not possible.



© 2003-2024 Stimulsoft

If the report was not rendered before showing, the **Angular Viewer** component will automatically render it. So you can use report templates and rendered reports to display reports.

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.LoadDocument(StiAngularHelper.MapPath(this, "Reports/
   SimpleList.mdc"));

   return StiAngularViewer.GetReportResult(this, report);
}
...
```

8.3.3 Connecting Data

Data to a report can be connected in various ways. The easiest way is to store connection settings in the report template. You can also connect the data from the code, this can be done when the report is loaded in the **GetReport** action.

HomeController.cs

```
...
public IActionResult GetReport()
{
   DataSet ds = new DataSet();
   ds.ReadXml(StiAngularHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(StiAngularHelper.MapPath(this, "Reports/
   TwoSimpleLists.mrt"));
   report.Dictionary.Databases.Clear();
   report.RegData("Demo", ds);

   return StiAngularViewer.GetReportResult(this, report);
}
...
```

Data for the report can be connected not only when the report is loaded. For example, you can connect new data at the moment of interactive actions in the viewer (applying report parameters, sorting, drill-down, collapsing). To do this, you should set the **Interaction** action for the **Angular Viewer** component, and, in the action handler, connect the data for the current report. The same way you can connect data in other actions of the viewer.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.Interaction = "ViewerInteraction";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   DataSet data = new DataSet();
   data.ReadXml(StiAngularHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = StiAngularViewer.GetReportObject(this);
   report.RegData("Demo", data);

   return StiAngularViewer.InteractionResult(this, report);
}
...
```

If you want to connect new data only for a certain interactive action of the viewer, for example, only when you apply report parameters, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class, they are passed to any server side on any request, and contain all necessary information and states of the client part of the viewer. To determine the type of the action of the viewer, it is enough to check the **Action** property of the viewer parameters.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiAngularViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      DataSet data = new DataSet();
      data.ReadXml(StiAngularHelper.MapPath(this, "Data/Demo.xml"));
```

© 2003-2024 Stimulsoft

```
    StiReport report = StiAngularViewer.GetReportObject(this);
    report.RegData("Demo", data);

    return StiAngularViewer.InteractionResult(this, report);
  }

  return StiAngularViewer.InteractionResult(this);
}
...
```

SQL data sources

The connection parameters to the SQL data source, as well as to any other ones, can be stored in the report template. If you want to set the connection parameters from the code before rendering the report (for example, for security reasons or depending on the authorized user), you can use the example below.

HomeController.cs

```
...
public IActionResult GetReport()
{
  OracleConnection connection = new OracleConnection("Data
  Source=Oracle8i;Integrated Security=yes");
  connection.Open();
  OracleDataAdapter adapter = new OracleDataAdapter();
  adapter.SelectCommand = new OracleCommand("SELECT * FROM Products",
  connection);

  DataSet dataSet = new DataSet("productsDataSet");
  adapter.Fill(dataSet, "Products");

  StiReport report = new StiReport();
  report.Load(StiAngularHelper.MapPath(this, "Reports/
  SqlSampleReport.mrt"));
  report.RegData("Products", dataSet);

  return StiAngularViewer.GetReportResult(this, report);
}
...
```

Also, for SQL data sources used in the report, you can specify the Query Timeout in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection,

© 2003-2024 Stimulsoft

and data sources in the report.

HomeController.cs

```
...
public IActionResult GetReport()
{

   StiReport report = new StiReport();
   report.Load(StiAngularHelper.MapPath("Report.mrt"));
   ((StiSqlDatabase)
   report.Dictionary.Databases["Connection"]).ConnectionString = @"Data
   Source=server;Integrated Security=True;Initial Catalog=DataBase";
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).SqlCommand = "select *
   from Table where Column = 100";
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiAngularViewer.GetReportResult(this, report);
}
...
```

Information

For SQL data sources of other types, the connection is created similarly, and an adapter corresponding to the type of the data source is connected. For example, for the MS SQL data source, you should connect SqlDataAdapter, for OracleDataAdapter is required for Oracle. Also, you should specify a connection string that matches the connection type.

The table below shows the connection string templates for different types of data sources.

Data Source	Connection String Template
MS SQL	Integrated Security=False; Data Source=myServerAddress;Initial Catalog=myDataBase; User ID=myUsername; Password=myPassword;
MySQL	Server=myServerAddress; Database=myDataBase;UserId=myUsername; Pwd=myPassword;

© 2003-2024 Stimulsoft

ODBC	Driver={SQL Server}; Server=myServerAddress;Database=myDataBase; Uid=myUsername; Pwd=myPassword;
OLE DB	Provider=SQLOLEDB.1; Integrated Security=SSPI;Persist Security Info=False; Initial Catalog=myDataBase;Data Source=myServerAddress
Oracle	Data Source=TORCL;User Id=myUsername;Password=myPassword;
MS Access	Provider=Microsoft.Jet.OLEDB.4.0;User ID=Admin;Password=pass;Data Source=C:\ \myAccessFile.accdb;
PostgreSQL	Server=myServerAddress; Port=5432; Database=myDataBase;User Id=myUsername; Password=myPassword;
Firebird	User=SYSDBA; Password=masterkey; Database=SampleDatabase.fdb;DataSource=myServerAddress; Port=3050; Dialect=3; Charset=NONE;Role=; Connection lifetime=15; Pooling=true; MinPoolSize=0;MaxPoolSize=50; Packet Size=8192; ServerType=0;
SQL CE	Data Source=c:\MyData.sdf; Persist Security Info=False;
SQLite	Data Source=c:\mydb.db; Version=3;
DB2	Server=myAddress:myPortNumber;Database=myDataBase;UID=myUsername;PWD=myPassword; Max Pool Size=100;Min Pool Size=10;
Infomix	Database=myDataBase;Host=192.168.10.10;Server=db_engine_tcp;Service=1492;Protocol=onsoctcp;UID=myUsername;Password=myPassword;
Sybase	Data Source=myASEserver;Port=5000;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
Teradata	Data Source=myServerAddress;User ID=myUsername;Password=myPassword;

© 2003-2024 Stimulsoft

VistaDB	Data Source=D:\folder \myVistaDatabaseFile.vdb4;Open Mode=ExclusiveReadWrite;
Universal(dotConnect)	Provider=Oracle;direct=true;data source=192.168.0.1;port=1521;sid=sid;user=user; password=pass
MongoDB	mongodb://<user>:<password>@localhost/test
OData	http://services.odata.org/v3/odata/OData.svc/

Information

The table shows the most commonly used templates for the connection string. You can view various connection string options at the special website.

Data from XML, JSON, Excel files

Connecting to XML and JSON data sources can be stored in the report template. If you want to specify data files from the code, you can use the example below.

HomeController.cs

```
...
public IActionResult GetReport()
{
   DataSet data = new DataSet();
   data.ReadXml(StiAngularHelper.MapPath(this, "Data/Demo.xml"));

   StiReport report = new StiReport();
   report.Load(StiAngularHelper.MapPath(this, "Reports/SimpleList.mrt"));
   report.RegData(data);

   return StiAngularViewer.GetReportResult(this, report);
}
...
```

HomeController.cs

```
...
public IActionResult GetReport()
{
   DataSet data
   = StiJsonToDataSetConverterV2.GetDataSetFromFile(StiAngularHelper.MapPat
```

© 2003-2024 Stimulsoft

```
    h(this, "Data/Demo.json"));

    StiReport report = new StiReport();
    report.Load(StiAngularHelper.MapPath(this, "Reports/SimpleList.mrt"));
    report.RegData(data);

    return StiAngularViewer.GetReportResult(this, report);
}
...
```

Information

The viewer has the possibility of obtaining data from an Excel file. To do this, you can use the following method.

```
DataSet dataSet = StiExcelConnector.Get().GetDataSet(new StiExcelOptions(ar
```

8.3.4 Localization

The **Angular Viewer** component supports the complete localization of its interface. To localize the report viewer interface, use the special **Localization** property. The value of this property should specify the path to the localization XML file (relative or absolute).

HomeController.cs

```
...
public IActionResult InitViewer()
{
    var requestParams = StiAngularViewer.GetRequestParams(this);
    var options = new StiAngularViewerOptions();
    options.Actions.ViewerEvent = "ViewerEvent";
    options.Localization = "Localization/en.xml";

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

When you load the report viewer, the localization file will be loaded automatically.

8.3.5 Using Themes

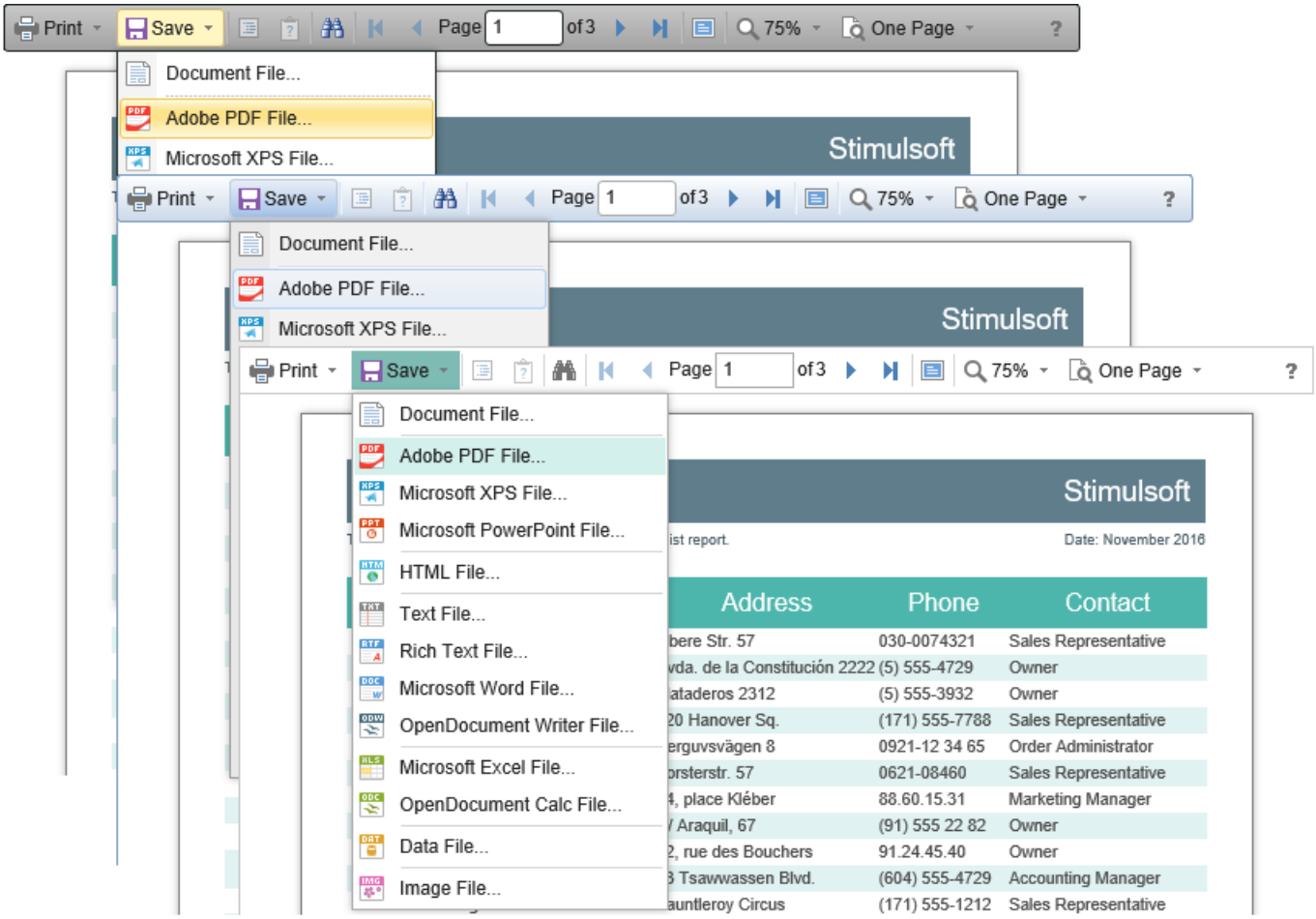
The **Angular Viewer** component can change the appearance of visual controls. To change the theme, use the **Theme** property, which can take one of the values of the **StiViewerTheme** enumeration.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Theme = StiViewerTheme.Office2022WhiteTeal;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.



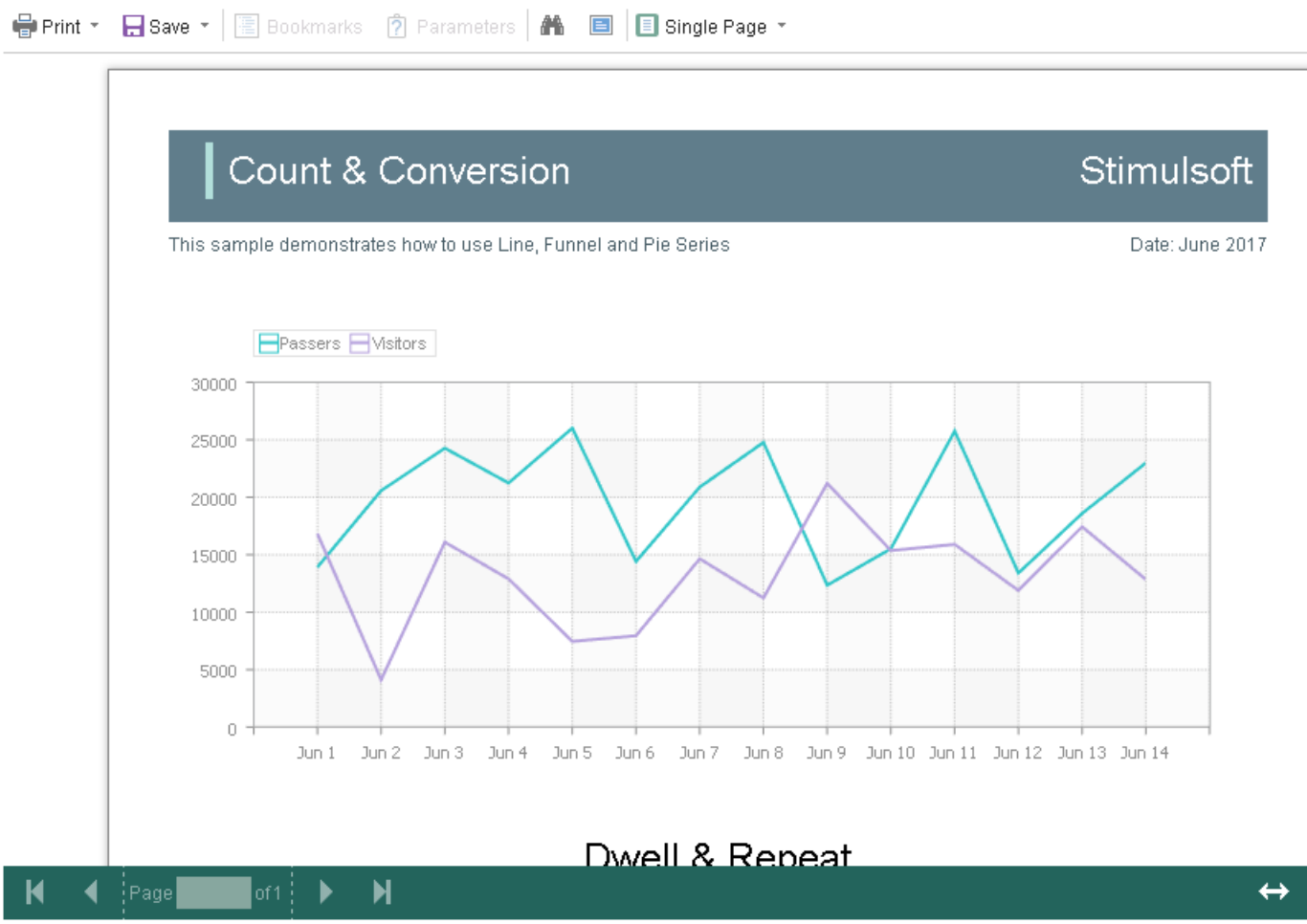
© 2003-2024 Stimulsoft

By default, the viewer has only the top toolbar on which all the report controls are located. If necessary, the toolbar can be split into top and bottom parts. The top panel will contain the menu for printing and exporting the report, as well as the buttons for working with parameters and bookmarks. The bottom panel will contain controls to switch between the report pages and setting zoom of pages. To enable this mode, enable the **ToolbarDisplayMode** property. It has values **Simple** and **Separated**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.ScrollbarsMode = true;
   options.Toolbar.DisplayMode = StiToolbarDisplayMode.Separated;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

© 2003-2024 Stimulsoft



In addition, it is possible to set the appearance parameters for the main elements of the viewer. For example, you can change the font and color of the control panel inscriptions of the viewer, set the background of the viewer, set the color of page borders, etc. Below is a list of available properties that change the appearance of the viewer, and their default values.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.BackgroundColor = Color.White;
   options.Appearance.PageBorderColor = Color.Blue;
   options.Appearance.ShowPageShadow = true;
   options.Toolbar.BackgroundColor = Color.White;
   options.Toolbar.BorderColor = Color.Gray;
```

© 2003-2024 Stimulsoft

```
    options.Toolbar.FontColor = Color.Black;
    options.Toolbar.FontFamily = "Arial";

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

8.3.6 Basic Features

The main features of the viewer include the following operations: displaying the report, switching between the report pages, changing the scale and displaying the preview mode. All specified operations are performed in the AJAX mode without restarting the browser page. For the correct work of these operations, you should define a special **ViewerEvent** action.

HomeController.cs

```
...
public IActionResult InitViewer()
{
    var requestParams = StiAngularViewer.GetRequestParams(this);
    var options = new StiAngularViewerOptions();
    options.Actions.ViewerEvent = "ViewerEvent";

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult ViewerEvent()
{
    // Some code before viewer event
    // ...

    return StiAngularViewer.ViewerEventResult(this);
}
...
```

Information

This action is mandatory. Without it, the correct operation of the viewer is not possible.

The **ViewerEvent** action returns a prepared HTML page of the report (or set of pages), built taking into account the current state of the viewer. If necessary, you can change the parameters of the current report in the specified action, as well as update the report data in case of interactive actions of the viewer.

© 2003-2024 Stimulsoft

8.3.7 Printing Reports

The **Angular Viewer** component provides several options for printing a report. Each has its own advantages and disadvantages.

Print to PDF
Printing will be done by exporting the report to the **PDF format**. The advantages are greater accuracy of positioning and printing of the report elements in comparison with other printing options. Among the drawbacks, one can mention the mandatory presence of a plug-in installed in a web browser for viewing PDF files (modern browsers have embedded PDF viewer and printer).

Print with Preview
The report will be printed in a separate pop-up browser window in the **HTML format**. The report can be previewed, and then sent to the printer or copied to another location in as text or HTML code. Advantages - cross-browser compatibility when printing, no need to install special plug-ins. The disadvantage is the relatively low accuracy of the position of the report elements, due to the peculiarities of the implementation of HTML formatting.

Print without Preview
The report will be printed directly to the printer without preview. After selecting this menu item, the system print dialog is displayed. Since printing in this mode is carried out in the **HTML format**, then the print quality is similar to the quality of printing a report with a preview.

Information

When printing to the **HTML format**, you should check the compliance of report page settings and printer parameters (paper size, orientation, margins, indents), as well as check your browser print settings, such as margins, headers, footers, watermarks printing, color printing.

The print function does not require additional settings for the viewer. If you need to perform any actions before printing a report, you can define a special **PrintReport** action.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.PrintReport = "PrintReport";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult PrintReport()
{
   // Some code before print
   // ...

   return StiAngularViewer.PrintReportResult(this);
}
...
```

Print setup

If you choose printing a report in the viewer panel, a menu with printing options is displayed. The **Angular Viewer** component is able to force the required printing mode. To do this, set the **PrintDestination** property to one of the following values of the **StiPrintDestination** enumeration.

- **Default** – the menu will be displayed (the default property value);
- **Pdf** – print to the PDF format;
- **Direct** – printing to the HTML format directly to the printer, the system print dialog will be displayed;
- **WithPreview** – print to the HTML format with preview in a pop-up window.

Index.cshtml

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Toolbar.PrintDestination = StiPrintDestination.Default;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

© 2003-2024 Stimulsoft

The **Angular Viewer** component is able to completely disable report printing. To do this, set the value of the **ShowPrintButton** property to **false**.

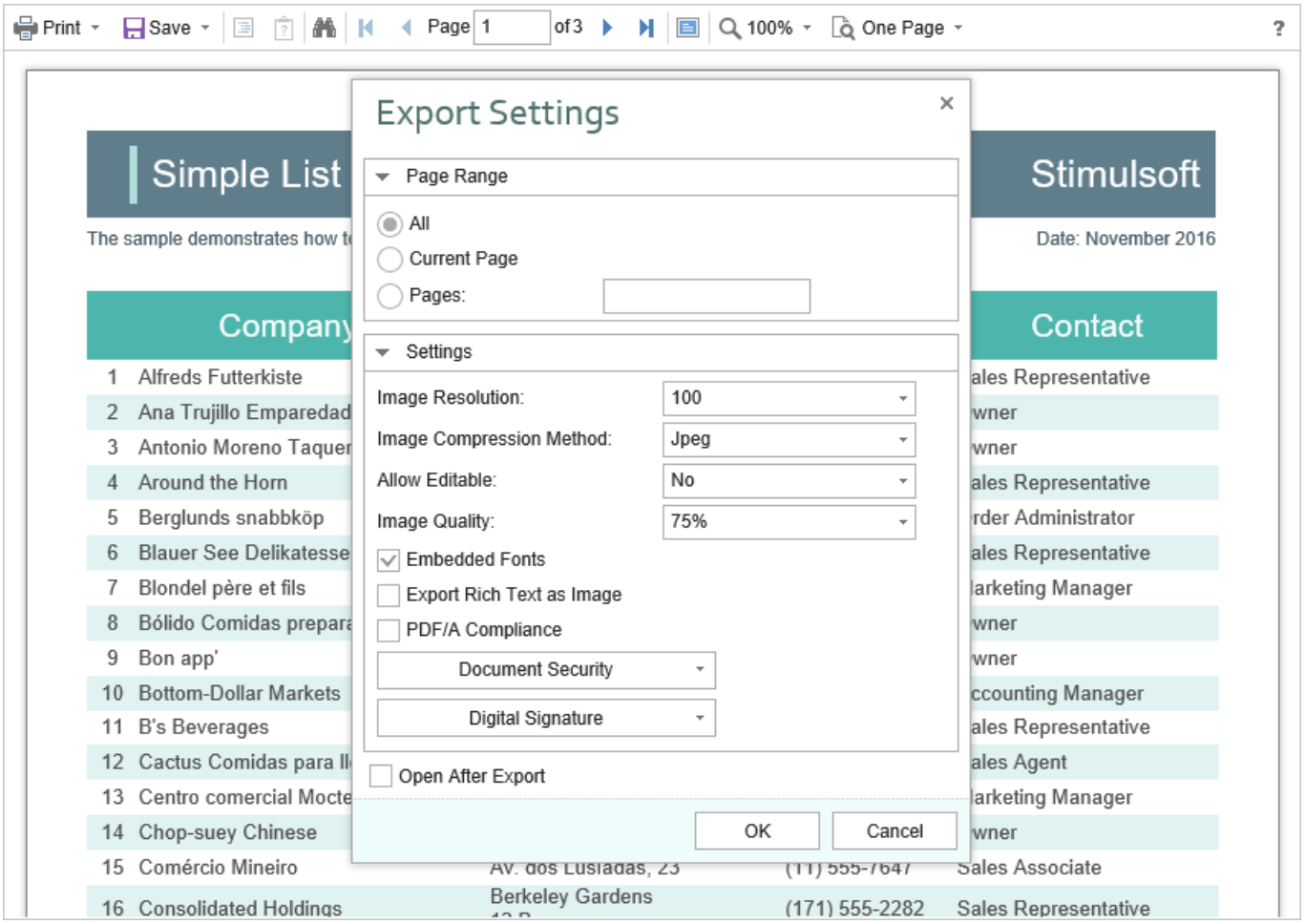
Index.cshtml

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Toolbar.ShowPrintButton = false;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

8.3.8 Exporting Reports

The **Angular Viewer** component allows you to export the displayed report to three dozen of various formats, such as **PDF**, **HTML**, **Word**, **Excel**, **XPS**, **RTF**, images, text and others.

© 2003-2024 Stimulsoft



The export function does not require additional settings for the viewer. If you need to perform any actions before exporting the report, you can define a special **ExportReport** action.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.ExportReport = "ExportReport";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult ExportReport()
{
   // Some code before export
```

© 2003-2024 Stimulsoft

```
    // ...

    return StiAngularViewer.ExportReportResult(this);
}
...
```

Export settings

Each report export format of the **Angular Viewer** component has a lot of settings, and each setting has its own default values. Sometimes you need to set other default values. For this purpose, a special **DefaultSettings** property of the viewer is used. It is a container for all the default export settings.

HomeController.cs

```
...
public IActionResult InitViewer()
{
    var requestParams = StiAngularViewer.GetRequestParams(this);
    var options = new StiAngularViewerOptions();
    options.Actions.ViewerEvent = "ViewerEvent";
    options.Exports.DefaultSettings.ExportToPdf.ImageQuality = 0.75f;
    options.Exports.DefaultSettings.ExportToPdf.ImageFormat =
    Stimulsoft.Report.Export.StiImageFormat.Color;
    options.Exports.DefaultSettings.ExportToHtml.ExportMode =
    Stimulsoft.Report.Export.StiHtmlExportMode.Div;
    options.Exports.DefaultSettings.ExportToHtml.UseEmbeddedImages = true;

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

If it is required, you can completely hide export dialogs. Exporting will always be done with default settings. For this, it is enough to set the value of the **ShowExportDialog** property to **false**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
    var requestParams = StiAngularViewer.GetRequestParams(this);
    var options = new StiAngularViewerOptions();
    options.Actions.ViewerEvent = "ViewerEvent";
    options.Exports.ShowExportDialog = false;

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
```

© 2003-2024 Stimulsoft

```
...
```

The **Angular Viewer** component contains 30+ export formats, and sometimes you need to disable unwanted formats. This allows you to simplify UI and the use of the viewer. To disable unused export formats, it is enough to set the values for the corresponding properties of the viewer listed in the list below to **false**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Exports.ShowExportToDocument = true;
   options.Exports.ShowExportToPdf = true;
   options.Exports.ShowExportToXps = true;
   options.Exports.ShowExportToPowerPoint = true;
   options.Exports.ShowExportToHtml = true;
   options.Exports.ShowExportToHtml5 = true;
   options.Exports.ShowExportToMht = true;
   options.Exports.ShowExportToText = true;
   options.Exports.ShowExportToRtf = true;
   options.Exports.ShowExportToWord2007 = true;
   options.Exports.ShowExportToOpenDocumentWriter = true;
   options.Exports.ShowExportToExcel = true;
   options.Exports.ShowExportToExcelXml = true;
   options.Exports.ShowExportToExcel2007 = true;
   options.Exports.ShowExportToOpenDocumentCalc = true;
   options.Exports.ShowExportToCsv = true;
   options.Exports.ShowExportToDbf = true;
   options.Exports.ShowExportToXml = true;
   options.Exports.ShowExportToDif = true;
   options.Exports.ShowExportToSylk = true;
   options.Exports.ShowExportToImageBmp = true;
   options.Exports.ShowExportToImageGif = true;
   options.Exports.ShowExportToImageJpeg = true;
   options.Exports.ShowExportToImagePcx = true;
   options.Exports.ShowExportToImagePng = true;
   options.Exports.ShowExportToImageTiff = true;
   options.Exports.ShowExportToImageMetafile = true;
   options.Exports.ShowExportToImageSvg = true;
   options.Exports.ShowExportToImageSvgz = true;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

The **Angular Viewer** component can completely disable the export menu. To do

© 2003-2024 Stimulsoft

this, set the value of the **ShowSaveButton** property to **false**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Toolbar.ShowSaveButton = false;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

8.3.9 Viewing Modes

The **Angular Viewer** component has two modes for displaying reports - with and without scrollbars. By default, the view mode without scrollbars is set. To enable the scrollbar view mode, set the value of the **ScrollbarsMode** property to **true**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.ScrollbarsMode = true;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

In the first mode (without scrollbars), the viewer displays a page or report as a whole, automatically stretching the preview space. If the width and height are specified, the viewer will truncate the page that is out of bounds. In the second mode, unlike the first one, when the page is out of bounds of the viewer's size, no truncation will be performed. Scrollbars will appear, using which you can view the entire page or report.

Information

In the report mode with scrollbars, you should set the height of the viewer,

© 2003-2024 Stimulsoft

otherwise the default height will be set to **650 pixels**.

```
<stimulsoft-viewer-angular [height]="'500px'" …
```

The **Angular Viewer** component provides the full-screen report mode. By default, the standard view mode is enabled, the viewer has the specified dimensions in the settings. To enable the full-screen mode, set the **FullScreenMode** property to **true**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance. FullScreenMode = true;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

Also, to enable or disable the full-screen mode, you can use the corresponding button on the control panel of the viewer.

The **Angular Viewer** component has three modes to display reports - page-by-page, entire report, and tabular display of report pages. To control the modes, the **ViewMode** property is used. It can have one of the specified values - **SinglePage**, **Continuous**, **MultiplePages**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Toolbar.ViewMode = StiWebViewMode.SinglePage;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

© 2003-2024 Stimulsoft

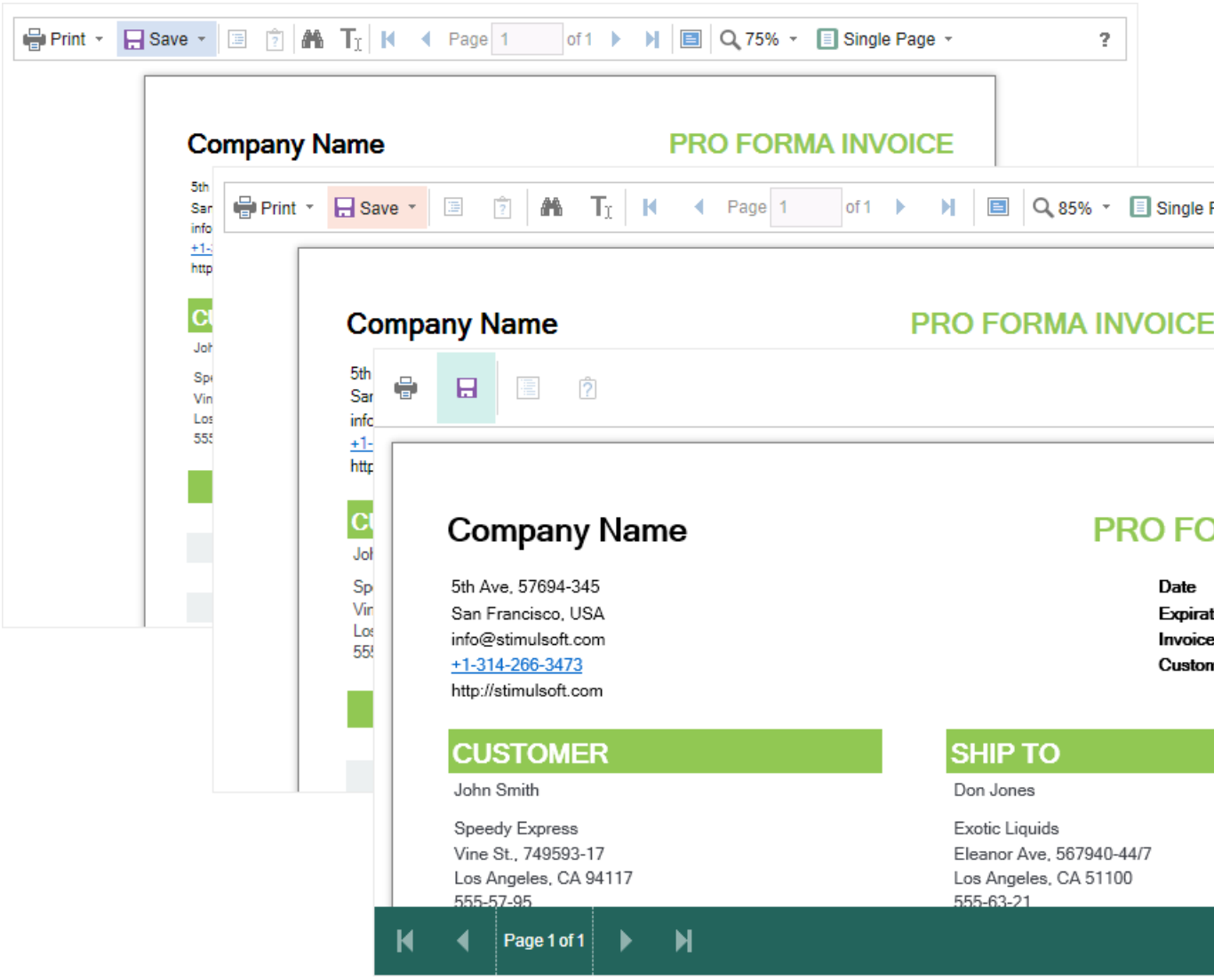
Angular Viewer component supports interaction on a regular PC display and work with a touchscreen of screens and of the mobile devices. **InterfaceType** property allows to control the interface modes. The property can have one of the following values:

- **Auto** – the viewer's interface is determined automatically depending of the device that is report is displayed on. That is the default value.
- **Mouse** – the standard interface with a mouse control will be used for all the screen types.
- **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability.
- **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.InterfaceType = StiInterfaceType.Auto;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

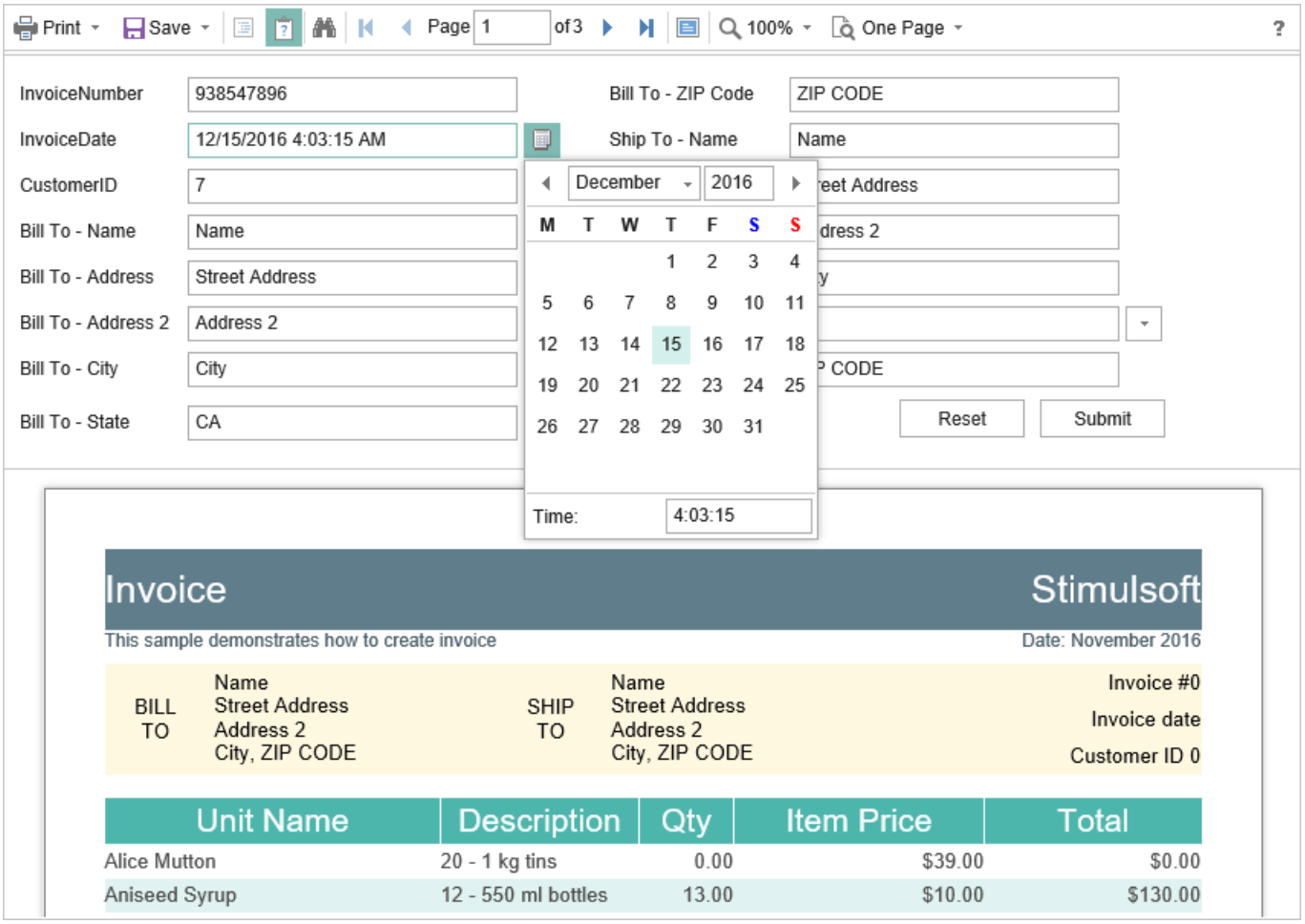
© 2003-2024 Stimulsoft



8.3.10 Work with Parameters

To work with report parameters in the **Angular Viewer**, there is a special settings panel. To add a parameter to the panel you need to define a variable in a report, requested by the user. When viewing a report in the viewer such a variable will be automatically added to the settings panel. It supports all types of report variables (normal variables, date and time, borders, lists, etc.).

© 2003-2024 Stimulsoft



To work with reports with parameters, no additional viewer settings are required. If you need to perform some actions before applying the parameters, you can define a special **Interaction** action.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.Interaction = "ViewerInteraction";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult ViewerInteraction()
{
   // Some code before any interaction
```

© 2003-2024 Stimulsoft

```
    // ...

    return StiAngularViewer.InteractionResult(this);
}
...
```

This action is called during any interactive actions of the viewer. If you need to perform any actions only when applying report parameters, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class, they are passed to any server side on any request, and contain all necessary information and states of the client part of the viewer. To determine the type of the action of the viewer, it is enough to check the **Action** property of the viewer parameters.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
    StiRequestParams requestParams =
    StiAngularViewer.GetRequestParams(this);
    if (requestParams.Action == StiAction.Variables)
    {
        // Some code before apply parameters
    }

    return StiAngularViewer.InteractionResult(this);
}
...
```

If you do not need to work with parameters, you can completely disable this feature. To do this, use the **ShowParametersButton** property in the **Toolbar** section of properties, which should be set to **false**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
    var requestParams = StiAngularViewer.GetRequestParams(this);
    var options = new StiAngularViewerOptions();
    options.Actions.ViewerEvent = "ViewerEvent";
    options.Toolbar.ShowParametersButton = false;

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

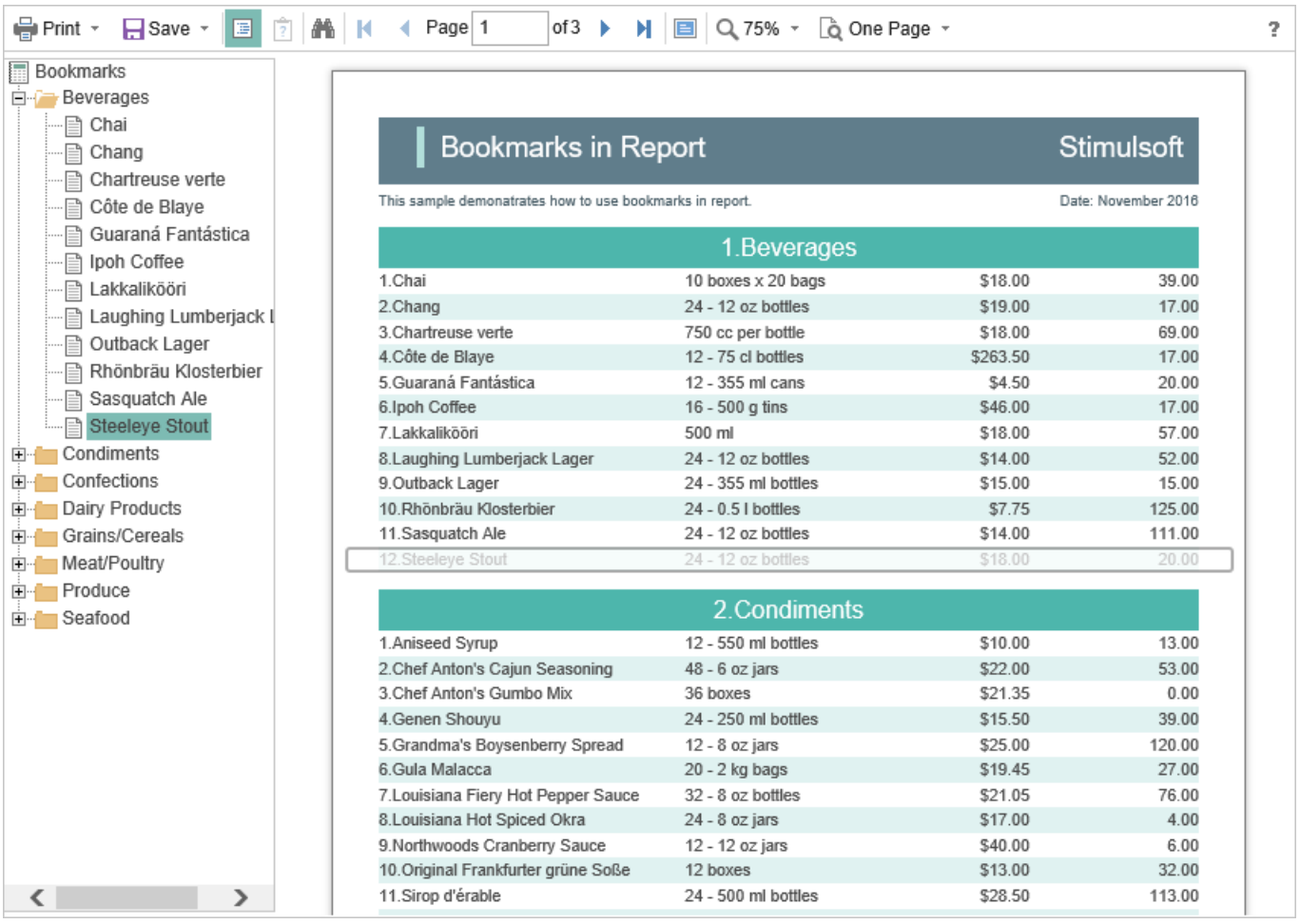
© 2003-2024 Stimulsoft

Information

With such a viewer configuration, the options panel will not be displayed, even if the parameters are present in the displayed report.

8.3.11 Work with Bookmarks

The **Angular Viewer** component supports report bookmarks. When displaying such a report on the left side of the page, a panel with bookmarks will be displayed. When you select a bookmark of the report, the viewer will carry out an automatic transition to the specified page, and the report item with a bookmark is highlighted.



By default, the bookmarks bar width is 180 pixels, the **Angular Viewer** component

© 2003-2024 Stimulsoft

allows you to change this value. For this, the **BookmarksTreeWidth** property, which value is specified in pixels, is used.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.BookmarksTreeWidth = 200;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

If work with report bookmarks is not required, you can disable this feature. For this, set the **ShowBookmarksButton** property to **false**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Toolbar.ShowBookmarksButton = false;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

Information

In this case, report bookmarks will not be displayed, even if they are present in the displayed report. This property has no effect on printing and exporting reports.

When printing a report with bookmarks, the bookmark tree will be hidden. If you want to print bookmarks with the report, it is necessary to set the **BookmarksPrint** property to **true**.

© 2003-2024 Stimulsoft

HomeController.cs

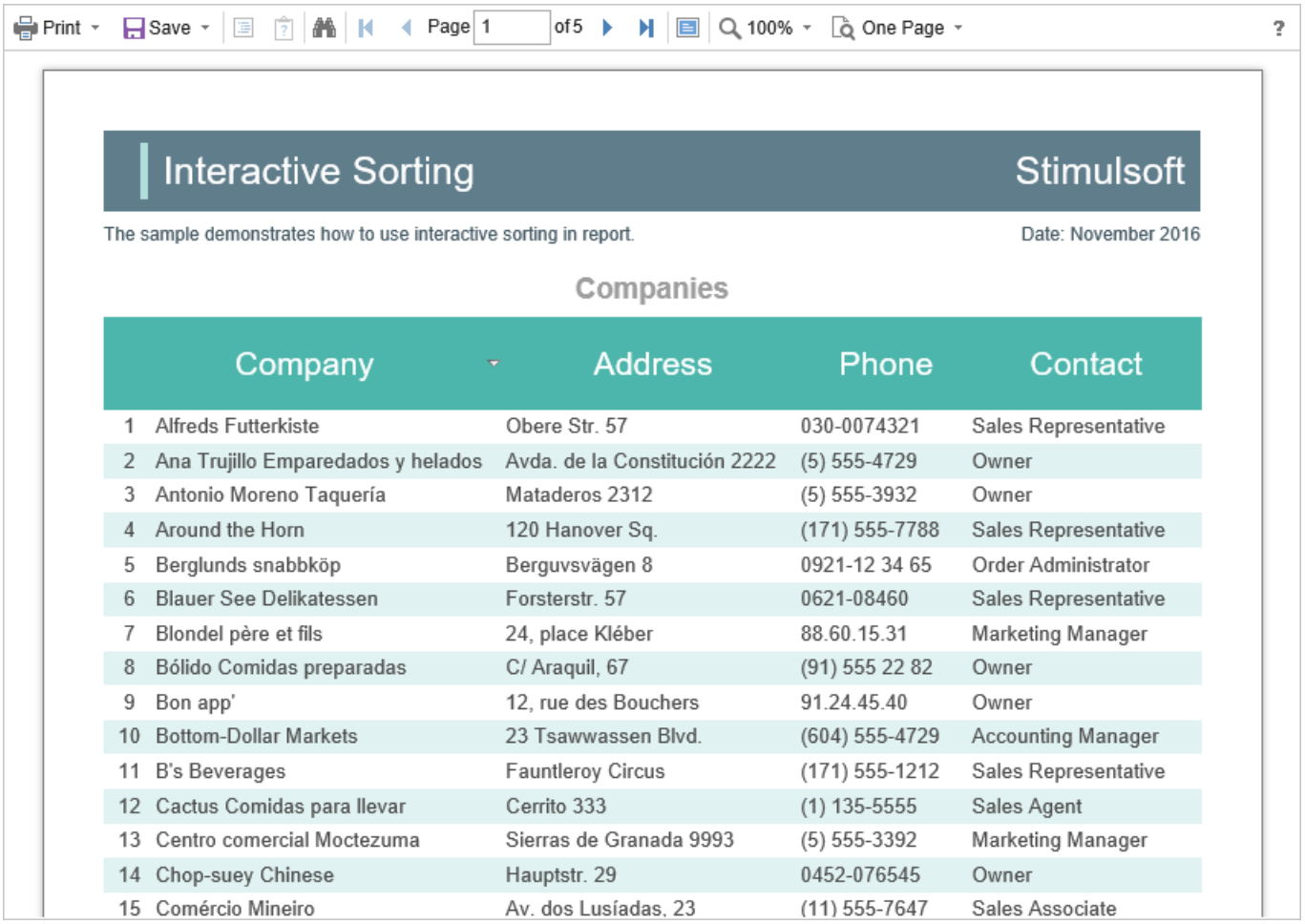
```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.BookmarksPrint = true;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

8.3.12 Dynamic Sorting, Collapsing, and Drill-Down

The **Angular Viewer** component supports dynamic sorting, collapsing, and drill-down of reports. Dynamic sorting provides the ability to change the direction of sorting in a rendered report. To do this, click on the component that has the dynamic sorting enabled. Dynamic sorting is carried out in the following directions - **Ascending** and **Descending**. Each time the component is clicked, the sorting direction is reversed.

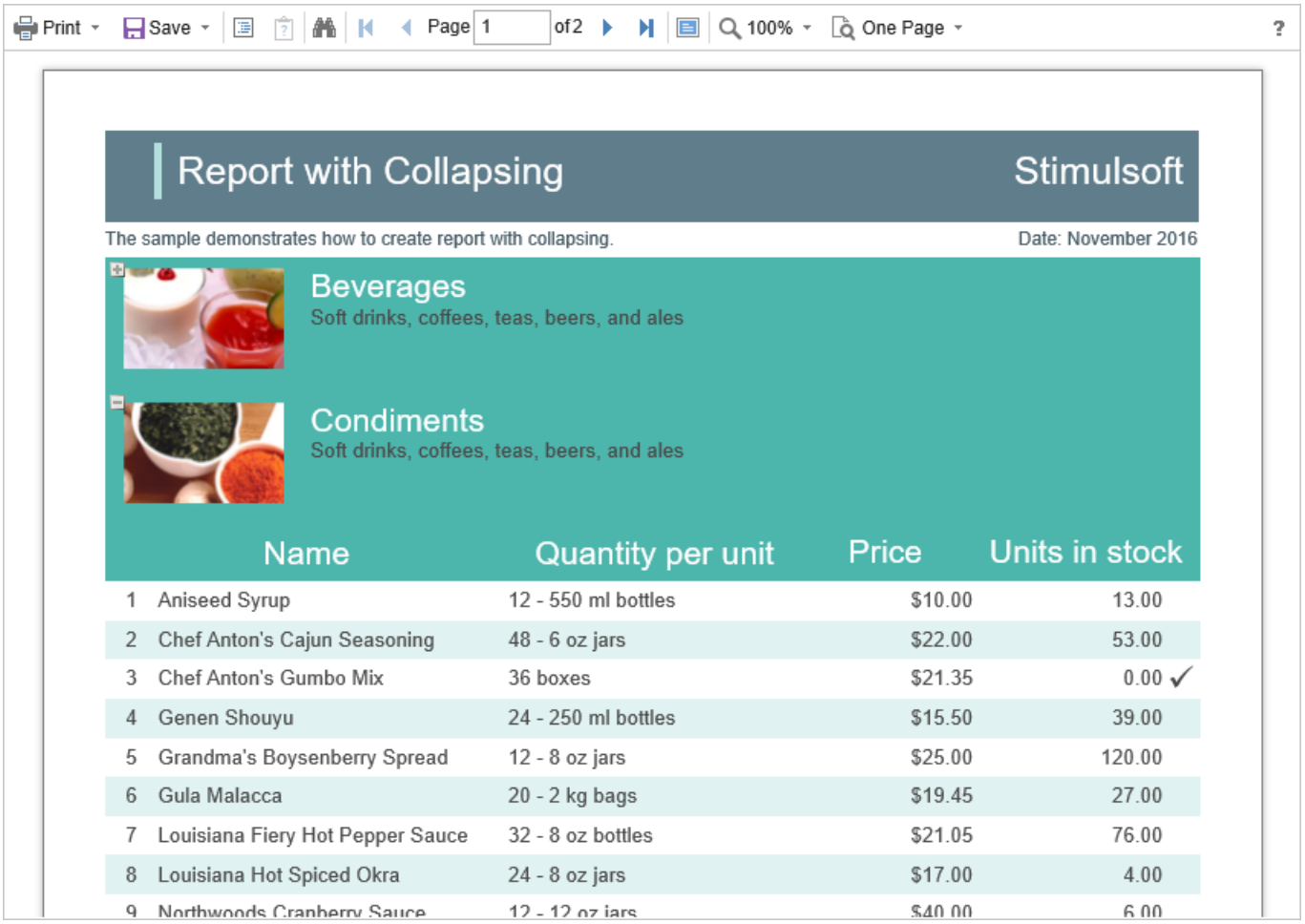
Multi-level sorting is allowed in the report. To do this, hold down the **Ctrl** key and sequentially click on the sorted components in the report. To reset sorting, you can click on any sorted component without holding down the **Ctrl** key.

© 2003-2024 Stimulsoft



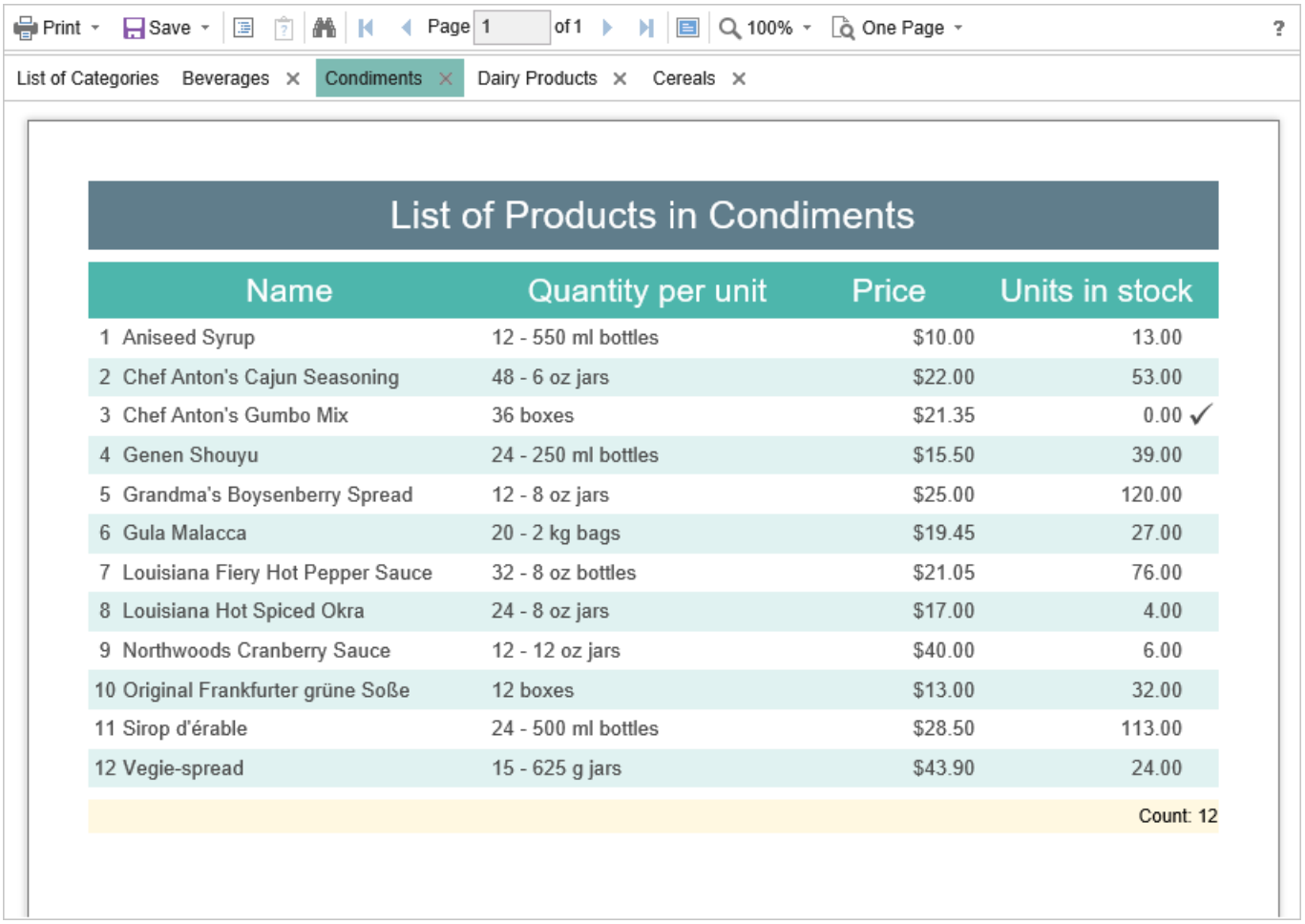
A report with dynamic collapsing is an interactive report in which blocks can collapse/expand their content when you click on the block title. Report elements, which can be collapsed/expanded, are indicated by special icons - **[-]** or **[+]**.

© 2003-2024 Stimulsoft



When using drill-down, under the main panel of the viewer, the drill-down panel with tabs for drill-down reports will be displayed. The currently displayed report will be highlighted.

© 2003-2024 Stimulsoft



To work with dynamic sorting, collapsing and drill-down reports, no additional viewer settings are required. To perform any actions before sorting, collapsing or drill-down of the report, a special **Interaction** action is used. It will be called when interactive action of the viewer.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.Interaction = "ViewerInteraction";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult ViewerInteraction()
```

© 2003-2024 Stimulsoft

```
{
   // Some code before any interaction
   // ...

   return StiAngularViewer.InteractionResult(this);
}
...
```

To get the type of action, you can use the parameters of the viewer. The viewer parameters are represented as an object of the **StiRequestParams** class, they are passed to any server side by any request, and contain all necessary information and states of the client part of the viewer. For each type of interactivity, the viewer has a certain type of action:

- **Sorting** – when using column sorting;
- **DrillDown** – when using drill-down in reports;
- **Collapsing** – when using collapsing report blocks.

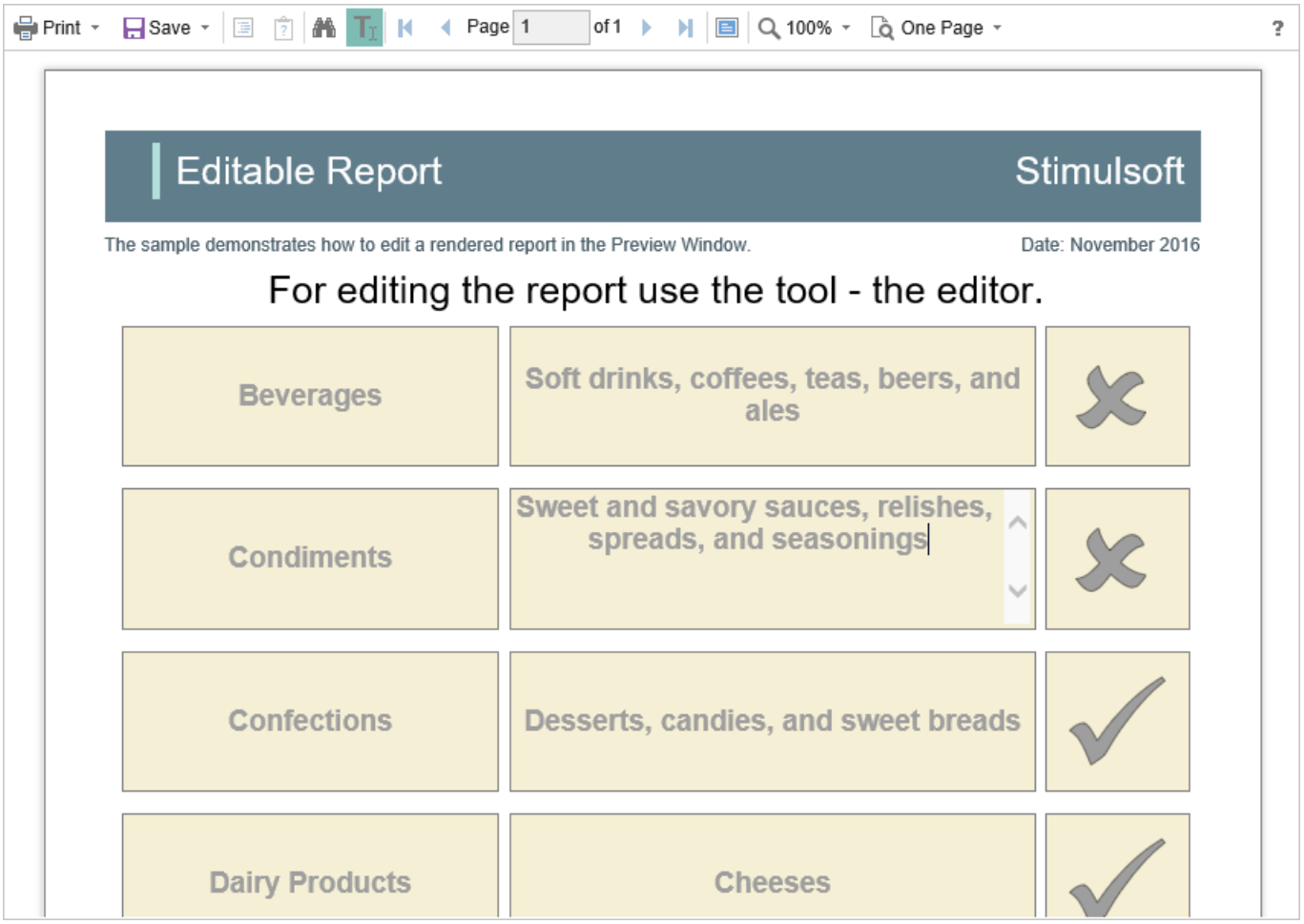
HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiAngularViewer.GetRequestParams(this);
   switch (requestParams.Action)
   {
      case StiAction.Sorting:
        break;

      case StiAction.DrillDown:
        break;

      case StiAction.Collapsing:
        break;
   }

   return StiAngularViewer.InteractionResult(this);
}
...
```

8.3.13 Editing Report

The **Angular Viewer** component has the ability to edit report items, such as text boxes and check boxes. In order the editing be possible, in the report template, you should mark the required components as editable. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be applied to the report.

© 2003-2024 Stimulsoft



For the report edit mode, no special settings of the viewer required.

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

8.3.14 Sending Report by Email

The **Angular Viewer** component provides the ability to send reports by email. To activate this feature, you should set the **ShowSendEmailButton** property of the viewer to **true**, and define the **EmailReport** action.

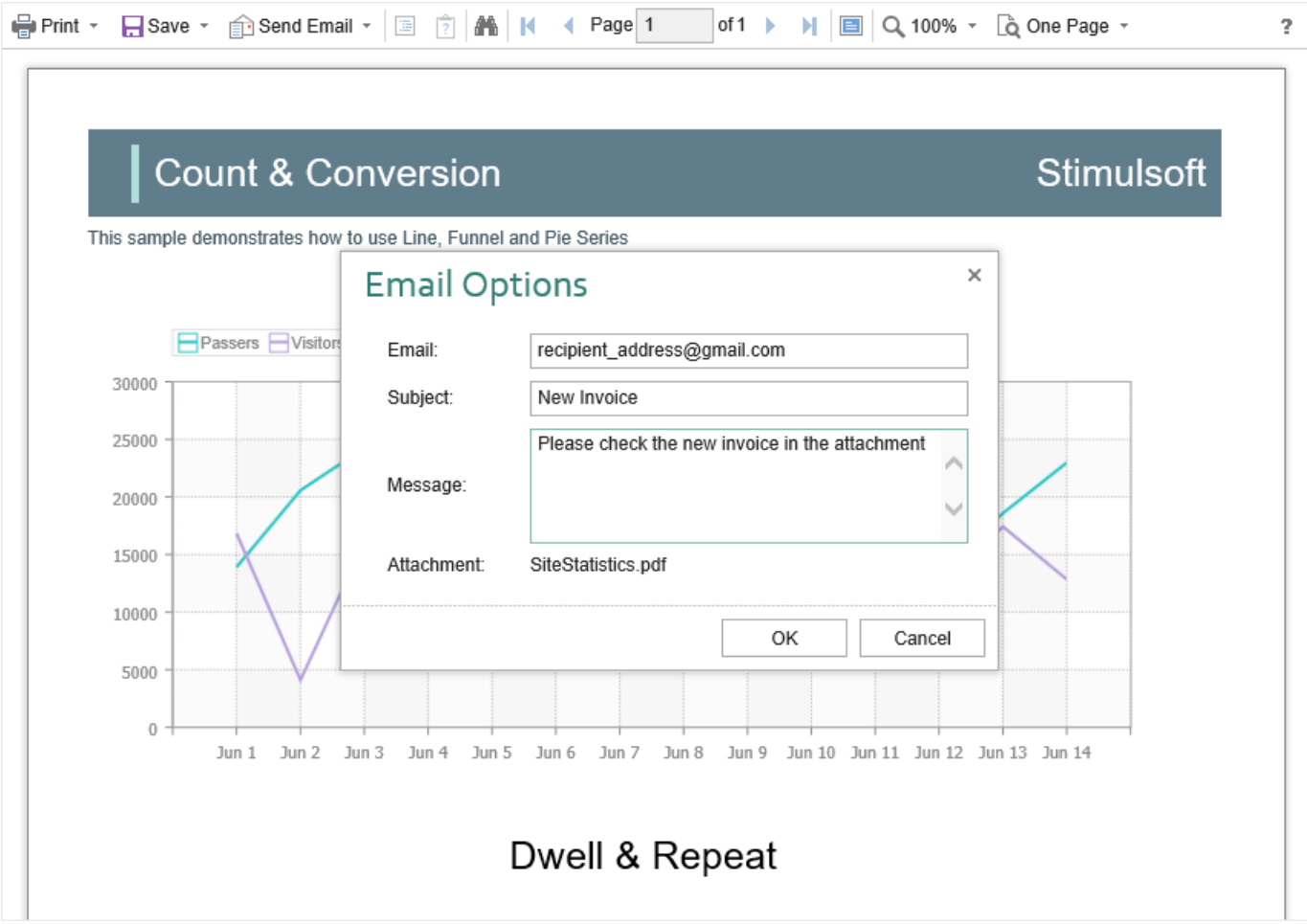
© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.EmailReport = "EmailReport";
   options.Toolbar.ShowSendEmailButton = true;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult EmailReport()
{
   StiEmailOptions options = StiAngularViewer.GetEmailOptions(this);

   // Passed from the viewer, can be checked and changed
   // options.AddressTo = "";
   // options.Subject = "";
   // options.Body = "";

   // Should be filled here
   options.AddressFrom = "admin_address@test.com";
   options.Host = "smtp.test.com";
   options.Port = 465;
   options.UserName = "admin_address@test.com";
   options.Password = "admin_password";

   // options.CC.Add("email@test.com");
   // options.BCC.Add("email@test.com");
   // options.EnableSsl = true;

   return StiAngularViewer.EmailReportResult(this, options);
}
...
```

When sending a report by email, the menu to select the attachment format is displayed. It corresponds to the menu for selecting the format for exporting the report. After selecting the format, the dialog to enter send email parameters, such as the recipient's email, subject and text of the message is displayed.

© 2003-2024 Stimulsoft



After confirmation of sending the email the above described **EmailReport** event will be called. You can check and correct the data entered in this form. The exported report file will be attached to the email automatically.

The **Angular Viewer** component allows you to set default values for the send email form. The **DefaultEmailAddress**, **DefaultEmailSubject** and **DefaultEmailMessage** properties can be used for this. By default, these properties are empty.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Email.DefaultEmailAddress = "recipient_address@gmail.com";
   options.Email.DefaultEmailSubject = "New Invoice";
```

© 2003-2024 Stimulsoft

```
    options.Email.DefaultEmailMessage = "Please check the new invoice in the
    attachment";

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

8.3.15 Calling Designer from Viewer

The **Angular Viewer** component has the ability to call the report designer. The special **Design** button in the toolbar of the viewer (the button is disabled by default) should be used. To use this feature, you should set the **ShowDesignButton** property to **true**, and also to define the **DesignReport** event handler.

HomeController.cs

```
...
public IActionResult InitViewer()
{
    var requestParams = StiAngularViewer.GetRequestParams(this);
    var options = new StiAngularViewerOptions();
    options.Actions.ViewerEvent = "ViewerEvent";
    options.Actions.DesignReport = "DesignReport";
    options.Toolbar.ShowDesignButton = true;

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult DesignReport()
{
    StiReport report = StiAngularViewer.GetReportObject(this);
    ViewBag.ReportName = report.ReportName;

    return View("Designer");
}
...
```

Information

The viewer does not run the designer, it only calls the specified action, in which you can get all the necessary parameters. Then, in the action, you can implement redirection to another View, which contains the report designer.

8.3.16 Caching

The **Angular Viewer** component allows you to use the server cache to store rendered reports. If you do not use caching, then, every time you request a page, you should load the report, connect data, and render it again. If you use caching,

© 2003-2024 Stimulsoft

every time you refresh the page, the previously rendered report will be loaded from the cache.

When using caching, it should be taken into account that every report saved in the cache takes up server memory and, with a large number of requests to reports, this can become a critical issue. Therefore, you need to choose between two options - either low memory requirements but high in performance, or low performance requirements but high in memory.

To use caching, you need to connect modules to work with a session or cache on the server side. To do this, just add the following services to the project in the start file of a project.

Startup.cs

```
...
public void ConfigureServices(IServiceCollection services)
{
   services.AddMemoryCache();
   services.AddSession();
   services.AddMvc();
}
...
```

You can manage caching with the following properties.

The CacheMode property

This property of the viewer enabled caching and sets its type. It can take one of the following values, specified in the **StiServerCacheMode** enumeration:

- **None** – caching is disabled, each action of the viewer requires loading the report from the file and, if it is a report template, then render it;
- **ObjectCache** – for caching, the server cache is used. The report object is saved in this cach (set by default);
- **StringCache** – for caching, the server cache is used. The report is saved as a packed string in this cache;
- **ObjectSession** – the current session, in which the report object is saved, is used for caching;
- **StringSession** – for caching, the current session is used. The report is saved as a packed string in this cache.

© 2003-2024 Stimulsoft

The CacheItemPriority property

This property sets the priority of the report stored in the server's cache. It affects the automatic clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.

The CacheTimeout property

This property specifies the amount of time in minutes for which you want to save the report in the server cache. If you use caching and the requested report is not found in the cache (the objects storage time has expired), then it will be requested again using a special **GetReport** event, then connect the report data and render it.

StiCacheHelper

The **Angular Viewer** component provides the ability to define your own methods of working with report caching. For this purpose, a special class **StiCacheHelper** is used. It contains methods for obtaining a report from the cache and saving the report to the cache. It is necessary to create a new class inherited from **StiCacheHelper** and reload the above methods, which respectively have the names - **GetReport** and **SaveReport**.

HomeController.cs

```
...
public class ViewerController : Controller
{
   public class StiMyCacheHelper : StiCacheHelper
   {
      public override StiReport GetReport(string guid)
      {
         string path =
         System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
         ), guid);
         if (System.IO.File.Exists(path))
         {
            StiReport report = new StiReport();
            string packedReport = System.IO.File.ReadAllText(path);
            if (guid.EndsWith("template"))
            report.LoadPackedReportFromString(packedReport);
```

© 2003-2024 Stimulsoft

```
            else report.LoadPackedDocumentFromString(packedReport);

            return report;
        }
        return null;

        //return base.GetReport(guid);
    }

    public override void SaveReport(StiReport report, string guid)
    {
        string packedReport = guid.EndsWith("template") ?
        report.SavePackedReportToString() :
        report.SavePackedDocumentToString();
        string path =
        System.IO.Path.Combine(this.HttpContext.Server.MapPath("CacheFiles"
        ), guid);
        System.IO.File.WriteAllText(path, packedReport);

        //base.SaveReport(report, guid);
    }
  }

  static ViewerController()
  {
    StiAngularViewer.CacheHelper = new StiMyCacheHelper();
  }
}
...
```

To initialize the work with report caching using the created class, it is enough to set it as a value of the static **StiAngularViewer.CacheHelper** property in the controller constructor.

Information

If report caching is disabled (the **CacheMode** property of the viewer is set to **None**), the specified class will not be used.

8.3.17 Additional Methods

For **Angular Viewer**, there are several additional methods that are used to get the object of the currently viewed report, parameters of the current state of the viewer and other useful data. These methods can be used in any actions of the viewer.

The GetReportObject() method

© 2003-2024 Stimulsoft

Returns the report object with which the viewer is currently working. It is possible to perform the necessary actions with it - register new data sets, change report properties, assign parameters or load another report to the object. Then, the report can be returned to the viewer, specifying it as a parameter in the resulting action method.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiReport report = StiAngularViewer.GetReportObject(this);
   report.ReportName = "MyReportName";

   return StiAngularViewer.InteractionResult(this, report);
}
...
```

The GetRouteValues() method

Returns values for URLs with which the viewer page was opened. Thus, it is possible to get the initial collection of run page parameters in any viewer action and use these values for any checks and conditions.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   RouteValueDictionary routeValues =
   StiAngularViewer.GetRouteValues(this);

   return StiAngularViewer.InteractionResult(this);
}
...
```

You can also get values of URL parameters by parameter name, specifying it as the parameter of the called action of the viewer.

HomeController.cs

```
...
public IActionResult ViewerInteraction(string id)
{
   return StiAngularViewer.InteractionResult(this);
}
```

© 2003-2024 Stimulsoft

```
...
```

The GetFormValues() method

Returns the values of the form that initiated (opened by the POST request) a page of the viewer. Thus, it is possible to get a collection of form parameters in any action of the viewer.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   NameValueCollection formValues = StiAngularViewer.GetFormValues(this);

   return StiAngularViewer.InteractionResult(this);
}
...
```

By default, this feature is disabled in order to optimize requests of the client side of the viewer to the server. To enable it, set the **PassFormValues** property to **true**.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Server.PassFormValues = true;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

The GetRequestParams() method

Returns all parameters of the current state of the viewer passed to the server side. They can be useful for determining the type of action that the viewer is currently executing - for example, to determine the type of export, as well as all action parameters.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiRequestParams requestParams =
   StiAngularViewer.GetRequestParams(this);
   if (requestParams.ExportFormat == StiExportFormat.Pdf)
   {
      StiReport report = StiAngularViewer.GetReportObject(this);

      // Some action with report for the PDF export
      // ...

      return StiAngularViewer.ExportReportResult(this, report);
   }

   return StiAngularViewer.ExportReportResult(this);
}
...
```

You can change the values of some parameters. After making changes, for correct operation of the viewer, you should transfer the changed parameter object to the input of thse resulting method.

HomeController.cs

```
...
public IActionResult ViewerInteraction()
{
   StiRequestParams requestParams =
   StiAngularViewer.GetRequestParams(this);
   if (requestParams.Action == StiAction.Variables)
   {
      requestParams.Interaction.Variables["Variable1"] = "MyValue";
      return StiAngularViewer.InteractionResult(this, requestParams);
   }

   return StiAngularViewer.InteractionResult(this);
}
...
```

The GetExportSettings() method

Returns all the parameters of the current report export. The type of the parameter object will correspond to the type of export selected in the viewer menu. Any export parameters can be changed and passed to the input of the resulting method. In this case, the report will be exported with the parameters transferred.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult ExportReport()
{
   StiExportSettings settings = StiAngularViewer.GetExportSettings(this);
   if (settings.GetExportFormat() == StiExportFormat.Pdf)
   {
      StiPdfExportSettings pdfSettings = (StiPdfExportSettings)settings;
      pdfSettings.EmbeddedFonts = true;
      pdfSettings.AllowEditable = StiPdfAllowEditable.No;
      return StiAngularViewer.ExportReportResult(this, settings);
   }

   return StiAngularViewer.ExportReportResult(this);
}
...
```

The MapPath() and MapWebRootPath() methods

Returns the absolute path, respectively, to the application or wwwroot directory. You can use this to upload report templates files, data files, etc. These methods are located in the StiAngularHelper static class.

HomeController.cs

```
...
public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(StiAngularHelper.MapPath(this, "Reports/SimpleList.mrt"));

   return StiAngularViewer.GetReportResult(this, report);
}
...
```

8.3.18 Timeout

When working with the **StiNetCoreViewer** component, you can set the timeout for various operations — storing the report in the cache, server response, and query execution. The timeout setting is done using the component properties and report options.

CacheTimeout Property

Sets the time in minutes that the server will store the rendered report since the last action of the viewer. The default setting is 10 minutes.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Server.CacheTimeout = 10;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

Using cache will increase the speed of the report viewer. See the chapter Caching for more information

RequestTimeout Property

Sets the time to wait for a response from the server in seconds, after which an error will be generated. The default value is 30 seconds. For big reports, it is recommended to increase this value.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Server.RequestTimeout = 30;

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

CommandTimeout Option

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to set the query timeout for the already created connection, and data sources in the report.

© 2003-2024 Stimulsoft

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Actions.GetReport = "GetReport";

   return StiAngularViewer.ViewerDataResult(requestParams, options);
}

public IActionResult GetReport()
{
   StiReport report = new StiReport();
   report.Load(Server.MapPath("Report.mrt"));
   ((StiSqlSource)
   report.Dictionary.DataSources["DataSourceName"]).CommandTimeout = 1000;

   return StiNetCoreViewer.GetReportResult(this, report);
}

public IActionResult ViewerEvent()
{
   return StiNetCoreViewer.ViewerEventResult(this);
}
...
```

8.3.19 Viewer Settings

The **Angular Viewer** is configured using properties that are located in the **StiNetCoreViewerOptions** class. All properties are divided into groups. Some of the groups contain subgroups for ease of use. The following is an example of setting the properties of the viewer.

HomeController.cs

```
...
public IActionResult InitViewer()
{
   var requestParams = StiAngularViewer.GetRequestParams(this);
   var options = new StiAngularViewerOptions();

   options.Theme = StiViewerTheme.Office2022WhiteTeal;
   options.Localization = "Localization/en.xml";
   options.Actions.GetReport = "GetReport";
   options.Actions.ViewerEvent = "ViewerEvent";
   options.Appearance.InterfaceType = StiInterfaceType.Auto;
   options.Appearance.ScrollbarsMode = true;
   options.Appearance.ShowTooltips = false;
   options.Exports.DefaultSettings.ExportToPdf.CreatorString = "Company
   Name";
```

© 2003-2024 Stimulsoft

```
    options.Exports.DefaultSettings.ExportToPdf.ImageQuality = 0.75f;
    options.Exports.ShowExportToDbf = false;
    options.Exports.ShowExportToDif = false;

    return StiAngularViewer.ViewerDataResult(requestParams, options);
}
...
```

Main settings (without groups)

Name	Description
Theme	Sets the viewer theme. The list of available themes can be found in the **StiViewerTheme** enumeration. The default value is **Office2022WhiteBlue**.
Localization	Sets the path to the XML localization file. The path can be absolute or relative. By default, English localization is used. It is built into the viewer and does not require additional XML files.
Width	Sets the width of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. The default width is 100%.
Height	Sets the height of the component in the required units that are specified in the **Unit** class. The value can be set in pixels - **Unit.Pixel()**, points - **Unit.Point()** and per cent - **Unit.Percentage()**. By default, the automatic height is set depending on the size of the report page, or 650 pixels in the view mode of the viewer with scrollbars.

Actions

Name	Description

© 2003-2024 Stimulsoft

GetReport	Specifies the name of the action method for preparing the rendered report. Specifies the name of the action method for preparing the constructed report. If report caching is enabled, this action will be called only once when the report is requested or if the requested report is not found in the server cache.
PrintReport	Specifies the name of the action method of report printing.
ExportReport	Specifies the name of the action method of the export the report to the specified format.
EmailReport	Specifies the name of the action method of sending the report by email.
Interaction	Specifies the name of the action method for the viewer to work with interactive operations, such as using parameters, dynamic sorting, collapsing and drill-down.
DesignReport	Specifies the name of the action method to go to the specified view by clicking the Design button on the viewer panel.
ViewerEvent	Specifies the name of the action method of basic viewer events and the processing actions of the viewer, such as printing and exporting a report, working with parameters, and interactivity, if these actions are not specified separately. In addition, this action is used to load scripts and styles of the viewer. This action is mandatory.

Server

Name	Description
Controller	Specifies the name of the report controller for the report viewer. If this property is not specified, then the current controller will be

© 2003-2024 Stimulsoft

	used to process requests.
RouteTemplate	Sets the route template that is returned when the report viewer actions are executed. If the property is not set, then the MVC project template will be used instead. The default value of the property is null.
RequestTimeout	Sets the response timeout from the server in seconds, after which an error will be generated. The default value is 20 seconds. For big reports, it is recommended to increase this value.
CacheTimeout	Sets the time in minutes that the server will store the report since the last action of the viewer. The default value is 20 minutes.
CacheMode	Sets the report caching mode. It can take one of the following values of the **StiServerCacheMode** enumeration: › **None** – caching is disabled, the report will be reloaded each time using the **GetReport** event; › **ObjectCache** – the cache is used as the storage, the report is stored as an object (default value); › **ObjectSession** – the session is used as the storage, the report is stored as an object; › **StringCache** – the server cache is used as the storage, the report is serialized to a packed string; › **StringSession** – the session is used as a storage, the report is serialized into a packed string.
CacheItemPriority	Sets the priority of the report stored in the server cache. This property affects the automatic

© 2003-2024 Stimulsoft

	clearing of the server memory in case of lack of memory. The lower the priority is, the greater is the chance of removing information from memory.
AllowAutoUpdateCache	Sets the mode for automatic cache update. The report stored in the cache or the server session will be automatically re-saved after a certain period of time when the viewer is idle (every 3 minutes). By default, the property is set to **true**.
UseRelativeUrls	Sets the viewer mode in which relative URLs are used for AJAX requests to the server. By default, the property is set to **true**.
PortNumber	Gets or sets a value which specifies the port number to use in the URL. A value of **0** defines automatic detection (default value). A value of **-1** removes the port number.
PassQueryParametersForResources	Enables transferring all request URL parameters when generating links to the resources of the viewer. If **false**, only the necessary parameters are used to request the resources of the viewer. This corresponds to the more correct work of the browser cache. By default, the property is set to **true**.
PassQueryParametersToReport	Enables using all the URL parameters of the request as the variable values. The variables names must match the parameters. The default value of the property is false.
PassFormValues	Enables passing the values of the POST form to the client side, if these values are required to be used in the actions of the viewer. If you enable this property, the additional **GetFormValues()** method will return a collection of form parameters. By default, the property is **false**.
ShowServerErrorPage	Enables displaying an HTML page with the details of the error that occurred on the server side. When the property is enabled, the details of the error will be displayed in the viewer

© 2003-2024 Stimulsoft

	window. If the property is disabled, only the numeric error code and a short error text in the dialog box will be displayed. By default, the property is set to **true**.
UseCompression	Enables compression of the viewer requests into the GZip stream. That allows to decrease the amount of the internet-traffic but slows down slightly the viewer. The default value of the property is false.
UseCacheForResources	Enables caching of the component resources on the server side. The following resources are supported: scripts, styles and images. This option improves the load speed of the component and also reduces the server load in multi-client environments. The default value is **true**.
UseLocalizedCache	Sets a value which enables the use of a different cache depending on the selected localization. The default value of the property is **false**.
AllowLoadingCustomFontsToClientSide	Allows you to pass custom fonts to the client side and convert them to CSS style for the correct display of text as HTML with a specified font. By default, the property is set to **false**.

Appearance

Name	Description
CustomCss	Sets the path to the CSS file of the viewer's styles. The standard styles of the chosen theme will not be loaded if this property has got a value. The default value of the property is an empty string.
BackgroundColor	Sets the background color of the viewer. By default it is set to **White**.
PageBorderColor	Sets the border color of the viewer. By default it

© 2003-2024 Stimulsoft

	is set to **Gray**.
RightToLeft	Sets the **Right to Left** mode for viewer controls. By default the property is set to **false**. By default, the property is set to **false**.
FullScreenMode	Sets the full-screen display mode of the viewer. By default, the property is set to **false**.
ScrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to **false**.
OpenLinksWindow	Sets the target window for opening links contained in the report. By default, the property is set to **Blank** (new window).
OpenExportedReportWindow	Sets the target window for opening the export file from the viewer. By default, the property is set to **Blank** (new window).
DesignWindow	Sets the destination window for launching the report designer. The default value of the property is **Self** (whish is the current window).
ShowTooltips	Enables showing tips for the viewer controls when the mouse hovers over. By default, the property is set to **true**.
ShowTooltipsHelp	Enables showing links to online documentation for the viewer controls. By default, the property is set to **true**.
ShowDialogsHelp	Sets a value which indicates that show or hide the help button in dialogs. By default, the property is set to **true**.
PageAlignment	Sets the position of the report page in the viewer window. It can take one of the following values of the **StiContentAlignment** enumeration: › **Left** – the page will be aligned left; › **Center** – the page will be centered (default value); › **Right** – the page will be aligned right.

© 2003-2024 Stimulsoft

ShowPageShadow	Enables displaying shadow for report pages. By default the property is set to **true**.
BookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default the property is set to **false**.
BookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
ParametersPanelPosition	Specifies the position of the report parameters panel. It can take one of the following **StiParametersPanelPosition** enumeration values: › **Top** - the panel will be docked to the top margin (default value); › **Left** - the panel will be docked to the left margin.
ParametersPanelMaxHeight	Sets the maximum height of the parameters bar in pixels. By default, the maximum height is 300 pixels.
ParametersPanelColumnsCount	Sets the number of columns to display report parameters. By default, there are 2 columns.
ParametersPanelSortDataItems	Gets or sets a value which indicates that variable items will be sorted. By default the property is set to **true**.
ParametersPanelDateFormat	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the date and time format set by the browser is used.
InterfaceType	Sets the type of interface used for the viewer. It can take one of the following **StiInterfaceType** enumeration values: › **Auto** – the viewer's interface is determined automatically depending of the device that is report is displayed on. That is the default value. › **Mouse** – the standard interface with a mouse

© 2003-2024 Stimulsoft

	control will be used for all the screen types. › **Touch** – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability. › **Mobile** - the Mobile interface will be used to control the viewer for all the screen types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use with the smartphones.
AllowMobileMode	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to **false**, then the mobile view will not be used. If the option is set to **true**, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option is set to **true**.
ChartRenderType	Sets the displaying mode of charts on the report page. It can take one of the following **StiChartRenderType** enumeration values: › **Image** – charts are displayed as static images; › **Vector** – charts are displayed in the vector mode as an SVG object; › **AnimatedVector** - charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (default value).
ReportDisplayMode	Sets the export mode for displaying report pages. It can take one of the following values of the **StiReportDisplayMode** enumeration: › **FromReport** - the export mode of the report elements is defined from report template settings - Div or Table;

© 2003-2024 Stimulsoft

	› **Table** – report elements are exported using HTML tables (default value); › **Div** – report elements are exported using DIV markup; › **Span** - report items are exported using SPAN markup.
DatePickerFirstDayOfWeek	Sets the first day of the week for the date picker. It can take one of the following values of the **StiFirstDayOfWeek** enumeration: › **Monday** – the first day of the week is Monday (default value); › **Sunday** – the first day of the week is Sunday.
DatePickerIncludeCurrentDayForRanges	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default the property is set to **false**.
AllowTouchZoom	Sets ability to change the scale of the report page by using the two-fingers gesture (Pinch to Zoom) for the touch-screens. The default value of the property is **true**.
ShowReportIsNotSpecifiedMessage	Sets a value which indicates that 'The report is not specified' message will be shown. The default value of the property is **true**.
PrintToPdfMode	Sets the Print to PDF mode. It has the following values: › **StiPrintToPdfMode.Hidden** - hidden print mode (default value); › **StiPrintToPdfMode.Popup** - the PDF document will be displayed before printing in a pop-up window.
ImagesQuality	Gets or sets the image quality that will be used on the viewer page. It has the following values: › **StiImagesQuality.Low** - low quality, used to speed up loading reports and saves memory; › **StiImagesQuality.Normal** - normal quality, suitable for most cases (default value); › **StiImagesQuality.High** - high quality, used

© 2003-2024 Stimulsoft

	for ultra high-definition displays, but may slow down the loading of pages.
CombineReportPages	Sets a value which indicates that if a report contains several pages, then they will be combined in preview. By default the property is set to **false**.

Toolbar

Name	Description
Visible	Enables displaying the viewer toolbar. By default, the property is set to **true**.
DisplayMode	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the **StiToolbarDisplayMode** enumeration: › **Simple** - all controls are located on the same control panel (default value); › **Separated** - the control panel is split into top and bottom panels.
BackgroundColor	Specifies the background color of the viewer toolbar. The default color of the selected theme is used.
BorderColor	Specifies the border color of the viewer toolbar. The default color of the selected theme is used.
FontColor	Specifies the text color for the toolbar and the viewer menu. The default color of the selected theme is used.
FontFamily	Specifies the font for the toolbar and the viewer menu. The default font of the selected theme is used.
Alignment	Sets the alignment mode for the controls on the viewer toolbar. It can take one of the following values of the **StiContentAlignment** enumeration:

© 2003-2024 Stimulsoft

	› **Left** – elements will be aligned left; › **Center** – elements will be centered; › **Right** – elements will be aligned right; › **Default** – the alignment depends on the RightToLeft property (default value).
ShowButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default the property is set to **true**.
ShowPrintButton	Enables showing the button - **Print** - on the viewer toolbar. By default, the property is set to **true**.
ShowOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **false**.
ShowSaveButton	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to true.
ShowSendEmailButton	Enables showing the button - **Send Email** - on the viewer toolbar. By default, the property is set to **false**. Also, you should add the EmailReport action.
ShowFindButton	Enables showing the button - **Find** - on the viewer toolbar. By default, the property is set to **true**.
ShowBookmarksButton	Enables showing the button - **Bookmarks** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
ShowParametersButton	Enables showing the button - **Parameters** - on the viewer toolbar. By default, the property is set to **true**. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.

© 2003-2024 Stimulsoft

ShowResourcesButton	Enables showing the button - **Resources** - on the toolbar of the viewer. By default, the property is set to **true**. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.
ShowEditorButton	Enables showing the button - **Editor** - on the viewer toolbar. By default, the property is set to **true**.
ShowFullScreenButton	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports or dashboards. . By default, the property is set to **true**.
ShowFirstPageButton	Enables showing the button - **First Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowPreviousPageButton	Enables showing the button - **Previous Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowCurrentPageControl	Enables showing the current report page indicator. By default, the property is set to **true**.
ShowNextPageButton	Enables showing the button - **Next Page** - on the viewer toolbar. By default, the property is set to **true**.
ShowLastPageButton	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to **true**.
ShowZoomButton	Enables showing the button to select the report zoom. By default, the property is set to **true**.
ShowViewModeButton	Enables showing the button to select the view mode of the report page. By default, the property is set to **true**.
ShowDesignButton	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to **false**.

© 2003-2024 Stimulsoft

ShowAboutButton	Enables showing the button - **About** - on the viewer toolbar. By default, the property is set to **true**.
ShowRefreshButton	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to **true**.
ShowPinToolbarButton	Enables displaying of the **Pin Toolbar** button on the viewer's toolbar. The button is available only in the Mobile mode of the viewer's interface. The default value of the property is **true**.
PrintDestination	Sets the report printing mode. It can take one of the following values of the **StiPrintDestination** enumeration: › **Default** – a menu with a choice of printing modes will be displayed (default value); › **Pdf** – printing will be done in the PDF format; › **Direct** – printing will be done to the HTML format directly to the printer, the system print dialog will be displayed; › **PopupWindow** – printing will be done in the HTML format via the preview window of the report.
ViewMode	Sets the mode for displaying report pages. It can take one of the following **StiWebViewMode** enumeration values: › **SinglePage** - displays one page of the report selected in the toolbar of the viewer (default value); › **Continuous** - displays all pages of the report; › **MultiplePages** - displays all report pages as a table.
Zoom	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of

© 2003-2024 Stimulsoft

	the following values: › **StiZoomMode.PageWidth** – when the viewer runs, the zoom, necessary to display the report by the page width, will be set; › **StiZoomMode.PageHeight** – when the viewer runs, the zoom, necessary to display the report by the page height, will be set.
MenuAnimation	Enables animation when the viewer menu shows/hides. By default the property is set to **true**.
ShowMenuMode	Sets the display mode of the viewer menu. It can take one of the following values of the **StiShowMenuMode** enumeration: › **Click** – shows menu by mouse click (default value); › **Hover** – shows menu by hovering the mouse cursor.
AutoHide	Enables auto-hiding of the viewer's toolbar. The property will work only for the Mobile mode of the viewer's interface. The default value of the property is **true**.

Export

Name	Description
DefaultSettings	This group of properties provides the ability to specify the default export settings for each export type. These settings will be applied to the export dialogs when the viewer runs or to the report, if export dialogs are disabled.
StoreExportSettings	Enables saving selected settings in the export dialogs. Settings will be stored in browser cookies. By default the property is set to **true**.
ShowExportDialog	Enables showing the export options dialog box.

© 2003-2024 Stimulsoft

	If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
ShowExportToDocument	Enables the export menu item - **Document File**. By default, the property is set to **true**.
ShowExportToPdf	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToXps	Enables the export menu item - **Microsoft XPS File**. By default, the property is set to **false**.
ShowExportToPowerPoint	Enables the export menu item - **Microsoft PowerPoint 2007/2010 File**. By default, the property is set to **true**.
ShowExportToHtml	Enables the export menu item - **HTML File**. By default, the property is set to **true**.
ShowExportToHtml5	Enables the export menu item - **HTML5 File**. By default, the property is set to **true**.
ShowExportToMht	Enables the export menu item - **MHT Web Archive**. By default, the property is set to **true**.
ShowExportToText	Enables the export menu item - **Text File**. By default, the property is set to **true**.
ShowExportToRtf	Enables the export menu item - **Rich Text File**. By default, the property is set to **true**.
ShowExportToWord2007	Enables the export menu item - **Microsoft Word 2007/2010 File**. By default, the property is set to **true**.
ShowExportToOpenDocumentWriter	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to **true**.
ShowExportToExcel	Enables the export menu item - **Microsoft Excel File**. By default, the property is set to **true**.
ShowExportToExcelXml	Enables the export menu item - **Microsoft Excel Xml File**. By default, the property is set to

© 2003-2024 Stimulsoft

	true.
ShowExportToExcel2007	Enables displaying the **Microsoft Excel 2007/2010 File** export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToOpenDocumentCalc	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to **true**.
ShowExportToCsv	Enables the export menu item - **CSV File**. By default, the property is set to **true**.
ShowExportToDbf	Enables the export menu item - **DBF File**. By default, the property is set to **true**.
ShowExportToXml	Enables the export menu item - **XML File**. By default, the property is set to **true**.
ShowExportToDif	Enables the export menu item - **Data Interchange Format (DIF) File**. By default, the property is set to **true**.
ShowExportToSylk	Enables the export menu item - **Symbolic Link (SYLK) File**. By default, the property is set to **true**.
ShowExportToJson	Enables the export menu item - **JSON File**. By default, the property is set to **true**.
ShowExportToImageBmp	Enables displaying the **BMP Image** export menu item when viewing reports, and the **BMP Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageGif	Enables displaying the **GIF Image** export menu item when viewing reports, and the **GIF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageJpeg	Enables displaying the **JPEG Image** export menu item when viewing reports, and the **JPEG Image** item when viewing dashboards. By default, the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportToImagePcx	Enables displaying the **PCX Image** export menu item when viewing reports, and the **PCX Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImagePng	Enables displaying the **PNG Image** export menu item when viewing reports, and the **PNG Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageTiff	Enables displaying the **TIFF Image** export menu item when viewing reports, and the **TIFF Image** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvg	Enables displaying the **Scalable Vector Graphics (SVG) File** export menu item when viewing reports, and the **Scalable Vector Graphics (SVG) File** item when viewing dashboards. By default, the property is set to **true**.
ShowExportToImageSvgz	Enables displaying the **Compressed SVG (SVGZ) File** export menu item when viewing reports, and the **Compressed SVG (SVGZ) File** item when viewing dashboards. By default, the property is set to **true**.
ShowOpenAfterExport	Enables displaying the **Open After Export** parameter in export settings menu. By default the property is set to **true**.

Email

Name	Description
ShowEmailDialog	Enables displaying settings for sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server side in the **EmailReport** action. By default the property is set to **true**.

© 2003-2024 Stimulsoft

ShowExportDialog	Enables displaying export options dialog box when sending email. If the property is set to **false**, the export will be done with the default settings. By default the property is set to **true**.
DefaultEmailAddress	Sets the default recipient email, i.e. the address to which the email with the attached report will be sent.
DefaultEmailSubject	Sets the default email subject (header).
DefaultEmailMessage	Sets the default email message (text).

8.3.20 API References

You can using Angular Viewer API. **StiAngularViewer** contains api object that allow to manipulate viewer or view states.

app.component.ts

```
...
export class AppComponent {
   @ViewChild('viewer') viewer: StimulsoftViewerComponent;
   ...
}
...
```

Enhance app.component.html, add reference to component, add display current page & zoom, add buttons that allow to zoom page to 50% & export report to PDF format with setting ImageQuality to 200:

app.component.html

```
...
Zoom is {{ viewer.api.zoom }}<br />
Current page {{ viewer.api.currentPage + 1 }}<br />
<input type="button" (click)="viewer.api.zoom = 50" value="Zomm to 50%" />
<input
   type="button"
   (click)="viewer.api.export('Pdf', { ImageResolution: 200 })"
   value="Export to PDF"
/>

<stimulsoft-viewer-angular
   #viewer
   [requestUrl]="'http://localhost:60801/Viewer/{action}'"
   [action]="'InitViewer'"
   [height]="'600px'"
></stimulsoft-viewer-angular>
```

© 2003-2024 Stimulsoft

```
...
```

Options
Angular **stimulsoft-viewer-angular** contains options.

Parameter	Description
requestUrl	Url to server instance, must contains placeholder {action} that will replace with action. Example: http://server.url:51528/Viewer/{action}
action	Controller action that handle viewer initial request.
properties	Properties that will transfer to controller action as JSON object.
width	Viewer width.
height	Viewer height.
backgroundColor	Viewer background color.
style	Style of viewer applied to main span as [style] ="style".

Events
Angular **stimulsoft-viewer-angular** contains events.

Parameter	Description
loaded	Occurs when report loaded.
error	Occurs on error, $event is ErrorMessage object contains error: string & type: any (if present).
export	Occurs on export, $event object contains exportFormat: string & exportSettings: {}.
email	Occurs on export & email, $event object contains exportFormat: string & exportSettings: {}.
print	Occurs on export & email, $event object

© 2003-2024 Stimulsoft

	contains format: string : 'PrintPdf' or 'PrintWithoutPreview' or 'PrintWithPreview'.

Methods

With API property of **stimulsoft-viewer-angular** (**StiAngularViewer**) you can perform different actions.

Parameter	Description
currentPage	Get or set the current page number.
pageCount	The total pages count.
viewMode	Get or set the view mode, can be 'SinglePage', 'Continuous' & 'MultiplePages'.
zoom	Get or set the page zoom in percent. From 1 to 1000.
zoomPageHeight()	Zoom page in height.
zoomPageWidth()	Zoom page in width.
printPdf()	Print current report to PDF.
printWithoutPreview()	Print current report without preview.
printWithPreview()	Print current report/dashboard with preview.
showExportForm(format: string)	Show export form. **format** The format to export, can be 'Document', 'Pdf', 'Xps', 'Ppt2007', 'Html', 'Html5', 'Mht', 'Text', 'Rtf', 'Word2007', 'Odt', 'Excel', 'ExcelBinary', 'ExcelXml', 'Excel2007', 'Ods', 'Csv', 'Dbf', 'Dif', 'Sylk', 'Json', 'Xml', 'ImageBmp', 'ImageGif', 'ImageJpeg', 'ImagePcx', 'ImagePng', 'ImageTiff', 'ImageEmf', 'ImageSvg', 'ImageSvgz'
showExportEmailForm(format: string)	Show export form & email. **format** The format to export, can be 'Document', 'Pdf', 'Xps', 'Ppt2007', 'Html', 'Html5', 'Mht', 'Text',

© 2003-2024 Stimulsoft

	'Rtf', 'Word2007', 'Odt', 'Excel', 'ExcelBinary', 'ExcelXml', 'Excel2007', 'Ods', 'Csv', 'Dbf', 'Dif', 'Sylk', 'Json', 'Xml', 'ImageBmp', 'ImageGif', 'ImageJpeg', 'ImagePcx', 'ImagePng', 'ImageTiff', 'ImageEmf', 'ImageSvg', 'ImageSvgz'
export(format: string, settings?: any)	Export report to selected format. Use default settings if not specified. **format** The format to export, can be 'Document', 'Pdf', 'Xps', 'Ppt2007', 'Html', 'Html5', 'Mht', 'Text', 'Rtf', 'Word2007', 'Odt', 'Excel', 'ExcelBinary', 'ExcelXml', 'Excel2007', 'Ods', 'Csv', 'Dbf', 'Dif', 'Sylk', 'Json', 'Xml', 'ImageBmp', 'ImageGif', 'ImageJpeg', 'ImagePcx', 'ImagePng', 'ImageTiff', 'ImageEmf', 'ImageSvg', 'ImageSvgz' **settings** The export settings
exportEmail(format: string, settings?: any, email?: string, subject?: string, message?: string)	Export report to selected format. Use default settings if not specified. Use default email settings if not specified. **format** The format to export, can be 'Document', 'Pdf', 'Xps', 'Ppt2007', 'Html', 'Html5', 'Mht', 'Text', 'Rtf', 'Word2007', 'Odt', 'Excel', 'ExcelBinary', 'ExcelXml', 'Excel2007', 'Ods', 'Csv', 'Dbf', 'Dif', 'Sylk', 'Json', 'Xml', 'ImageBmp', 'ImageGif', 'ImageJpeg', 'ImagePcx', 'ImagePng', 'ImageTiff', 'ImageEmf', 'ImageSvg', 'ImageSvgz' **settings** The export settings **email** The email

© 2003-2024 Stimulsoft

	message The email message **subject** The email subject

9 Reports and Dashboards for JS

JavaScript is a prototype-oriented scripting programming language. Reports.JS provides tools for designing, viewing, converting reports and dashboards for JavaScript.

9.1 Get Started for Browser

This example shows how to install and run Stimulsoft JS for Browser. Also, you can find more samples Reports.JS and Dashboards.JS on the GitHub.

Step 1: Open your html file in editor. For example, **index.html**.

Step 2: Add reference to viewer **CSS file** in HTML head:

index.html

```
...
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>JS Project</title>

   <link href="https://unpkg.com/stimulsoft-dashboards-js/Css/
   stimulsoft.viewer.Office2022.whiteblue.css" rel="stylesheet"/>

   //Also, you must to specify designer css file link, if you need to use
   js designer
   <link href="https://unpkg.com/stimulsoft-dashboards-js/Css/
   stimulsoft.designer.Office2022.whiteblue.css" rel="stylesheet"/>
</head>
</html>
...
```

Step 3: Add the references to stimulsoft js scripts in HTML head.

index.html

© 2003-2024 Stimulsoft

```
...
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8" />
   <title>JS Project</title>

   <link href="https://unpkg.com/stimulsoft-dashboards-js/Css/
   stimulsoft.viewer.Office2022.whiteblue.css" rel="stylesheet"/>

   //Also, you must to specify designer css file link, if you need to use
   js designer
   <link href="https://unpkg.com/stimulsoft-dashboards-js/Css/
   stimulsoft.designer.Office2022.whiteblue.css" rel="stylesheet"/>

   <script src="https://unpkg.com/stimulsoft-reports-js/Scripts/
   stimulsoft.reports.js" type="text/javascript"></script>

   //Also, you must to specify dashboard script link, if you need to use
   dashboards
   <script src="https://unpkg.com/stimulsoft-dashboards-js/Scripts/
   stimulsoft.dashboards.js" type="text/javascript"></script>
   <script src="https://unpkg.com/stimulsoft-reports-js/Scripts/
   stimulsoft.viewer.js" type="text/javascript"></script>

   //Also, you must to specify designer script link, if you need to use js
   designer
   <script src="https://unpkg.com/stimulsoft-reports-js/Scripts/
   stimulsoft.designer.js" type="text/javascript"></script>
</head>
</html>
...
```

Information

It is important to follow the order in which scripts are connected:

- The **stimulsoft.reports.js** script should be the first;
- The **stimulsoft.viewer.js** script must be earlier than **stimulsoft.designer.js** script.

Step 4: Create the **onLoad()** function and a new viewer, if you want to show report or dashboard.

index.html

```
...
<script type="text/javascript">
function onLoad(){
```

© 2003-2024 Stimulsoft

```
   // Create the report viewer with default options
   var viewer = new Stimulsoft.Viewer.StiViewer(null, "StiViewer", false);

   // Show the report viewer in div content
   viewer.renderHtml("content");
}
</script>
...
//Call function onLoad() and put viewer in <div>.
<body onLoad="onLoad()">
   <div id="content">Component should be here</div>
</body>
...
```

Step 5: Create a new report object, load a report template file in it, and assign report object to the report viewer.

index.html

```
...
<script type="text/javascript">
function onLoad(){
...
   // Create a new report object
   var report = new Stimulsoft.Report.StiReport();

   // Load report template in the report object
   report.loadFile("report url");

   // Assign report object to report viewer
   viewer.report = report;
...
}
</script>
...
```

Step 6: Create the **onLoad()** function and a new designer, if you want to create or edit report or dashboard.

index.html

```
...
<script type="text/javascript">
function onLoad(){
   // Create the report designer with default options
   var designer = new Stimulsoft.Designer.StiDesigner(null, "Designer",
   false);

   // Show the report designer in div content
   designer.renderHtml("content");
```

© 2003-2024 Stimulsoft

```
    // Create a new report object
    var report = new Stimulsoft.Report.StiReport();

    // Load report template in the report object
    report.loadFile("report url");

    // Assign report object to report designer
    designer.report = report;
}
</script>
...
//Call function onLoad() and put designer in <div>.
<body onLoad="onLoad()">
    <div id="content">Component should be here</div>
</body>
...
```

Step 7: Save changes and open **index.html** file in browser.

9.2 Get Started for Node.js

This example shows how to install and run Stimulsoft JS for Node.js. Also, you can find more samples Reports for Node.js and Dashboards for Node.js on the GitHub.

Step 1: Create a folder;

Step 2: Add a report template in this folder;

Step 3: Create **index.js** file in this folder;

Step 4: Open console;

Step 5: Install the Reports.JS module in this folder:

console

```
npm install stimulsoft-reports-js
```

Or

Install the Dashboards.JS module in this folder:

© 2003-2024 Stimulsoft

console

```
npm install stimulsoft-dashboards-js
```

Step 6: Open **index.js** file in the editor;

Step 7: Add the required code.

index.js

```
...
// Stimulsoft reports module loading
var Stimulsoft = require('stimulsoft-reports-js');

// Loading fonts
Stimulsoft.Base.StiFontCollection.addOpentypeFontFile("Roboto-Black.ttf");

//Creating a new report object
var report = Stimulsoft.Report.StiReport.createNewReport();

// Load report template in the report object
report.loadFile("report1.mrt");

// Save report object in mrt file
report.saveFile("report2.mrt");
...
```

Step 8: Save changes in the **index.js** file;

Step 9: Open console and run **index.js**.

console

```
node index
```

Step 10: You can perform various actions on reports with using **Node.js**. For example, let's export the report to pdf. Open **index.js** in the editor, define this code and save changes.

© 2003-2024 Stimulsoft

index.js

```
...
// Stimulsoft Reports module
var Stimulsoft = require('stimulsoft-reports-js');

// Loading fonts
Stimulsoft.Base.StiFontCollection.addOpentypeFontFile("Roboto-Black.ttf");

// Creating new report
var report = new Stimulsoft.Report.StiReport();

// Loading report template
report.loadFile("report1.mrt");

// Renreding report
report.renderAsync(() => {
   console.log("Report rendered. Pages count: ",
   report.renderedPages.count);

   // Export to PDF
   report.exportDocumentAsync((pdfData) => {

      // Converting Array into buffer
      var buffer = Buffer.from(pdfData)

      // File System module
      var fs = require('fs');

      // Saving string with rendered report in PDF into a file
      fs.writeFileSync('./SimpleList.pdf', buffer);
      console.log("Rendered report saved into PDF-file.");
   },
   Stimulsoft.Report.StiExportFormat.Pdf);
});
...
```

Step 11: Open console and run **index.js**.

console

```
node index
```

9.3 HTML5 Viewer

YouTube

Watch videos for the JS HTML5 Viewer. Subscribe to the Stimulsoft channel to find new video lessons uploaded. . Leave your questions and suggestions in the

© 2003-2024 Stimulsoft

comments to the video.

Samples

See GitHub examples for working with the JS HTML5 Viewer component. All examples are separate projects, grouped into one solution for Visual Studio.

The **HTML5 Viewer** (**StiViewer**) component is designed to view reports in the web browser. You do not need to install the .NET Framework, ActiveX components or any special plug-ins on the client side. All that is needed is any modern Web browser.

With help of **HTML5 Viewer**, you can view, print and export reports on any computer with any operating system installed. Since the viewer only uses HTML and JavaScript technologies, it can be run on devices where there is no Flash or Silverlight support - tablets, smartphones. Also, the viewer supports Mobile and Touch interfaces, which automatically enable when using mobile devices and touch screen monitors.

The **HTML5 Viewer** component uses the **JavaScript** technology to perform all actions (uploading a report, paging, scaling, interactivity in reports, etc.), which allows you to get rid of reloading the entire page and speed up work.

HTML5 Viewer supports many themes, animated interface, bookmarks, interactive reports, editing of report elements on the page, full screen mode, search panel, and other necessary functionality for viewing reports.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Viewer** in project, you need to install the npm package of

© 2003-2024 Stimulsoft

stimulsoft-reports-js:
```
npm install stimulsoft-reports-js
```

If this is not possible, you should add the following scripts to the project.

viewer.html

```
...
<script src="scripts/stimulsoft.report.js"></script>
<script scr="scripts/stimulsoft.viewer.js"></script>
...
```

Install the npm package to use the HTML5 Viewer in a project to view reports and dashboards stimulsoft-dashboards-js:
```
npm install stimulsoft-dashboards-js
```

If this is not possible, you should add the following scripts to the project.

viewer.html

```
...
<script src="Scripts/stimulsoft.reports.js"></script>
<script src="Scripts/stimulsoft.dashboards.js"></script>
<script src="Scripts/stimulsoft.viewer.js"></script>
...
```

Information

The script variants that you can use in your projects can be found in the Scripts of Reports.JS Package and Scripts of Dashboards.JS Package chapters.

9.3.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

The **StiViewer** component is designed to use only HTML5 and JavaScript technology, and does not require a server for its work (it is necessary only for hosting project files). When you run the report viewer, the following actions occur:

- JavaScript component adds code of the viewer interface to the current HTML page;
- If a report object has been assigned, the report will be rendered, then, the first page of the report will be displayed;
- Each action in the viewer (for example, paging, printing or exporting a report, etc.) calls a specific JavaScript event in which you can perform the necessary manipulations with the server report, which eliminates the re-build of the report.

9.3.2 Showing Reports

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `report.render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer.

To show the report, you should add the scripts and styles required for the **StiViewer** component to the HTML page of a project.

viewer.html

```
...
<script src="scripts/stimulsoft.reports.js" type="text/javascript"></
script>
<script src="scripts/stimulsoft.dashboards.js"></script>
<script src="scripts/stimulsoft.viewer.js" type="text/javascript"></
script>
...
```

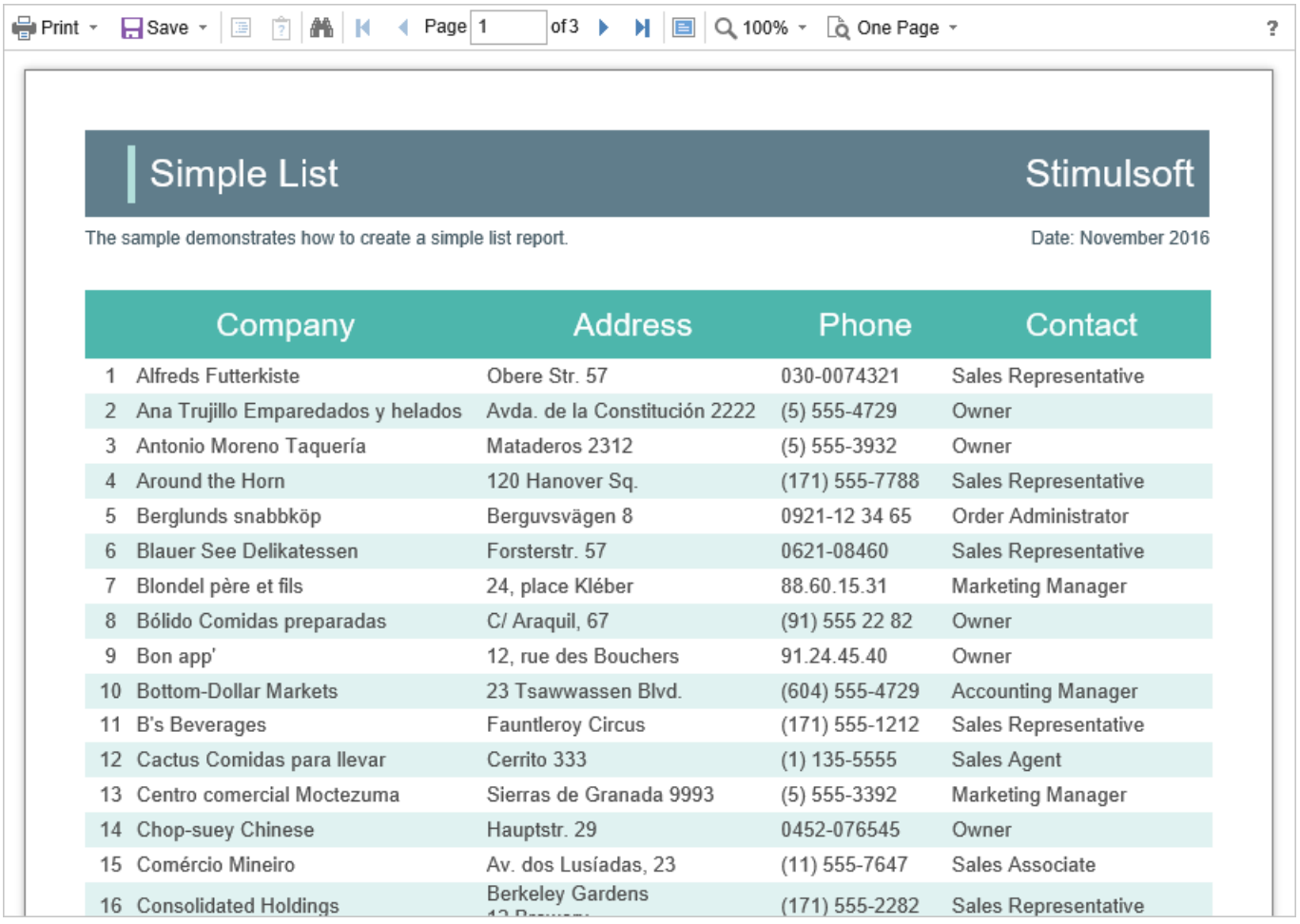
Then you should add JavaScript code of report loading to the HTML page, and

© 2003-2024 Stimulsoft

assign the resulting object to the viewer. In this case, the viewer will be deployed in the current DOM element at the place where the script is located.

viewer.html

```
...
<script type="text/javascript">
   var report = new Stimulsoft.Report.StiReport();
   report.loadFile("SimpleList.mrt");
   //report.loadFile("Dashboard.mrt");

   var viewer = new Stimulsoft.Viewer.StiViewer();
   viewer.report = report;
</script>
...
```



You can create a **StiViewer** object using the **Stimulsoft.Viewer.StiViewer()** constructor, which can accept non-required arguments as input:

© 2003-2024 Stimulsoft

› **options** is a set of options that can be found in the **Stimulsoft.Viewer.StiViewerOptions** class. All options are split into categories. A detailed description of the categories and options can be found in the Viewer Settings topic.
› **viewerId** - viewer identification is used when deploying a component as a DOM object, the default value is “StiViewer”.
› **renderAfterCreate** - defines the viewer location. If it is set to **true**, the viewer will be displayed in the same place in the DOM tree where code to create an object is placed. If it is set to **false**, the viewer will be located at the place where the **renderHtml()** method is called. For example, this can be the initialization of the viewer in the page header.

viewer.html

```
...
<script type="text/javascript">
   var viewer = new Stimulsoft.Viewer.StiViewer(null, "StiViewer", false);
</script>
...
```

And the subsequent output of the viewer in the current DIV element.

viewer.html

```
...
<div>Page content</div>
<div>
   <script type="text/javascript">
   // Render the report viewer in this place
      viewer.renderHtml();
   </script>
</div>
...
```

As an argument of the **renderHtml(id)** output method of the viewer, it is allowed to specify the element identifier of the HTML page in which the viewer should be displayed.

viewer.html

```
...
<script type="text/javascript" >
   var viewer = new Stimulsoft.Viewer.StiViewer(null, "StiViewer", false);
   viewer.renderHtml("content");
```

© 2003-2024 Stimulsoft

```
</script>
...
```

The specified element must be located on the HTML page on which the report viewer is used.

viewer.html

```
...
<div id="content"></div>
...
```

9.3.3 Using Themes

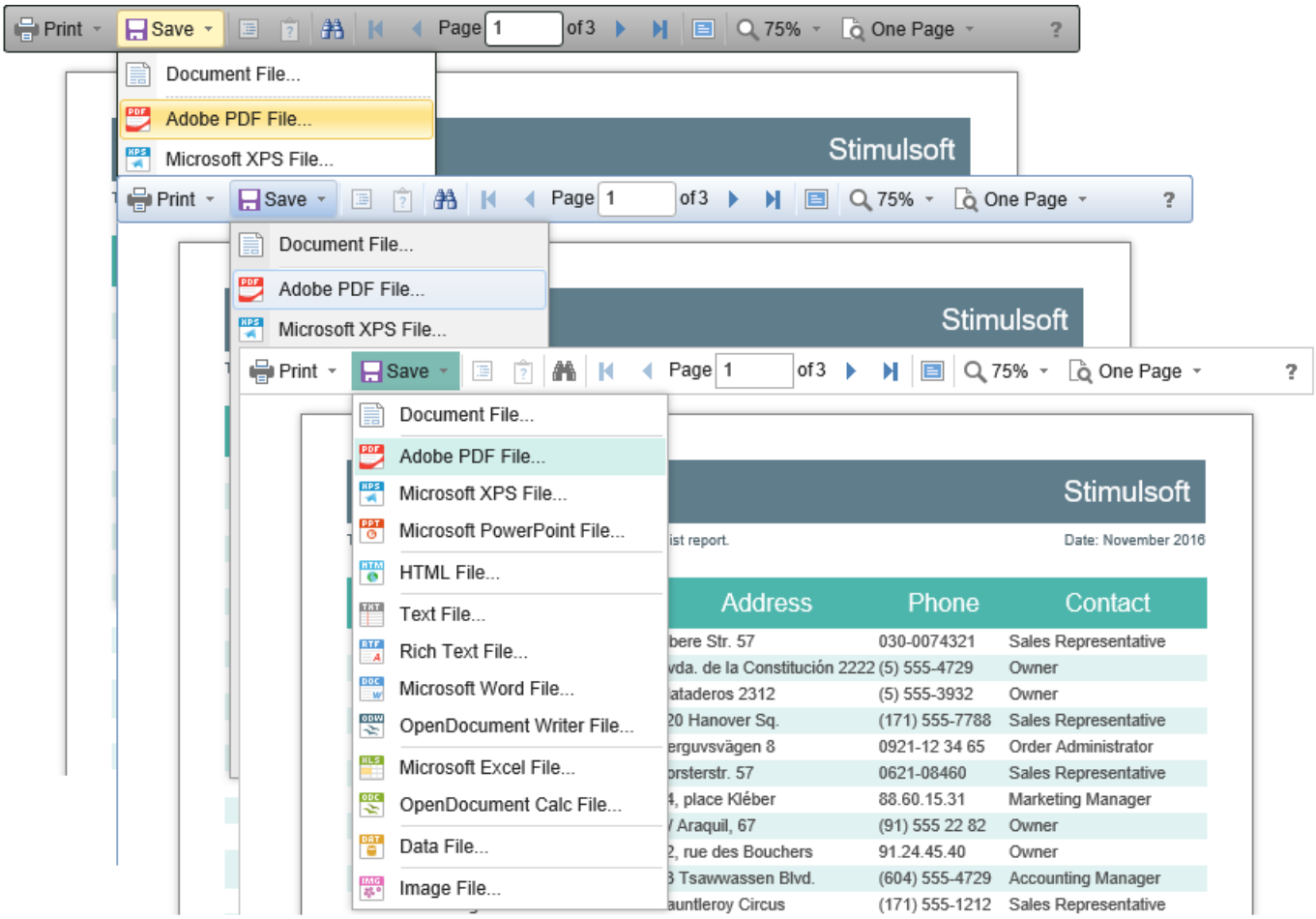
The **HTML5 Viewer** component has the ability to change themes for visual controls. You can use the theme component option or the **setTheme()** method for this.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.appearance.theme =
Stimulsoft.Viewer.StiViewerTheme.Office2022WhiteBlue;
...
viewer.setTheme(Stimulsoft.Viewer.StiViewerTheme.Office2022WhiteBlue);
...
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.

© 2003-2024 Stimulsoft

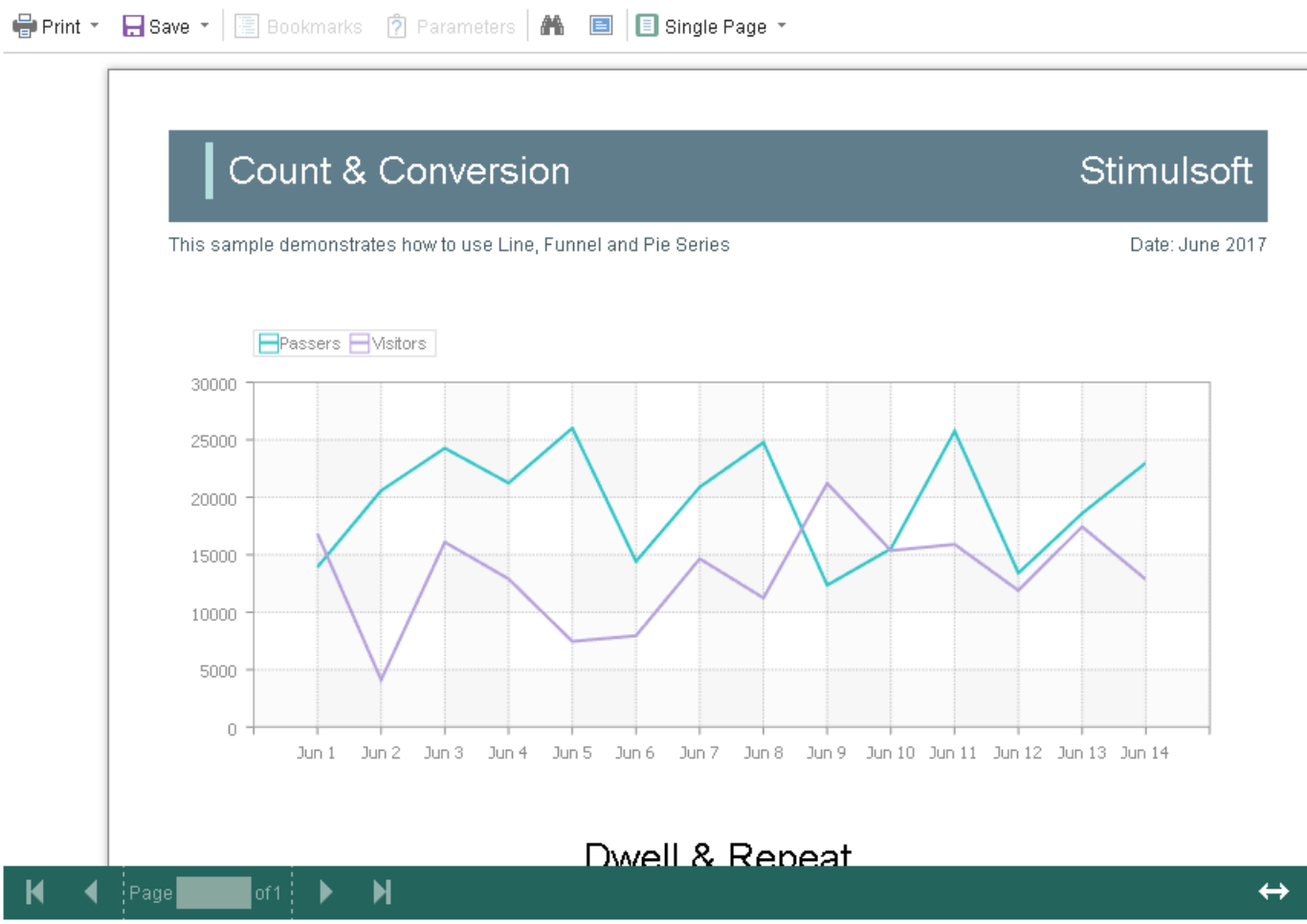


By default, the viewer has only the top toolbar on which all the report controls are located. If necessary, the toolbar can be split into top and bottom parts. The top panel will contain the menu for printing and exporting the report, as well as the buttons for working with parameters and bookmarks. The bottom panel will contain controls to switch between the report pages and the menu to zoom pages. Use the **displayMode** property to enable this mode. The property has values **Simple** and **Separated**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.displayMode =
Stimulsoft.Viewer.StiToolbarDisplayMode.Simple;
options.appearance.scrollbarsMode = true;
...
```

© 2003-2024 Stimulsoft



In addition, it is possible to set the parameters of appearance for the main items of the viewer. For example, you can change the font and color for text of the control panel of the viewer, set the background of the viewer, set the color of page borders, etc. Below is a list of available properties that change the appearance of the viewer, and their default values.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();

options.appearance.backgroundColor =
Stimulsoft.System.Drawing.Color.white;
options.appearance.pageBorderColor = Stimulsoft.System.Drawing.Color.red;
options.appearance.showPageShadow = false;

options.toolbar.backgroundColor = Stimulsoft.System.Drawing.Color.aqua;
options.toolbar.borderColor = Stimulsoft.System.Drawing.Color.darkGreen;
options.toolbar.fontColor = Stimulsoft.System.Drawing.Color.white;
```

© 2003-2024 Stimulsoft

```
options.toolbar.fontFamily = "Arial";
...
```

9.3.4 Printing Reports

Information

Please note that the print option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component has several options for printing a report. Each has its own advantages and disadvantages.

Print to PDF
Printing will be done by exporting the report to the **PDF format**. The advantages are greater accuracy of positioning and printing of the report elements in comparison with other printing options. Among the drawbacks one can mention the mandatory presence of a plug-in installed in a web browser for viewing PDF files (modern browsers have embedded PDF viewer and printer).

Information

Internet Explorer and Edge browsers do not support direct output of PDF content from JavaSctipt code, so when printing as PDF, you will be prompted to save the file, and only then it can be printed.

Print with Preview
The report will be printed in a separate pop-up browser window in the **HTML format**. The report can be previewed, and then sent to the printer or copied to another location as text or HTML code. The advantages are cross-browser compatibility when printing, no need to install special plug-ins. The disadvantage is the relatively low accuracy of the position of the report elements, due to the issues with implementation of HTML formatting.

Print without Preview

© 2003-2024 Stimulsoft

The report will be printed directly to the printer without preview. After selecting this menu item, the system print dialog is displayed. Since printing in this mode is carried out in the **HTML format**, then the print quality is similar to the quality of printing a report with a preview.

Information

The report is printed using the built-in methods of the current browser, so the presentation of the dialog box may differ in operating systems and browsers. Also, the browser does not allow managing print settings from JavaScript code, so the required settings will need to be performed in the dialog box.

Print setup

If you choose printing a report in the viewer panel, a menu with printing options is displayed. The **HTML5 Viewer** component is able to force the required printing mode. To do this, set the **printDestination** property to one of the following values of the **StiPrintDestination** enumeration.

- **Default** – the menu will be displayed (the default property value);
- **Pdf** – print to the PDF format;
- **Direct** – printing to the HTML format directly to the printer, the system print dialog will be displayed;
- **WithPreview** – print to the HTML format with preview in a pop-up window.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.printDestination =
Stimulsoft.Viewer.StiPrintDestination.Default;
...
```

The **HTML5 Viewer** component is able to completely disable report printing. To do this, set the value of the **showPrintButton** property to **false**.

viewer.html

© 2003-2024 Stimulsoft

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.showPrintButton = false;
...
```

Print Report from Code

Also it is possible to print a report using the code. To do this, you can use the special **print()** method on the report object.

viewer.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.renderAsync(function(){
   report.print();
});
...
```

When printing a report, it is possible to specify a print range. The special **StiPagesRange** class is used for this, it is allowed to set the following parameters as the arguments of the constructor.

- Range type (the following values are available **Stimulsoft.Report.StiRangeType.All**, **Stimulsoft.Report.StiRangeType.Pages**, **Stimulsoft.Report.StiRangeType.CurrentPage**);
- Range as a string representation (page numbers are separated with commas or hyphenated);
- Current page number.

viewer.html

```
...
var pageRange = new
Stimulsoft.Report.StiPagesRange(Stimulsoft.Report.StiRangeType.CurrentPage
, "1,3-8", 5);
report.print(pageRange);
...
```

9.3.5 Exporting Reports

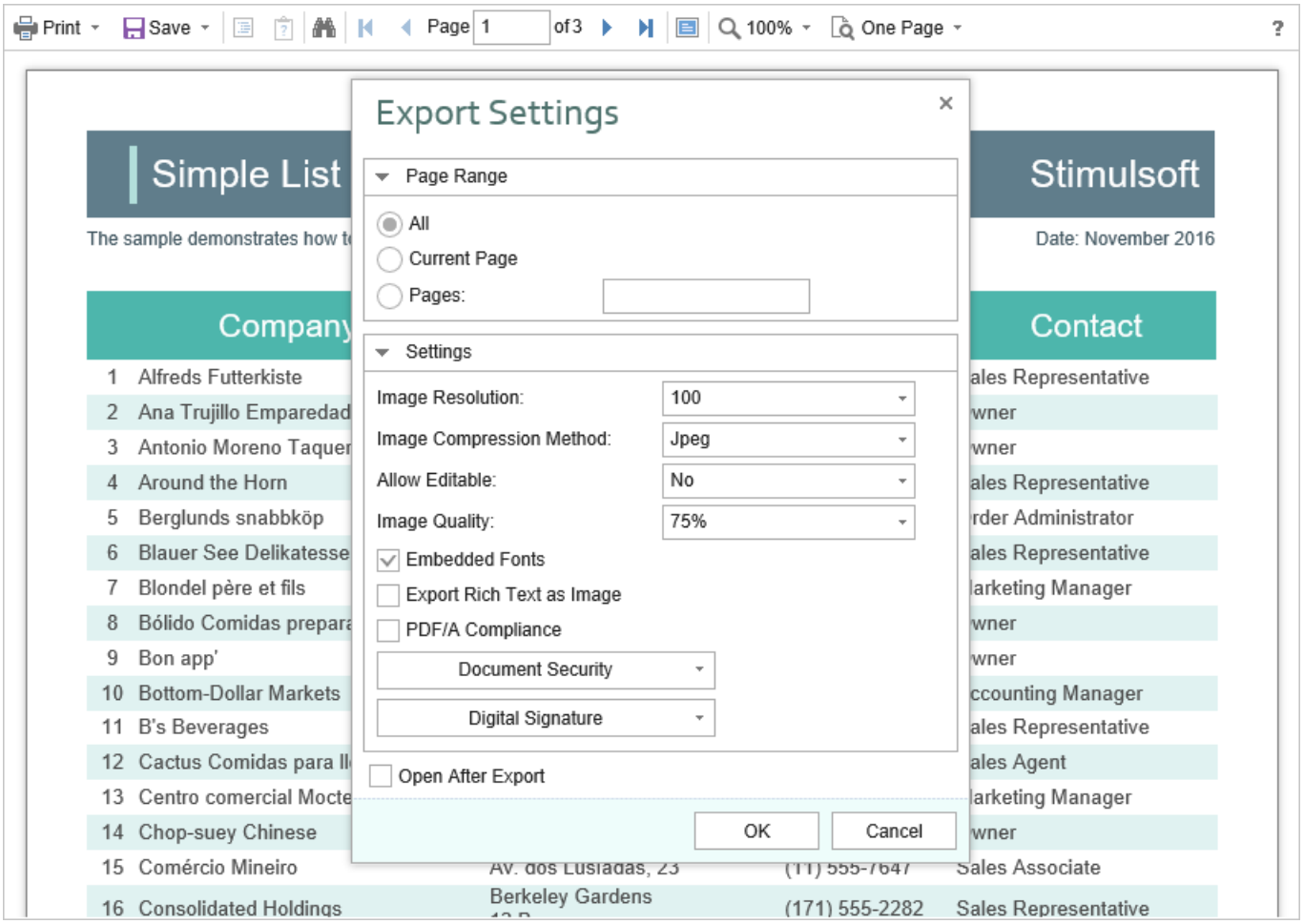
Information

Since dashboards and reports use the same unified template format - MRT,

© 2003-2024 Stimulsoft

methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Viewer** component allows you to export the displayed report to around three dozen of various formats, such as **PDF**, **HTML**, **Word**, **Excel**, **CSV**. The export function does not require additional settings of the viewer. You may export dashboard to PDF, Excel files.



Export events

To perform any actions before exporting a report, a special **onBeginExportReport** event is used. In this event, you can find out the type of report export, get the report, as well as get the report export settings and, if necessary, change them.

© 2003-2024 Stimulsoft

viewer.html

```
...
viewer.onBeginExportReport = function (event) {
   switch (event.format) {
      case Stimulsoft.Report.StiExportFormat.Html:
         event.settings.zoom = 2;  // Set zoom to 200%
         break;
   }
}
...
```

Read more about viewer events.

Export Settings

The **HTML5 Viewer** component contains 7 export formats, and sometimes you need to disable unused formats. This allows you to simplify UI and the use of the viewer. To disable unused export formats, it is enough to set the values for the corresponding properties of the viewer listed in the list below to **false**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.exports.showExportToDocument = false;
options.exports.showExportToPdf = false;
options.exports.showExportToHtml = false;
options.exports.showExportToHtml5 = false;
options.exports.showExportToWord2007 = false;
options.exports.showExportToExcel2007 = false;
options.exports.showExportToCsv = false;
...
```

Also, if required, you can completely remove displaying dialog boxes of export, the export will always be carried out with the default settings. You should to set the value of the **showExportDialog** property to **false**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.exports.showExportDialog = false;
...
```

© 2003-2024 Stimulsoft

Export Report from Code

Also you can export the report using code. You can use the special **exportDocument()** method on the report object.

viewer.html

```
...
// Create a new report instance
var report = new Stimulsoft.Report.StiReport();
// Load report from url
report.loadFile("../reports/SimpleList.mrt");
// Render report
report.renderAsync(function(){
   report.exportDocument(Stimulsoft.Report.StiExportFormat.Pdf); // Export
   report to PDF format
});
...
```

9.3.6 Viewing Modes

The **HTML5 Viewer** component has two modes for displaying reports - with and without scrollbars. By default, the view mode without scrollbars is set. To enable the scrollbar view mode, set the value of the **scrollbarsMode** property to **true**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.appearance.scrollbarsMode = true;
...
```

In the first mode (without scrollbars), the viewer displays a page or report entirely, automatically stretching the preview space. If the width and height are specified, the viewer will truncate the page that is out of bounds. In the second mode, unlike the first one, when the page is out of bounds of the viewer's size, no truncation will be performed. Scrollbars will appear, using which you can view the entire page or report.

Information

In the report mode with scrollbars, you should set the height of the viewer;

© 2003-2024 Stimulsoft

otherwise the default height will be set to **650 pixels**.

The **HTML5 Viewer** component has the full-screen report and dashboard mode. By default, the standard view mode is enabled; the viewer has the specified size in the settings. To enable the full-screen mode, set the **fullScreenMode** property to **true**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.appearance.fullScreenMode = true;
...
```

Also, to enable or disable the full-screen mode, you can use the corresponding button on the control panel of the viewer.

The **HTML5 Viewer** component has three modes to display reports - page-by-page, entire report, and tabular display of report pages. To control the modes, the **viewMode** property is used. It can have one of the specified values - **SinglePage**, **Continuous**, **MultiplePages**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.viewMode = Stimulsoft.Viewer.StiWebViewMode.Continuous;
...
```

The **HTML5 Viewer** component supports interaction on a regular PC and touch screen displays and on mobile devices. The **interfaceType** property allows you to control the interface modes. The property can have one of the following values:
› **Auto** – the interface of the viewer is determined automatically depending of the device that the report is displayed on. This is the default value.
› **Mouse** – the standard interface with a mouse control will be used for all the screen types.
› **Touch** – the touch interface will be used to control the viewer. The interface design was optimized for the touch screen displays. The elements of the viewer interface are increased in size to simplify the control of the viewer and to improve its

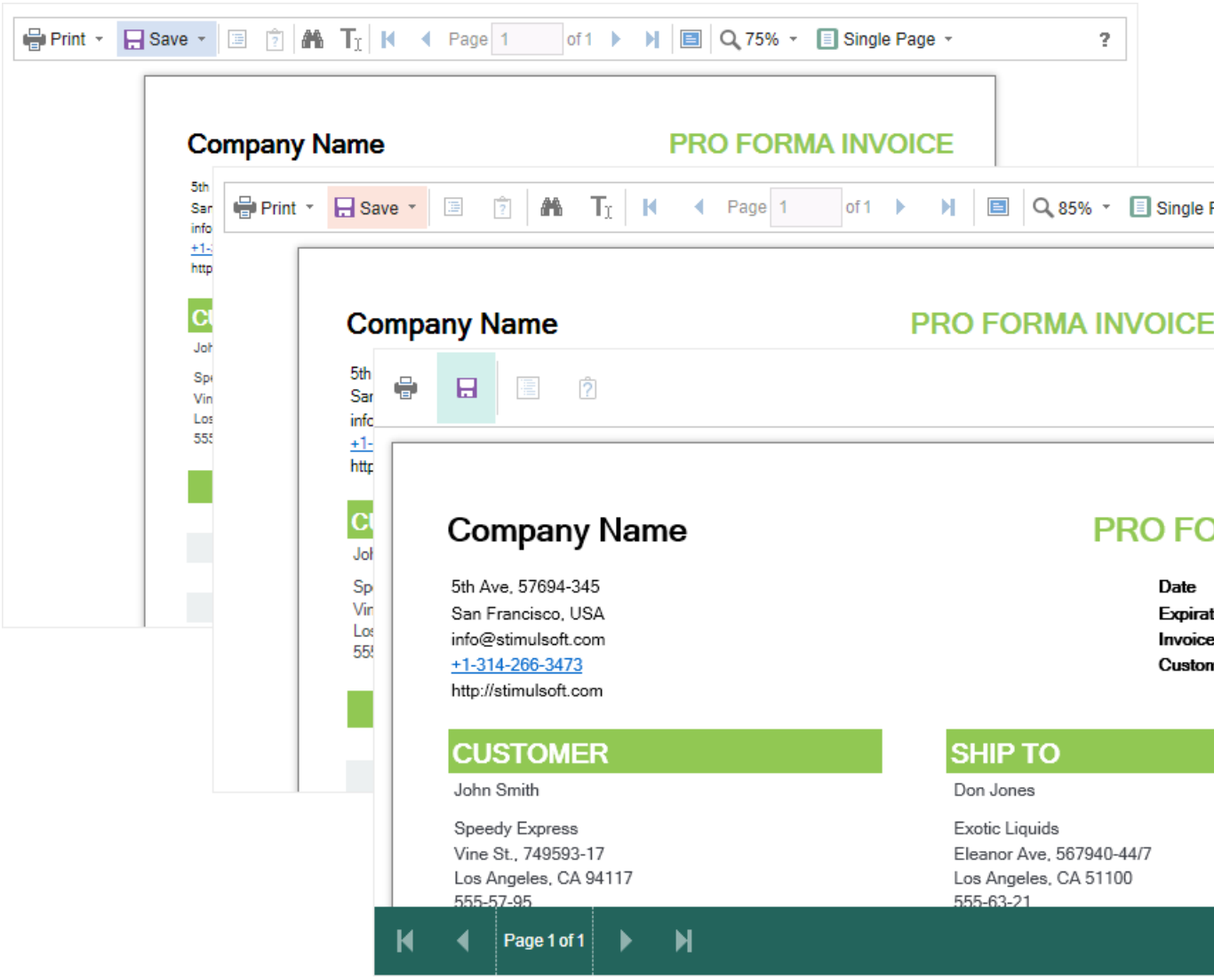
© 2003-2024 Stimulsoft

usability.

› **Mobile** - the mobile interface will be used to control the viewer for all the screen types. The mobile interface was designed to control the viewer using mobile smartphone displays. This interface design was simplified and adapted to use with the smartphones.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.appearance.interfaceType =
Stimulsoft.Viewer.StiInterfaceType.Auto;
...
```



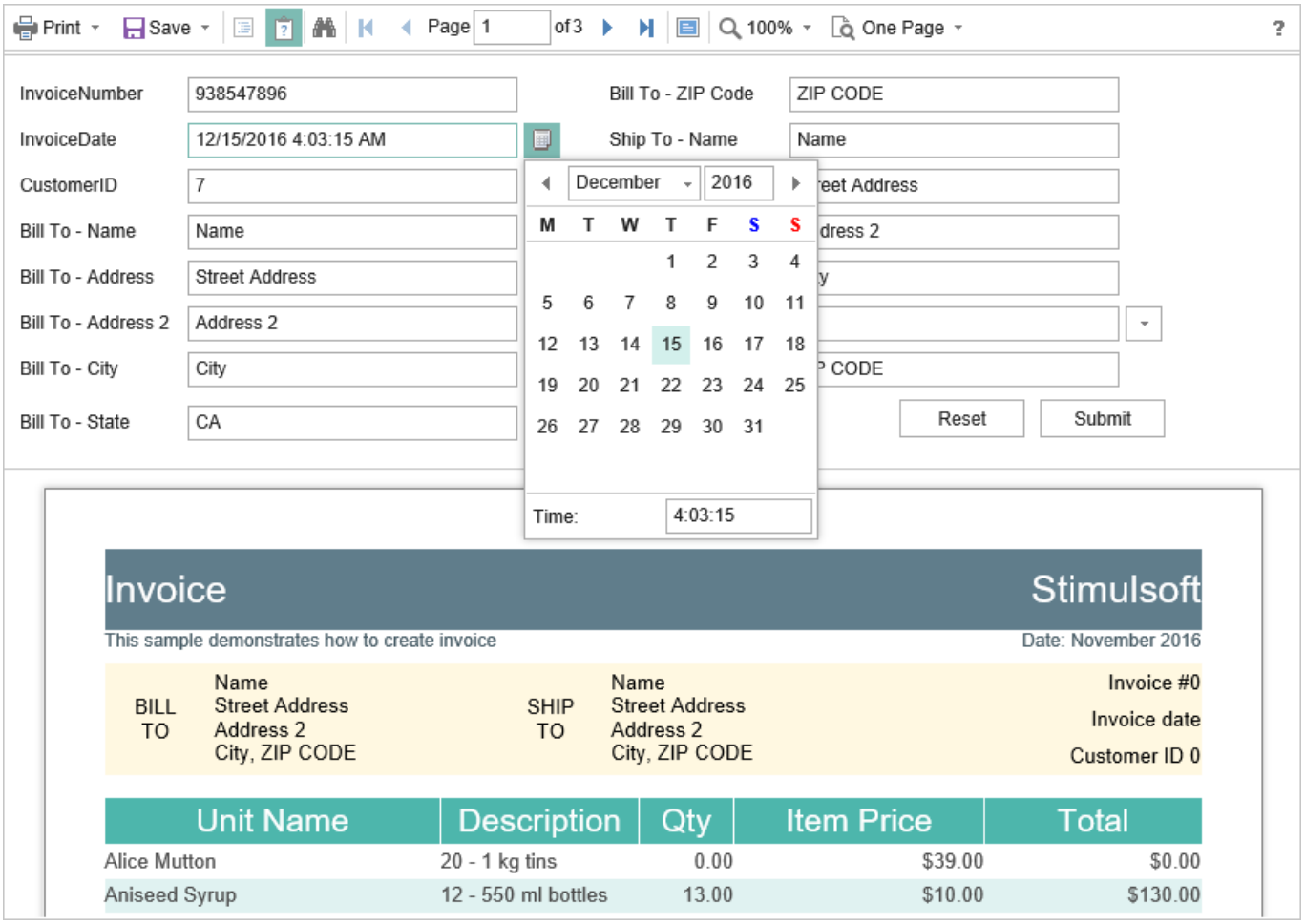
© 2003-2024 Stimulsoft

9.3.7 Work with Parameters

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

We have a special settings panel to work with report parameters in the **HTML5 Viewer**. To add a parameter to the panel you need to define a variable in a report, requested by the user. When viewing a report in the viewer such a variable will be automatically added to the settings panel. It supports all types of report variables (normal variables, date and time, borders, lists, etc.).



© 2003-2024 Stimulsoft

To work with reports with parameters, no additional viewer settings are required. If you need to perform some actions before applying the parameters, you can define a special **onInteraction** event. When using the parameters panel, the action type will be set to the **Variable** value.

viewer.html

```
...
viewer.onInteraction = function (args) {
   if (args.action == "Variables") {
      var variables = args.variables;
   }
}
...
```

If you do not need to work with parameters, you can completely disable this feature. To do this, use the **showParametersButton** property in the **Toolbar** section of properties, which should be set to **false**.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.showParametersButton = false;
...
```

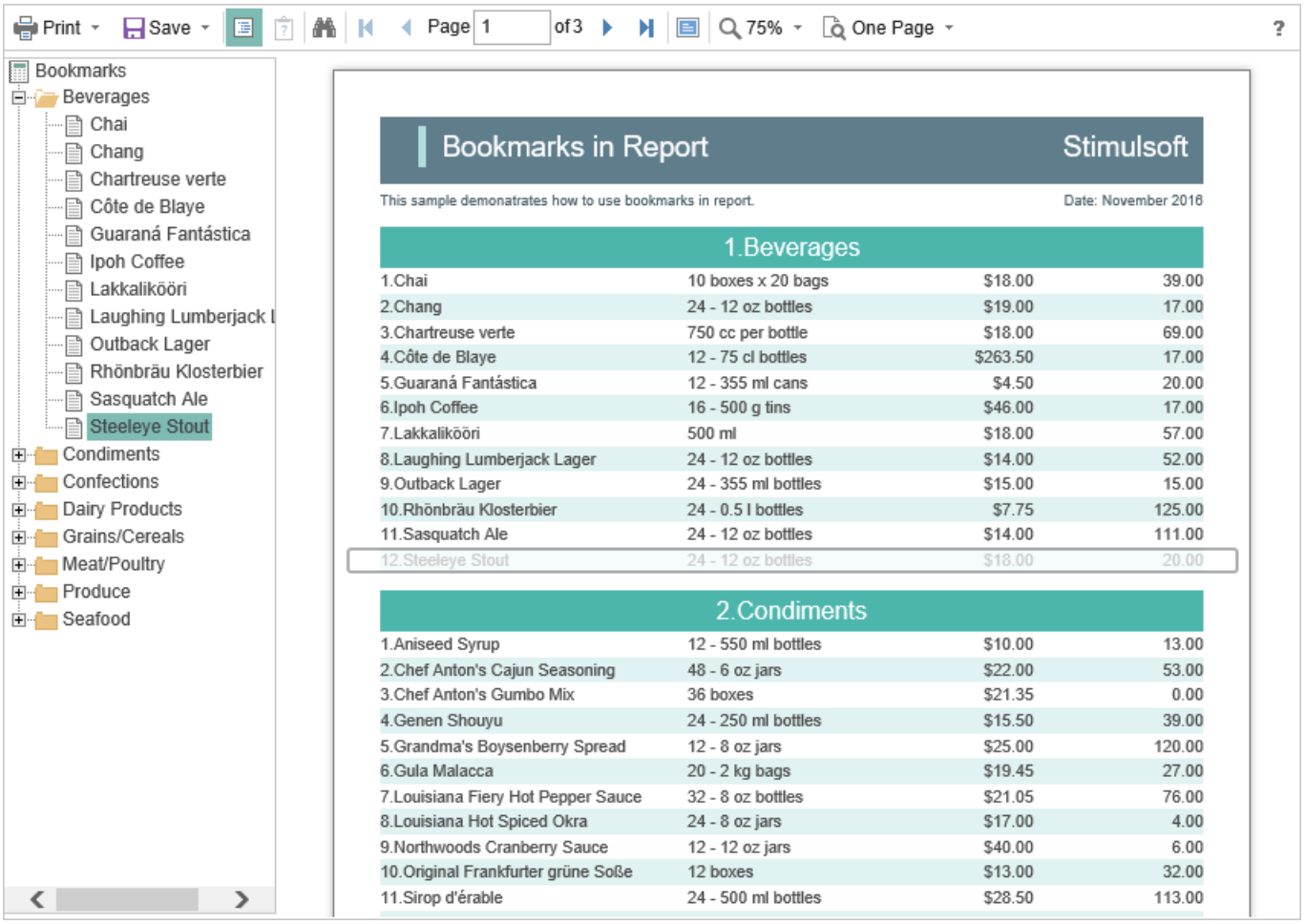
Information

With such a viewer configuration, the options panel will not be displayed, even if the parameters are present in the displayed report.

9.3.8 Work with Bookmarks

The **HTML5 Viewer** component supports report bookmarks. When displaying such a report on the left side of the page, a panel with bookmarks will be displayed. When you select a bookmark of the report, the viewer will carry out an automatic transition to the specified page, and the report item with a bookmark is highlighted.

© 2003-2024 Stimulsoft



By default, the bookmarks bar width is 180 pixels. The **HTML5 Viewer** component allows you to change this value. The **bookmarksTreeWidth** property, which value is specified in pixels, is used.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.appearance.bookmarksTreeWidth = 200;
...
```

If work with report bookmarks is not required, you can disable this functionality. For this, you should set the **showBookmarksButton** property to **false**.

viewer.html

© 2003-2024 Stimulsoft

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.showBookmarksButton = true;
...
```

Information

In this case, report bookmarks will not be displayed, even if they are present in the displayed report. This property has no effect on printing and exporting reports.

When printing a report with bookmarks, the bookmark tree will be hidden. If you want to print bookmarks with the report, it is necessary to set the **bookmarksPrint** property to **true**.

viewer.html

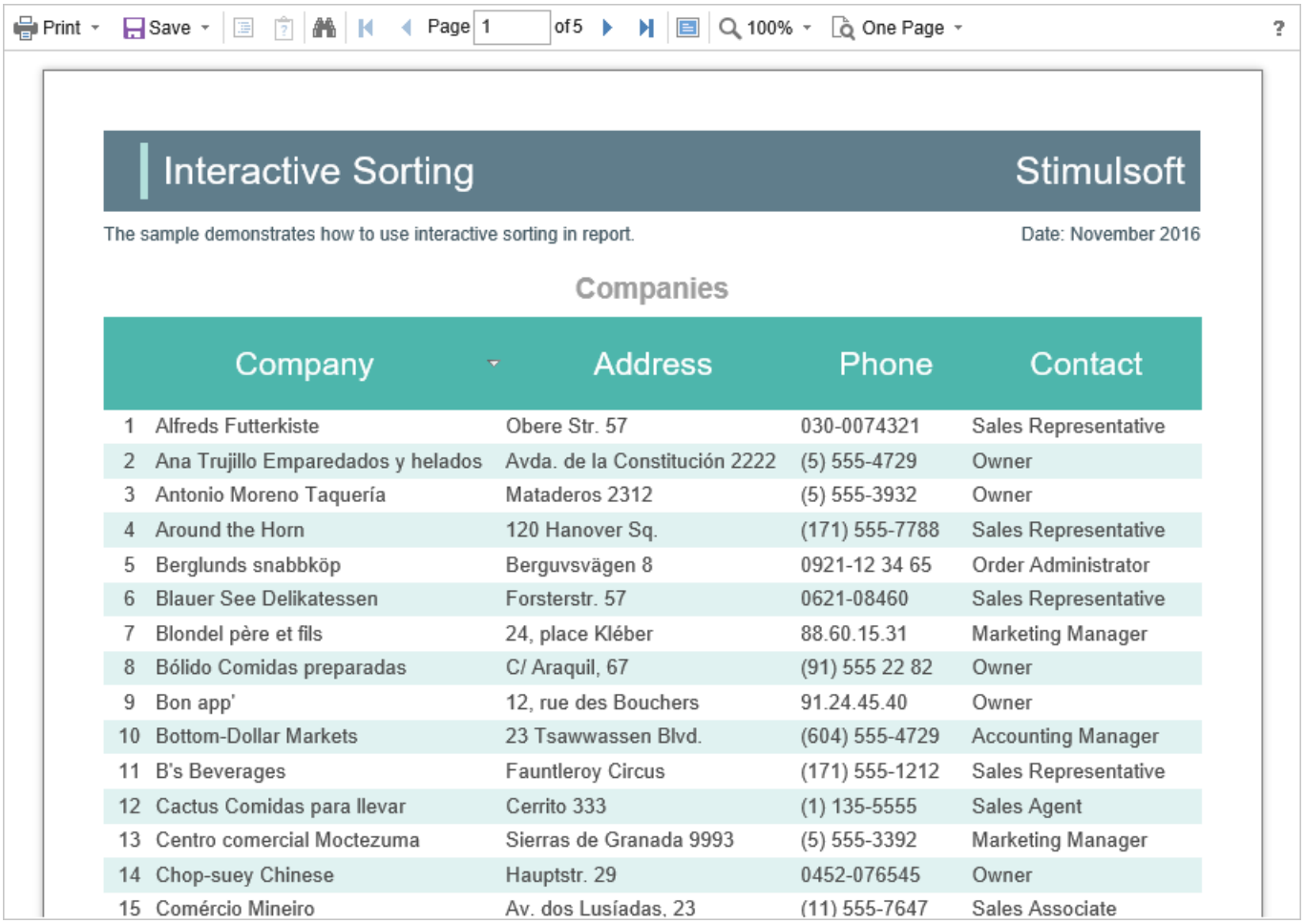
```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.appearance.bookmarksPrint = true;
...
```

9.3.9 Dynamic Sorting, Collapsing, and Drill-Down

The **HTML5 Viewer** component supports dynamic sorting, collapsing, and drill-down reports. Dynamic sorting provides the ability to change the direction of sorting in a rendered report. To do this, click on the component that has the dynamic sorting enabled. Dynamic sorting is carried out in the following directions - **Ascending** and **Descending**. Each time the component is clicked, the sorting direction is reversed.

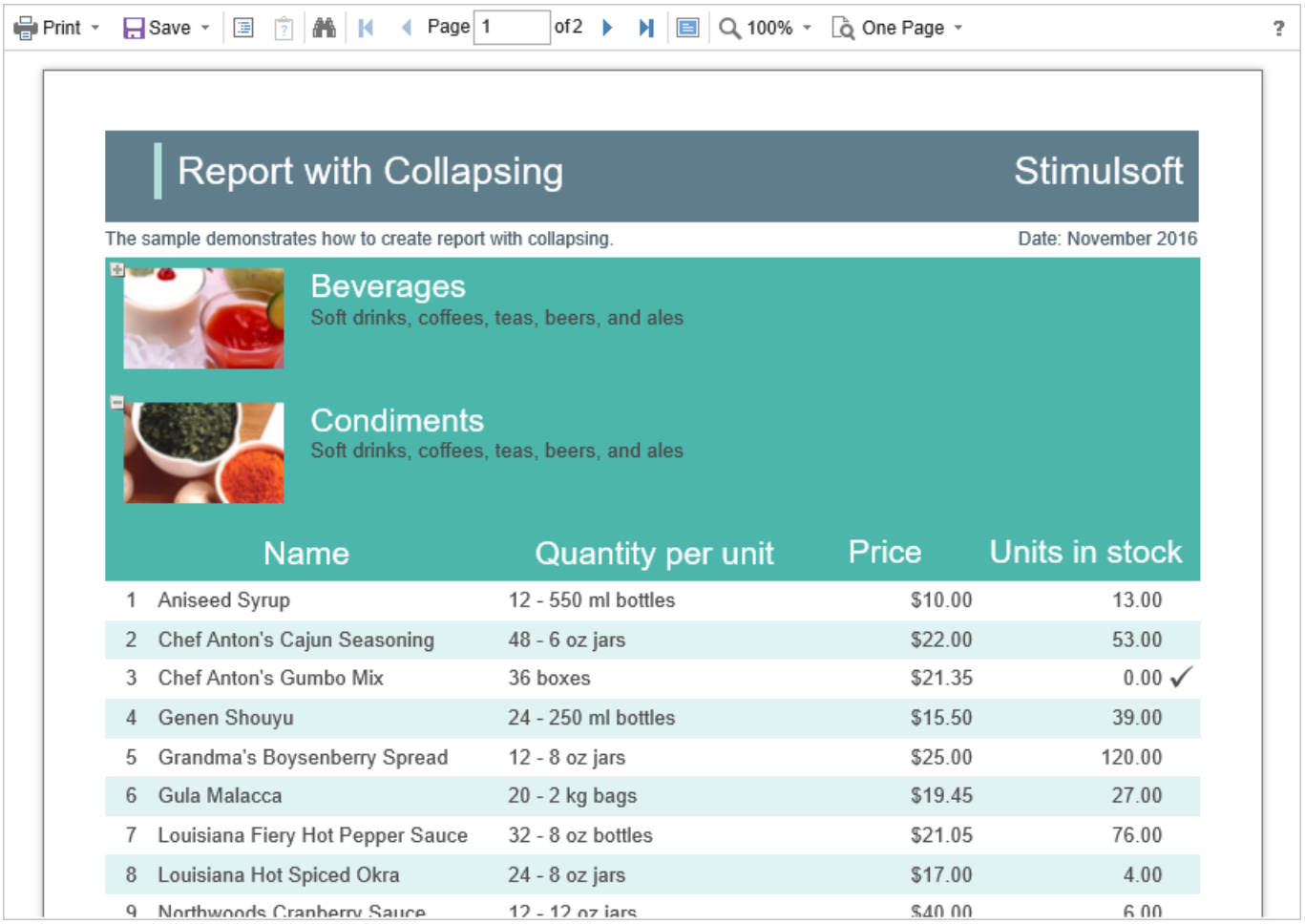
Multi-level sorting is allowed in the report. To do this, hold down the **Ctrl** key and sequentially click on the sorted components in the report. To reset sorting, you can click on any sorted component without holding down the **Ctrl** key.

© 2003-2024 Stimulsoft



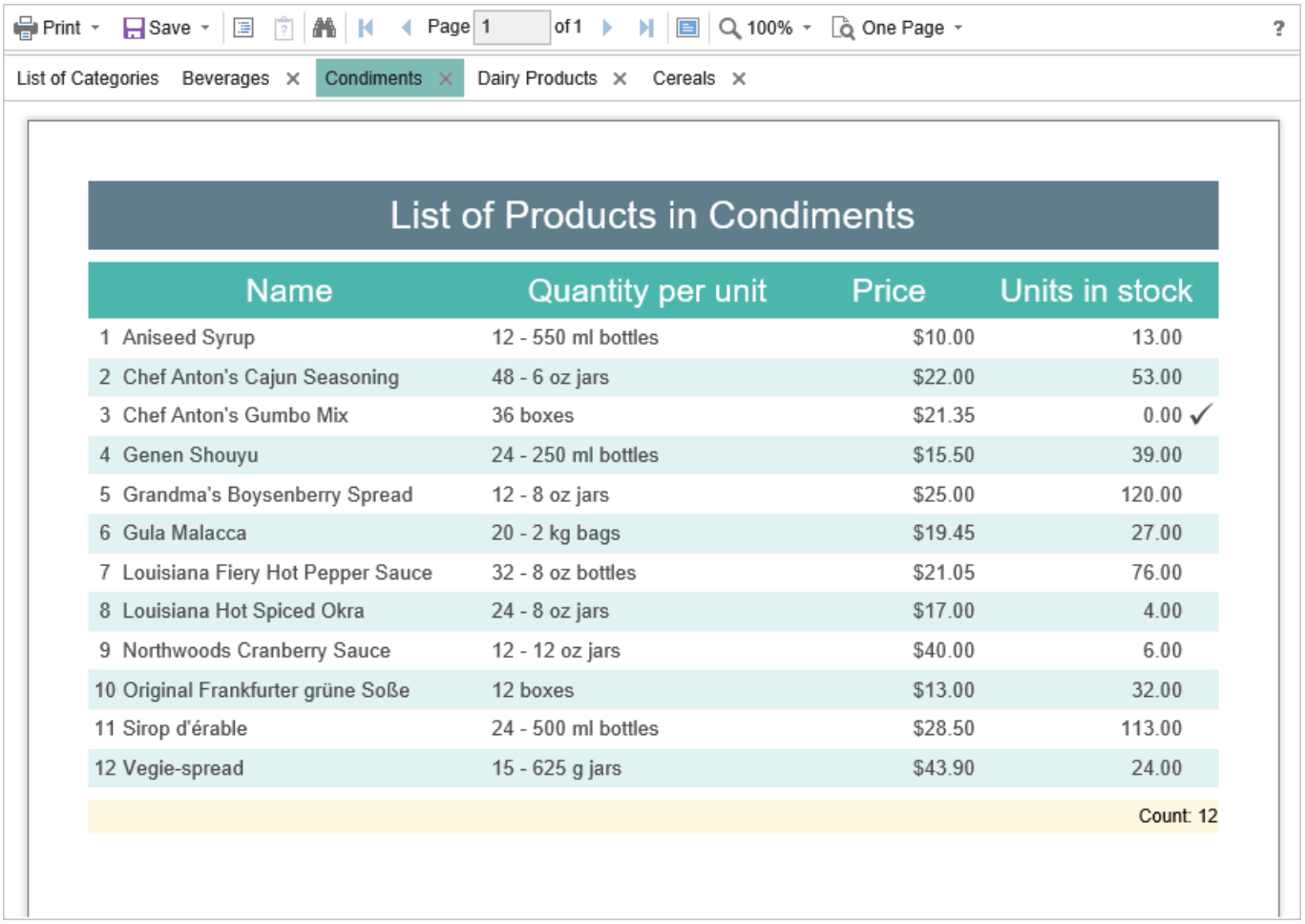
A report with dynamic collapsing is an interactive report in which blocks can collapse/expand their content when you click on the block title. Report elements, which can be collapsed/expanded, are indicated by special icons - **[-]** or **[+]**.

© 2003-2024 Stimulsoft



When using drill-down under the main panel of the viewer, the drill-down panel with tabs for drill-down reports will be displayed. The currently displayed report will be highlighted.

© 2003-2024 Stimulsoft



For the work with dynamic sorting, folding and detailing of reports, no additional settings of the viewer are required. The special **onInteraction** event is used to perform any actions before sorting, minimizing or detailing a report. It will be triggered by interactive actions of the viewer. For each type of interactivity, the viewer provides a certain type of action:

- **Sorting** – when using column sorting;
- **DrillDown** – when using drill-down in reports;
- **Collapsing** – when using collapsing report blocks.

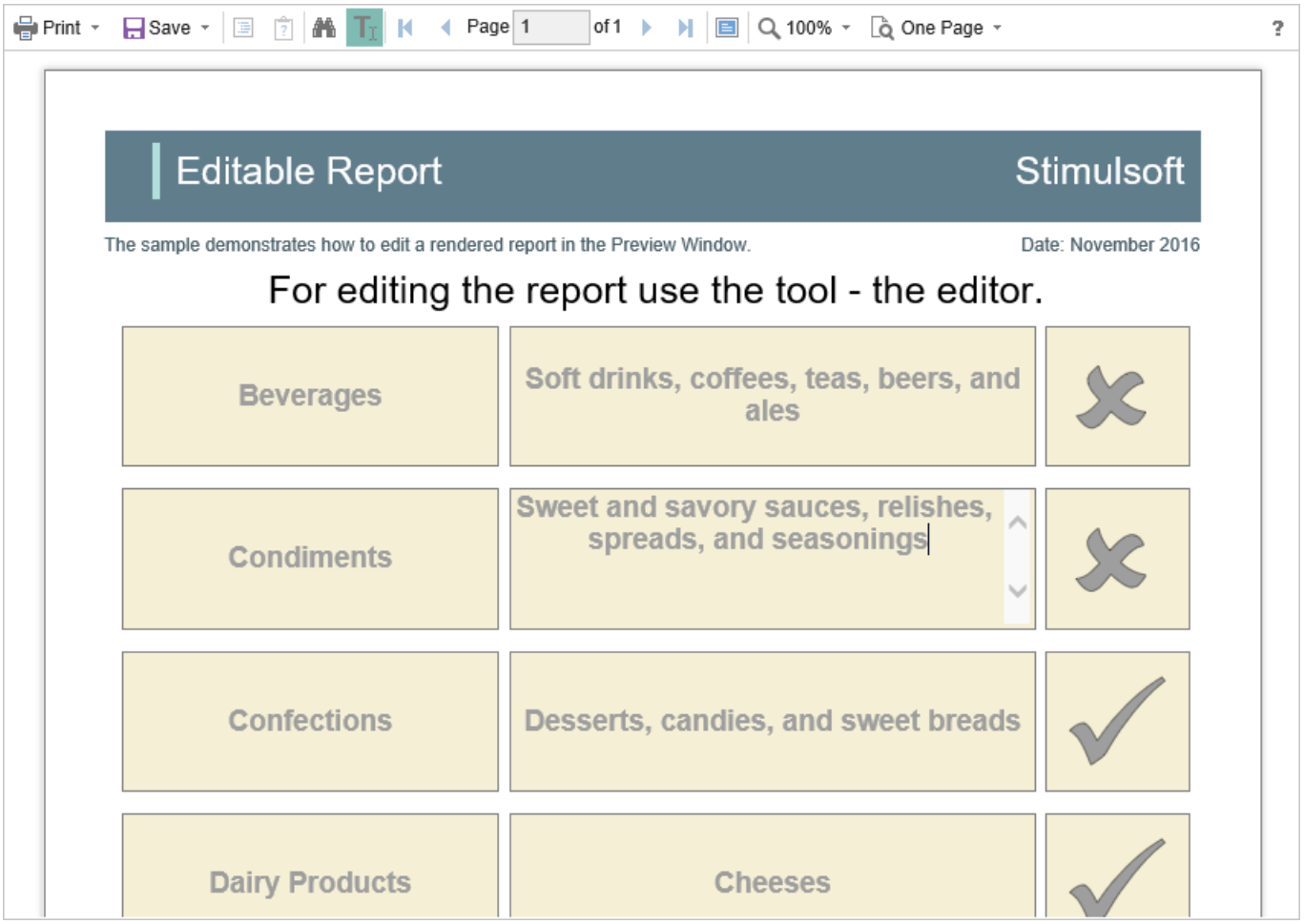
viewer.html

```
...
viewer.onInteraction = function (args) {
   switch (args.action) {
      case "Sorting":
        break;

      case "DrillDown":
```

```
        break;

      case "Collapsing":
        break;
    }
}
...
```

9.3.10 Editing Report

The **HTML5 Viewer** component has the ability to edit report items, such as text boxes and check boxes. In order the editing be possible, in the report template, you should mark the required components as editable. After displaying a report in the viewer, you need to click the corresponding button on the viewer panel to start editing. After editing, it is necessary to click the button once more, and all changes will be applied to the report.



For the report edit mode, no special settings of the viewer required.

© 2003-2024 Stimulsoft

Information

The edited settings will be applied when you print or export a report, and the original report remains unchanged. After restarting the viewer, all the values will be returned to the initial ones.

9.3.11 Sending Report by Email

Information

Please note that the Send Report by Email option is available only for reports, and not for dashboards.

The **HTML5 Viewer** component provides the ability to send reports by email. To activate this feature, you should set the **showSendEmailButton** property of the viewer to **true**, and add the **onEmailReport** event handler.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.showSendEmailButton = true;
...
```

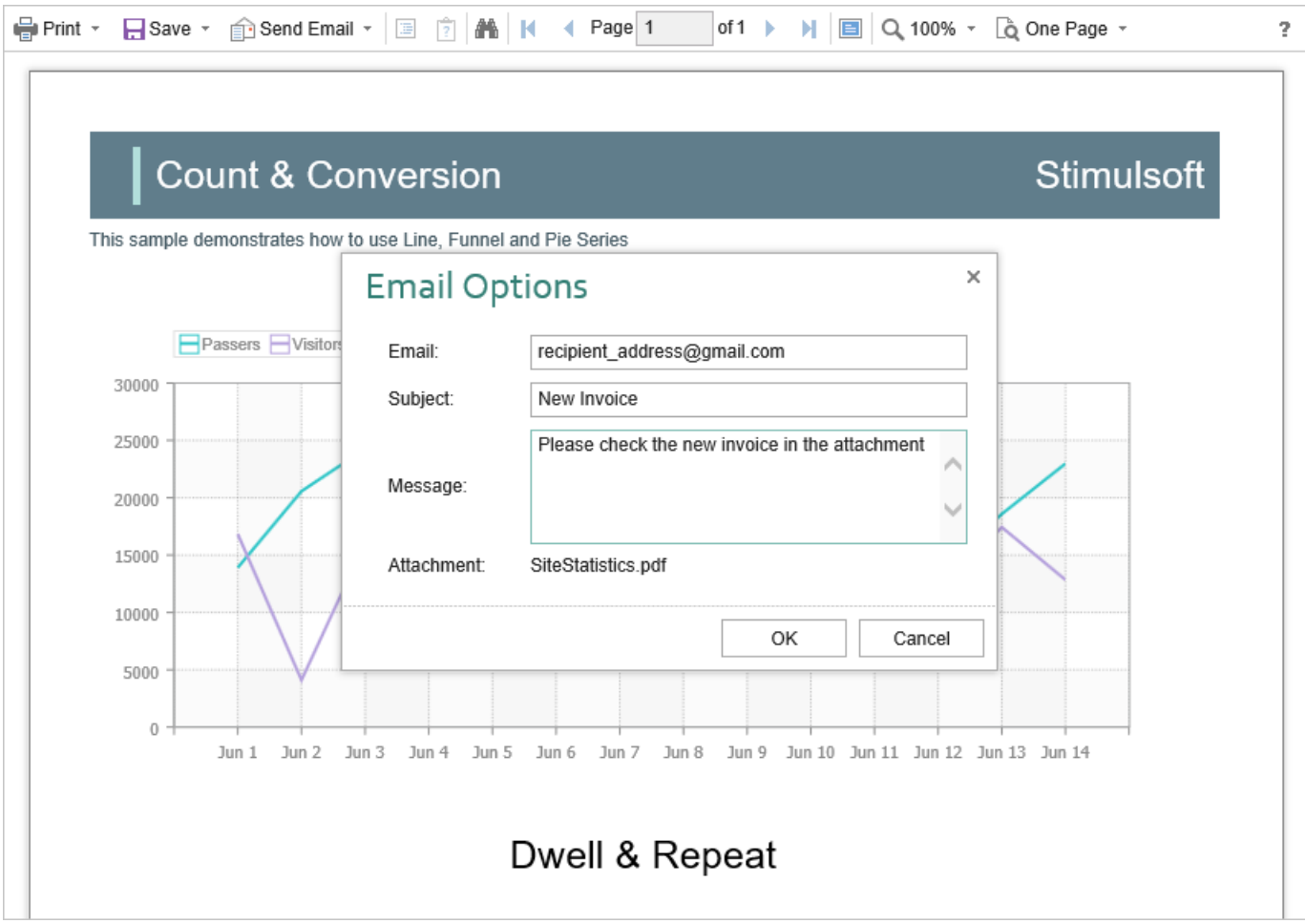
viewer.html

```
...
viewer.onEmailReport = function (args) {
   // args.settings -  send email form
   // args.settings.email  -  email adress
   // args.settings.subject  -  email subject
   // args.settings.message  -  email message

   // args.format  -  export format - PDF, HTML, HTML 5, Excel2007,
   Word2007, CSV
   // args.fileName - report file name (name of attachement)
   // args.data  -  byte array with exported report file
}
...
```

© 2003-2024 Stimulsoft

In the **onEmailReport** event you can find the type of the export, get the report, and also get the report export settings and change them, if necessary. Also in this event, you should set parameters of sending e-mail, such as sender address, server name, port number, user name and password - all these parameters will be used to send email.

When you send a report by email the menu to select the attachment format is displayed. This matches the menu to select an export format. After choosing the format, the dialog to insert send e-mail parameters such as email recipient, subject and message will pop-up.



After the confirmation of sending, the **onEmailReport** event described above will be triggered. You can check and correct the data entered in that form.

Information

© 2003-2024 Stimulsoft

Pure JavaScript does not have built-in e-mail methods, so the **StiViewer** component provides only an interface for sending email. To work with e-mails, you should use any server that supports work with e-mail. To the server side, the data for sending e-mails can be sent using an AJAX request or another convenient method.

The **HTML5 Viewer** component allows you to set default values for the send email form. The **defaultEmailAddress**, **defaultEmailSubject** and **defaultEmailMessage** properties can be used for this. By default, these properties are empty.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.showSendEmailButton = true;

options.email.defaultEmailAddress = "recipient_address@gmail.com";
options.email.defaultEmailSubject = "New Invoice";
options.email.defaultEmailMessage = "Please check the new invoice in the
attachment";
...
```

9.3.12 Calling Designer from Viewer

The **HTML5 Viewer** component has the ability to call the report designer. The special **Design** button in the toolbar of the viewer (the button is disabled by default) should be used. To use this feature, you should set the **showDesignButton** property to **true**, and also to define the **onDesignReport** event handler.

viewer.html

```
...
var options = new Stimulsoft.Viewer.StiViewerOptions();
options.toolbar.showDesignButton = true;
...
```

viewer.html

```
...
viewer.onDesignReport = function (args) {
   window.open("https://www.stimulsoft.com?reportName=" +
   args.report.reportName);
}
...
```

© 2003-2024 Stimulsoft

Information

The viewer does not run the designer itself, it only calls the specified event and passes the previewed report as arguments. In the event, you can set redirect to the HTML page, on which the report designer is placed.

9.3.13 Viewer Events

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

In order to write interactive applications it is needed to respond to changes in the application. The event system which is represented in Stimulsoft Report Viewer can be used for this. The following events exist:

onPrepareVariables

Asynchronous event is called before filling in the variables in the report at the beginning of the report rendering. The event occurs immediately after execution `onPrepareVariables` event of the `StiReport` object. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **PrepareVariables**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.

© 2003-2024 Stimulsoft

async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

viewer.html

```
...
viewer.onPrepareVariables = (args, callback) => {
  args.variables[0].value = "Replace value";
}
...
```

onBeginProcessData

Asynchronous event is called before requesting the data for the report. The event occurs immediately after execution `onBeginProcessData` event of the `StiReport` object. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **BeginProcessData**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
command	a string ID for the current command. Can have values "TestConnection" and "ExecuteQuery" to call the command to test a connection and data acquisition respectively.
database	a string name of the current database.
connection	the name of the current connection to a data

© 2003-2024 Stimulsoft

	source, specified in report template.
headers	an array of the query headers.
withCredentials	an argument is used to provide the ability to set a cookies in the query.
connectionString	a connection string to the data source for a report.
dataSource	the name of the current data source, specified in the report template. It is set only for SQL data sources.
queryString	a string with the SQL query to the database for retrieving data. Used only with the command = "ExecuteQuery".
timeout	an argument is used to provide the ability to define the seconds of the query timeout.
parameters	a list of the query parameters.
escapeQueryParameters	a flag is used to provide the ability to use the escaped part of the request. By default, the value is set to **true**.
pathData	an argument is used to provide the ability to define a path to data file.
tryParseDateTime	a flag is trying to parse data as DateTime.
relationDirection	an argument is used to provide the ability to change the direction of the relation between data sources.
pathSchema	an argument is used to provide the ability to define a path to XSD files.
firstRowIsHeader	a flag is used to provide the ability to use a first row as data header in Excel data source.
collectionName	a string collection name of the OData data source.
separator	a string separator of the CSV data source.
dataType	an argument is used to provide the ability to set the data type of the GIS data source.

© 2003-2024 Stimulsoft

codePage	an argument is used to provide the ability to set a code page of the CSV or DBF data source.

viewer.html

```
...
//Replace connection string
viewer.onBeginProcessData = (args) => {
   if (args.database == "MySQL")
      args.connectionString = "new connection string";
}
...

//Add a some data
viewer.onBeginProcessData = (args, callback) => {
   if (args.database == "MySQL"){
      args.preventDefault = true;
      var result = {
         success: true,
         rows: [
            ["value1", 1, false],
            ["value2", 1, true],
            ["value3", 2, false]
         ],
         columns: [
            "Column1_name",
            "Column2_name",
            "Column3_name"
         ],
         types:[
            "string",
            "int",
            "boolean"
         ]
      }

      // https://github.com/stimulsoft/DataAdapters.JS/
      callback(result);
   }
}
...
```

onEndProcessData

Is called after retrieving data for the report. The event occurs immediately after execution `onEndProcessData` event of the `StiReport` object. The event handler argument “event” is an object with the next fields:

Name	Description

© 2003-2024 Stimulsoft

sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **EndProcessData**.
report	a report object **StiReport**.
command	a string ID for the current command. Can have values "TestConnection" and "ExecuteQuery" to call the command to test a connection and data acquisition respectively.
dataSource	the name of the current data source, specified in the report template. It is set only for SQL data sources.
connection	the name of the current connection to a data source, specified in report template.
database	a string name of the current database.
result	a result dataset in a specific JSON format. It has two collections – columns and rows, with descriptions of columns and rows of data sources.

viewer.html

```
...
viewer.onEndProcessData = (args) => {
   if (args.command == "ExecuteQuery" && args.dataSource == "Categories")
      args.result.rows.push(rowData) ;
      // https://github.com/stimulsoft/DataAdapters.JS/
}
...
```

onPrintReport

Asynchronous event is called before printing the report. This is not relevant when viewing dashboards. The event handler argument “event” is an object with the next fields:

Name	Description

© 2003-2024 Stimulsoft

sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **PrintReport**.
report	a report for saving.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
printAction	a sting method name for printing.

viewer.html

```
...
//Remove image before report print
viewer.onPrintReport = (args) => {
   var page = args.report.renderedPages.getByIndex(0);
   var image = page.components.getByName("Image1");
   if (image)
      page.components.remove(image);
}
...
```

onBeginExportReport

Asynchronous event is called before export but after applying the export options. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **BeginExportReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.

© 2003-2024 Stimulsoft

async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
action	the action, which initiated export event - **StiExportAction**.
settings	export report settings.
format	export report format. Can have the following values: › Stimulsoft.Report.StiExportFormat.Pdf, › Stimulsoft.Report.StiExportFormat.Excel2007, › Stimulsoft.Report.StiExportFormat.Word2007, › Stimulsoft.Report.StiExportFormat.Html, › Stimulsoft.Report.StiExportFormat.Html5, › Stimulsoft.Report.StiExportFormat.Document. › Stimulsoft.Report.StiExportFormat.Text, › Stimulsoft.Report.StiExportFormat.Csv, › Stimulsoft.Report.StiExportFormat.ImageSvg, › Stimulsoft.Report.StiExportFormat.Ppt2007, › Stimulsoft.Report.StiExportFormat.Odt, › Stimulsoft.Report.StiExportFormat.Ods.
formatName	name of a selected report export format corresponds to the name of the constants in format list.
fileName	an exporting file name.
openAfterExport	The flag indicates that a report will be exported in a new browser tab (the `true` value) or after export completed file saving will be invoked (the `false` value).

viewer.html

```
...
viewer.onBeginExportReport = function (args) {
  switch (event.format) {
    case Stimulsoft.Report.StiExportFormat.Html:
    args.settings.zoom = 2;  // Set zoom to 200%
```

© 2003-2024 Stimulsoft

```
      break;
    }
    console.log("exporting");
}
...
```

onEndExportReport

Asynchronous event is called after export report. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, it is set to **EndExportReport**.
fileName	an exporting file name.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
action	the action, which initiated export event - **StiExportAction**.
format	export report format. Can have the following values: › Stimulsoft.Report.StiExportFormat.Pdf, › Stimulsoft.Report.StiExportFormat.Excel2007, › Stimulsoft.Report.StiExportFormat.Word2007, › Stimulsoft.Report.StiExportFormat.Html, › Stimulsoft.Report.StiExportFormat.Html5, › Stimulsoft.Report.StiExportFormat.Document. › Stimulsoft.Report.StiExportFormat.Text,

© 2003-2024 Stimulsoft

	Stimulsoft.Report.StiExportFormat.Csv, Stimulsoft.Report.StiExportFormat.ImageSvg, Stimulsoft.Report.StiExportFormat.Ppt2007, Stimulsoft.Report.StiExportFormat.Odt, Stimulsoft.Report.StiExportFormat.Ods.
formatName	name of a selected report export format corresponds to the name of the constants in format list.
fileName	an exporting file name.
openAfterExport	The flag indicates that a report will be exported in a new browser tab (the `true` value) or after export completed file saving will be invoked (the `false` value).
data	export data to the string or byte array representation.

viewer.html

```
...
viewer.onEndExportReport = (args) => {
   args.fileName = "SampleFileName.txt";
}
...
```

onInteraction

Asynchronous event is called while interactive action of the viewer (dynamic sorting, collapsing, drill-down, applying of parameters) until processing values by the report generator. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, it is set to **Interaction**.
report	a report object **StiReport**.

© 2003-2024 Stimulsoft

preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
action	The identifier of the current interactive action can take the following values: • `Sorting` - This action happens when using sorting columns; • `DrillDown` - This action happens when using sorting columns; • `Collapsing` - This action happens when using of collapsing report blocks.

viewer.html

```
...
viewer.onInteraction = (args) => {
   if (args.action == "Variables")
      args.variables["Variable1"] = "New Value";
}
...
```

onEmailReport

It is called before sending the report by email. This is not relevant when viewing dashboards. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **EmailReport**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

© 2003-2024 Stimulsoft

report	a report object **StiReport**.
settings	export email settings.: › email - address of the recipient; › subject - email subject; › message - email message.
format	export report format. Can have the following values: › Stimulsoft.Report.StiExportFormat.Pdf, › Stimulsoft.Report.StiExportFormat.Excel2007, › Stimulsoft.Report.StiExportFormat.Word2007, › Stimulsoft.Report.StiExportFormat.Html, › Stimulsoft.Report.StiExportFormat.Html5, › Stimulsoft.Report.StiExportFormat.Document. › Stimulsoft.Report.StiExportFormat.Text, › Stimulsoft.Report.StiExportFormat.Csv, › Stimulsoft.Report.StiExportFormat.ImageSvg, › Stimulsoft.Report.StiExportFormat.Ppt2007, › Stimulsoft.Report.StiExportFormat.Odt, › Stimulsoft.Report.StiExportFormat.Ods.
formatName	name of a selected report export format corresponds to the name of the constants in format list.
fileName	an exporting file name.
data	export data to the string or byte array representation.

viewer.html

```
...
viewer.onEmailReport = (args, callback) => {
   args.async = true;

   var emailAddress = args.settings.email;
   var emailMessage = args.settings.message;
   var emailSubject = args.settings.subject;
   var emailAttachmentFileName = args.fileName;
   var emailAttachment = args.data;
   sendEmail(emailAddress, emailMessage, emailSubject,
   emailAttachmentFileName, emailAttachment);
```

© 2003-2024 Stimulsoft

```
   setTimeout(() => {
      callback();
   }, 5000);
}
...
```

onDesignReport

It is called by pressing the Design button. The event handler argument “event” is an object with the following fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event, by default, it is set to **DesignReport**.
report	a report for designing.

viewer.html

```
...
var viewerOptions = new Stimulsoft.Viewer.StiViewerOptions();
viewerOptions.toolbar.showDesignButton = true;
var viewer = new Stimulsoft.Viewer.StiViewer(viewerOptions, "StiViewer",
false);
viewer.renderHtml("content");

viewer.onDesignReport = (args) => {
   var viewerDiv = document.getElementById("content");
   viewerDiv.innerHTML = "";
   var designerOptions = new Stimulsoft.Designer.StiDesignerOptions();
   designerOptions.appearance.fullScreenMode = true;
   var designer = new Stimulsoft.Designer.StiDesigner(designerOptions,
   "StiDesigner", false);
   designer.renderHtml("content");

   designer.report = args.report;
}
...
```

onShowReport

Asynchronous event is called after a report is rendered before its displaying in the

© 2003-2024 Stimulsoft

viewer. The event handler argument “event” is an object with the following fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event, by default, it is set to **ShowReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

viewer.html

```
...
viewer.onShowReport = function (args) {
   console.log("showing");
}
...
```

onOpenReport

Asynchronous event that provides the ability to use custom method of opening templates. The event is called before the dialog box is opened and before report is assigned to the viewer. The event handler argument “event” is an object with the following fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event, by default, it is set to **OpenReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the

© 2003-2024 Stimulsoft

	event. By default, the value is set to **true**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

viewer.html

```
...
viewer.onOpenReport = (args) => {
   args.async = true;
   args.report = anotherReport;
   callback();
}
...
```

onOpenedReport

Asynchronous event that provides the ability to change of the report before it is assigned to the viewer. The event is called after report has been opened and before report is assigned to the viewer. The event handler argument “event” is an object with the following fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event, by default, it is set to **OpenedReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

viewer.html

© 2003-2024 Stimulsoft

```
...
viewer.onOpenedReport = (args) => {
   if (args.report.reportAuthor != "Stimulsoft") {
      args.preventDefault = true;
      window.alert("report.reportAuthor == " + args.report.reportAuthor);
   }
}
...
```

9.3.14 Viewer Settings

The **HTML5 Viewer** is configured using properties that are located in the **Stimulsoft.Viewer.StiViewerOptions** class. All properties are divided into groups. Some of the groups contain subgroups for ease of use. The following is an example of setting the properties of the viewer.

viewer.html

```
...
<script type="text/javascript">
   var report = new Stimulsoft.Report.StiReport();
   report.loadFile("SimpleList.mrt");

   var options = new Stimulsoft.Viewer.StiViewerOptions();

   options.width = "1000px";
   options.height = "1000px";

   options.appearance.theme =
   Stimulsoft.Viewer.StiViewerTheme.Office2022WhiteBlue;
   options.appearance.reportDisplayMode =
   Stimulsoft.Report.Export.StiHtmlExportMode.Div;
   options.appearance.scrollbarsMode = true;
   options.appearance.backgroundColor =
   Stimulsoft.System.Drawing.Color.dodgerBlue;
   options.appearance.showTooltips = false;

   options.toolbar.showPrintButton = false;
   options.toolbar.showDesignButton = false;
   options.toolbar.showAboutButton = false;

   options.exports.showExportToPdf = true;
   options.exports.ShowExportToWord2007 = true;

   var viewer = new Stimulsoft.Viewer.StiViewer(options);
   viewer.report = report;
</script>
...
```

Please note that all dashboard elements have their own save options and full-screen buttons for preview. There are no special options to control displaying them, but

© 2003-2024 Stimulsoft

they can be disabled through the properties of the element. The code below should be added after loading the report before passing it to the viewer.

Default.aspx.cs

```
...
var dbsElementInteraction =
report.getComponentByName("RegionMap1").dashboardInteraction;
  dbsElementInteraction.showFullScreenButton = false;
  dbsElementInteraction.showSaveButton = false;
...
```

Main settings (without groups)

Name	Description
`width`	Sets the width of the component in “px” or “%”.
`height`	Sets the height of the component in “px” or “%”. Works only in the mode if the `options.appearance.scrollbarsMode` property is set to `true`.

Appearance

Name	Description
`options.appearance.theme`	Specifies the theme of the viewers' layout. The list of available themes can be found in the `StiViewerTheme` enumeration. The default value is `Office2022WhiteBlue`.
`appearance.backgroundColor`	Sets the background color of the viewer. By default. it is set to `Stimulsoft.System.Drawing.Color.white`.
`appearance.pageBorderColor`	Sets the border color of the viewer.

© 2003-2024 Stimulsoft

	By default, it is set to `Stimulsoft.System.Drawing.Color.gray`.
`appearance.rightToLeft`	Sets the **Right to Left** mode for viewer controls. By default, the property is set to `false`.
`appearance.fullScreenMode`	Sets the full-screen display mode of the viewer. By default, the property is set to `false`.
`appearance.scrollbarsMode`	Sets the preview mode with scrollbars. By default, the property is set to `false`.
`appearance.openLinksWindow`	Sets the target window to open links contained in the report. By default, it is set to '`_blank`' (new window). It can have one of the standard values '`_blank`', '`_self`', '`_top`', as well as the name of the window or frame.
`appearance.showTooltips`	Enables displaying tips for the viewer controls when the mouse hovers over. By default, the property is set to `true`.
`appearance.showTooltipsHelp`	Sets a value which indicates that show or hide the help link in tooltips. By default, the property is set to `true`.
`appearance.showDialogsHelp`	Sets a value which indicates that show or hide the help button in dialogs. By default, the property is set to `true`.
`appearance.pageAlignment`	Sets the position of the report page in the viewer window. • `Stimulsoft.Viewer.StiContentAlignment.DefaultValue` - page alignment is determined from the template

© 2003-2024 Stimulsoft

	settings (default value); • `Stimulsoft.Viewer.StiContentAlignment.Left` – the page will be aligned left; • `Stimulsoft.Viewer.StiContentAlignment.Center` – the page will be centered; • `Stimulsoft.Viewer.StiContentAlignment.Right` – the page will be aligned right.
`appearance.showPageShadow`	Enables displaying shadow for report pages. By default, the property is set to `true`.
`appearance.bookmarksPrint`	Enables printing of report bookmarks (besides the report itself). By default, the property is set to `false`.
`appearance.bookmarksTreeWidth`	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
`appearance.parametersPanelMaxHeight`	Sets the maximum height of the parameters bar in pixels. By default, the maximum height is 300 pixels.
`appearance.parametersPanelColumnsCount`	Sets the number of columns to display report parameters. By default, it is set to 2 columns.
`appearance.parametersPanelSortDataItems`	Gets or sets a value which indicates that variable items will be sorted. By default the property is set to `true`.
`appearance.parametersPanelDateFormat`	Sets the date and time format for variables of the corresponding type in the parameters panel. By default, the property is set to `String.empty`.
`appearance.reportDisplayMode`	Sets the export mode for displaying report pages. It can take one of the following values of the

© 2003-2024 Stimulsoft

	`reportDisplayMode` enumeration: • `FromReport` - the export mode of the report elements is defined from report template settings - Div or Table (default value); • `Table` – report elements are exported using HTML tables; • `Div` – report elements are exported using DIV markup.
`appearance.interfaceType`	Sets the type of interface used for the viewer. It can take one of the following values: • `Stimulsoft.Viewer.StiInterfaceType.Auto` – the viewer's interface is determined automatically depending of the device that is report is displayed on (default value); • `Stimulsoft.Viewer.StiInterfaceType.Mouse` – the standard interface with a mouse control will be used for all the screen types; • `Stimulsoft.Viewer.StiInterfaceType.Touch` – the Touch interface will be used to control the viewer. The interface design was optimized for the 'touchscreen' display types. The viewer interface elements have been increased in size to simplify the control of the viewer and to improve its usability; • `Stimulsoft.Viewer.StiInterfaceType.Mobile` – the Mobile interface will be used to control the viewer for all screen

© 2003-2024 Stimulsoft

	types. The Mobile interface was designed to control the viewer using the mobile smartphone display. This interface design was simplified and adapted to use on smartphones.
`appearance.allowMobileMode`	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to `false`, then the mobile view will not be used. If the option is set to `true`, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option is set to `true`.
`appearance.chartRenderType`	Sets the chart displaying mode on the report page. • `Stimulsoft.Viewer.StiChartRenderType.AnimatedVector` – charts are displayed in the vector mode as an SVG object, the chart elements are displayed with animation (set by default). • `Stimulsoft.Viewer.StiChartRenderType.Vector` – charts are displayed in the vector mode as an SVG object;
`appearance.datePickerFirstDayOfWeek`	Sets the first day of week in the date picker. • `Stimulsoft.Viewer.StiFirstDayOfWeek.Auto` - Sets **Monday** or **Sunday** as the first day depending on the browser culture default value); • `Stimulsoft.Viewer.StiFirstDayOfWeek.Monday` - Sets **Monday** as the first day of the

© 2003-2024 Stimulsoft

	week; • `Stimulsoft.Viewer.StiFirstDayOfWeek.Sunday` - Sets **Sunday** as the first day of the week.
`appearance.datePickerIncludeCurrentDayForRanges`	Sets a value, which indicates that the current day will be included in the ranges of the date picker. By default the property is set to `false`.
`appearance.allowTouchZoom`	Sets a value which allows touch zoom in the viewer. By default the property is set to `true`.
`appearance.combineReportPages`	Sets a value which indicates that if a report contains several pages, then they will be combined in preview. By default the property is set to `false`.

Toolbar

Name	Description
`toolbar.visible`	Enables displaying the viewer toolbar. By default, the property is set to `true`.
`toolbar.displayMode`	Specifies the display mode of the toolbar of the viewer. It can take one of the following values of the `displayMode` enumeration: • `Simple` - all controls are located on the same control panel (default value); • `Separated` - the control panel is split into top and bottom panels.
`toolbar.backgroundColor`	Sets the color of the viewer toolbar. By default, the property is set to `Stimulsoft.System.Drawing.Co`

© 2003-2024 Stimulsoft

	lor.empty.
toolbar.borderColor	Sets the color of the borders of the Viewer toolbar. By default, the property is set to Stimulsoft.System.Drawing.Color.empty.
toolbar.fontColor	Sets the text color for the toolbar and the viewer menu. By default, the property is set to Stimulsoft.System.Drawing.Color.empty.
toolbar.fontFamily	Sets the font for the toolbar and the viewer menu. By default, the property is set to 'Arial'.
toolbar.alignment	Sets the alignment mode for the controls on the viewer toolbar. • Stimulsoft.Viewer.StiContentAlignment.Default – the alignment depends on the RightToLeft property (set by default). • Stimulsoft.Viewer.StiContentAlignment.Left – elements will be aligned left; • Stimulsoft.Viewer.StiContentAlignment.Center – elements will be centered; • Stimulsoft.Viewer.StiContentAlignment.Right – elements will be aligned right.
toolbar.showButtonCaptions	Enables text of the buttons on the toolbar of the viewer. By default, the property is set to true.
toolbar.showOpenButton	Enables displaying the **Open** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to true.

© 2003-2024 Stimulsoft

`toolbar.showPrintButton`	Enables showing the button - **Print** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showSaveButton`	Enables displaying the **Save** button on the toolbar of the viewer when viewing reports or dashboards.. By default, the property is set to `true`.
`toolbar.showSendEmailButton`	Enables showing the button - **Send Email** - on the toolbar of the viewer. By default, the property is set to `false`. Also, you should add the onEmailReport event handler.
`toolbar.showFindButton`	Sets a visibility of the **Find** button in the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showBookmarksButton`	Enables showing the button - **Bookmarks** - on the toolbar of the viewer. By default, the property is set to `true`. If the button is hidden, the bookmarks panel will not be displayed even if there are bookmarks in the report.
`toolbar.showParametersButton`	Enables showing the button - **Parameters** - on the toolbar of the viewer. By default, the property is set to `true`. If the button is hidden, the parameters panel will not be displayed even if there are parameters in the report.
`toolbar.showResourcesButton`	Enables showing the button - **Resources** - on the toolbar of the viewer. By default, the property is set to `true`. If the button is hidden, the resources panel will not be displayed even if there are resources in the report.

© 2003-2024 Stimulsoft

`toolbar.showEditorButton`	Enables showing the button - **Editor** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showFullScreenButton`	Enables displaying the **Full Screen** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to `true`.
`toolbar.showRefreshButton`	Sets a visibility of the **Refresh** button in the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showFirstPageButton`	Enables showing the button - **First Page** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showPreviousPageButton`	Enables showing the button - **Previous Page** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showCurrentPageControl`	Enables showing the current report page indicator. By default, the property is set to `true`.
`toolbar.showNextPageButton`	Enables showing the button - **Next Page** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showLastPageButton`	Enables showing the button - **Last Page** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showZoomButton`	Enables showing the Zoom button. By default, the property is set to `true`.
`toolbar.showViewModeButton`	Enables showing the button for selecting the display mode of report pages. By default, the property is set

© 2003-2024 Stimulsoft

	to `true`.
`toolbar.showDesignButton`	Enables displaying the **Design** button on the toolbar of the viewer when viewing reports or dashboards. By default, the property is set to `true`.
`toolbar.showAboutButton`	Enables showing the button - **About** - on the toolbar of the viewer. By default, the property is set to `true`.
`toolbar.showPinToolbarButton`	Sets a visibility of the **Pin** button in the toolbar of the viewer in mobile mode. By default, the property is set to `true`.
`toolbar.printDestination`	Sets the report printing mode. It can take one of the following values of the `StiPrintDestination` enumeration: • `Default` – a menu with a choice of printing modes will be displayed (default value); • `Pdf` – printing will be done to the PDF format; • `Direct` – printing will be done to the HTML format directly to the printer, the system print dialog will be displayed; • `PopupWindow` – printing will be done in the HTML format via the preview window of the report.
`toolbar.viewMode`	Sets the mode for displaying report pages. • `Stimulsoft.Viewer.StiWebViewMode.OnePage` – displays one page of the report selected in the toolbar of the viewer (the default value);

© 2003-2024 Stimulsoft

	• `Stimulsoft.Viewer.StiWebViewMode.Continuous` – displays all pages of the report; • `Stimulsoft.Viewer.StiWebViewMode.MultiplePages` – displays all report pages as a table.
`toolbar.zoom`	Sets the zoom for displaying report pages. The default setting is 100 percent. The values are from 10 to 500 percent. You can also set one of the following values: • `Stimulsoft.Viewer.StiZoomMode.PageWidth` – when the viewer runs, the zoom, necessary to display the report by the page width, will be set; • `Stimulsoft.Viewer.StiZoomMode.PageHeight` – when the viewer runs, the zoom, required to display the page height of the report, will be set.
`toolbar.menuAnimation`	Enables animation when the viewer menu shows/hides. By default, the property is set to `true`.
`toolbar.showMenuMode`	Sets the display mode of the viewer menu. It can take one of the following values of the `StiShowMenuMode` enumeration: • `Stimulsoft.Viewer.StiShowMenuMode.Click` – shows menu by mouse click (default value); • `Stimulsoft.Viewer.StiShowMenuMode.Hover` – shows menu by hovering the mouse cursor.

© 2003-2024 Stimulsoft

`toolbar.autoHide`	Sets a value which allows automatically hide the viewer toolbar in mobile mode. By default, the property is set to `false`.

Exports

Name	Description
`export.storeExportSettings`	Sets a value which allows store the export settings in the cookies. By default, the property is set to `true`.
`exports.showExportDialog`	Enables showing export options dialog box. If the property is set to `false`, the export will be done with the default settings. By default, the property is set to `true`.
`exports.showExportToDocument`	Enables the export menu item - **Document File**. By default, the property is set to `true`.
`exports.showExportToPdf`	Enables displaying the **Adobe PDF file** export menu item when viewing reports, and the **Adobe PDF** item when viewing dashboards. By default, the property is set to `true`.
`exports.showExportToXps`	Enables the export menu item - **XPS File**. By default, the property is set to `false`.
`exports.showExportToHtml`	Enables the export menu item - **HTML File**. By default, the property is set to `true`.
`exports.showExportToHtml5`	Enables the export menu item - **HTML5 File**. By default, the property is set to `true`.
`exports.showExportToWord2007`	Enables the export menu item - **Microsoft Word 2007/2010 File**. By

© 2003-2024 Stimulsoft

	default, the property is set to `true`.
`exports.showExportToExcel2007`	Enables displaying the **Microsoft Excel 2007/2010 File** export menu item when viewing reports, and the **Microsoft Excel** item when viewing dashboards. By default, the property is set to `true`.
`exports.showExportToCsv`	Enables the export menu item - **CSV**. By default, the property is set to `true`.
`exports.showExportToJson`	Enables the export menu item - **JSON**. By default, the property is set to `false`.
`exports.showExportToText`	Enables the export menu item - **Text File**. By default, the property is set to `true`.
`exports.showExportToOpenDocumentWriter`	Enables the export menu item - **OpenDocument Writer File**. By default, the property is set to `true`.
`exports.showExportToOpenDocumentCalc`	Enables the export menu item - **OpenDocument Calc File**. By default, the property is set to `true`.
`exports.showExportToPowerPoint`	Enables the export menu item - **Microsoft PowerPoint File**. By default, the property is set to `true`.
`exports.showExportToImageSvg`	Enables displaying the **Image** export menu item, with the ability to export the report to an SVG file. By default, the property is set to `true`.

Email

Name	Description
`email.showEmailDialog`	Enables displaying settings for

© 2003-2024 Stimulsoft

	sending the report via email. If the dialog box is disabled, the email will be sent with the settings set on the server side in the `onEmailReport` event. By default, the property is set to `true`.
`email.showExportDialog`	Enables displaying export options dialog box when sending email. If the property is set to `false`, the export will be done with the default settings. By default, the property is set to `true`.
`email.defaultEmailAddress`	Sets the default recipient email, i.e. the address to which the email with the attached report will be sent.
`email.defaultEmailSubject`	Sets the default email subject (header).
`email.defaultEmailMessage`	Sets the default email message (text).

9.4 HTML5 Designer

YouTube

Watch videos for working with JS HTML5 Designer. Subscribe to the Stimulsoft channel to watch new video lessons. Leave your questions and suggestions in the comments to the video.

Samples

See on GitHub examples of working with the JS HTML5 Designer component. All samples are separate projects, grouped into one solution.

The **HTML5 Designer** (**StiDesigner**) component is designed to create reports in the web browser. You do not need to install the .NET Framework, ActiveX components or

© 2003-2024 Stimulsoft

any special plug-ins on the client side. All that is needed is any modern Web browser.

With help of **HTML5 Designer** you can create, edit, save and preview reports on any computer with any operating system installed. Since the designer uses only HTML and JavaScript technologies, it can be run on devices where there is no Flash or Silverlight support - tablets, smartphones. Also, the designer supports the Touch interface, which is automatically enabled when using devices with a touch screen.

The **HTML5 Designer** component uses the **JavaScript** technology to perform all actions on reports, which allows you to get rid of reloading the entire page and speed up work.

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To use the **HTML5 Designer** in projects, you need to install the npm package of stimulsoft-reports-js:

```
npm install stimulsoft-reports-js
```

If this is not possible, you should add the following scripts to the project.

designer.html

```
...
<script src="scripts/stimulsoft.report.js"></script>
<script scr="scripts/stimulsoft.viewer.js"></script>
<script src="scripts/stimulsoft.designer.js"></script>

<!-- Stimulsoft Blockly editor for JS Designer -->
<script src="scripts/stimulsoft.blockly.editor.js"></script>
...
```

Install the npm package to use the HTML5 Viewer in a project to view reports and dashboards stimulsoft-dashboards-js:

```
npm install stimulsoft-dashboards-js
```

© 2003-2024 Stimulsoft

If this is not possible, you should add the following scripts to the project.

viewer.html

```
...
<script src="scripts/stimulsoft.report.js"></script>
<script src="scripts/stimulsoft.dashboards.js"></script>
<script scr="scripts/stimulsoft.viewer.js"></script>
<script src="scripts/stimulsoft.designer.js"></script>
...
```

Information

The script variants that you can use in your projects can be found in the Scripts of Reports.JS Package and Scripts of Dashboards.JS Package chapters.

9.4.1 How this Works

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **StiDesigner** component is developed using only HTML5 and JavaScript technology, and does not require a server for its work (it is necessary only for hosting project files). When you run the report viewer, the following actions occur:

› The JavaScript component adds designer interface code to the current HTML page;

› If a report object is assigned, the report will be uploaded to the designer page to edit it;

› Each action in the designer (for example, report preview, saving a report template, exporting a report, applying parameters, sorting and detailing a report) calls a specific JavaScript event in which you can perform the necessary manipulations with the report.

9.4.2 Editing Reports

Information

© 2003-2024 Stimulsoft

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **StiDesigner** component is a tool for creating and editing reports. To start working with the report designer, you should add the scripts and styles required for the component to the HTML page of the project.

designer.html

```
...
<script src="scripts/stimulsoft.reports.js" type="text/javascript"></
script>
<script src="scripts/stimulsoft.dashboards.js"></script>
<script scr="scripts/stimulsoft.viewer.js" type="text/javascript"></
script>
<script src="scripts/stimulsoft.designer.js" type="text/javascript"></
script>
...
```

To edit report event scripts using the Blockly tool in the designer, additionally you should add an appropriate script file, which contains a visual part of editor:

designer.html

```
...
<script src="scripts/stimulsoft.reports.js" type="text/javascript"></
script>
<script src="scripts/stimulsoft.dashboards.js"></script>
<script scr="scripts/stimulsoft.viewer.js" type="text/javascript"></
script>
<script src="scripts/stimulsoft.designer.js" type="text/javascript"></
script>
<script src="scripts/stimulsoft.blockly.editor.js" type="text/
javascript"></script>
...
```

Information

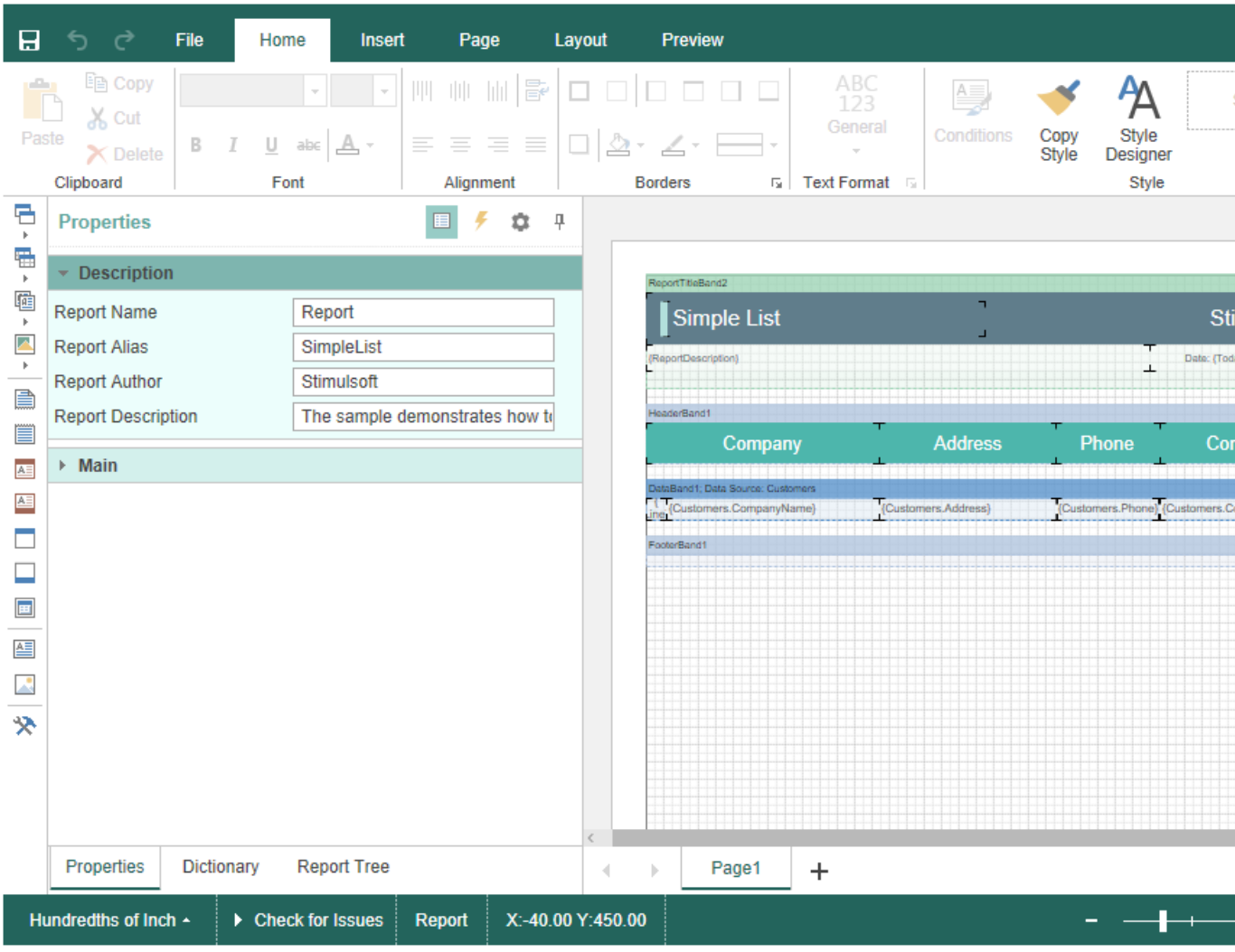
Also, you need to add the scripts and styles for the report viewer, because this component is used to preview reports on the corresponding designer tab.

© 2003-2024 Stimulsoft

Then you should add the JavaScript code of report loading to the HTML page and assign the resulting object to the designer.

designer.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
//report.loadFile("Dashboard.mrt");

var designer = new Stimulsoft.Designer.StiDesigner(null, "Designer", false);
designer.report = report;
...
```



© 2003-2024 Stimulsoft

You can create a **StiDesigner** object using the **Stimulsoft.Designer.StiDesigner()** constructor, which can take the following optional arguments as input:

- **options** – is a set of options that can be found in the **Stimulsoft.Designer.StiDesignerOptions** class. All options are divided into categories. A detailed description of the categories and options can be found in the chapter Designer Settings.
- **designerId** – designer ID, used when deploying a component as a DOM object, the default value is **StiDesigner**.
- **renderAfterCreate** – defines the designer location. If it is set to **true**, the designer will be displayed in the same place in the DOM tree where the code to create an object is located. If it is set to **false**, the designer will be located at the place where the **renderHtml()** method is called. For example, this can be the initialization of the designer in the page header.

designer.html

```
...
<script type="text/javascript">
   var designer = new Stimulsoft.Designer.StiDesigner(null, "StiDesigner",
   false);
</script>
...
```

And the subsequent output of the designer in the current DIV element.

designer.html

```
...
<div>Page content</div>
<div>
   <script type="text/javascript">
      // Render the report designer in this place
      designer.renderHtml();
   </script>
</div>
...
```

As an argument of the **renderHtml(id)** designer output method, it is allowed to specify the element identifier of the HTML page in which the designer should be displayed.

designer.html

© 2003-2024 Stimulsoft

```
...
<script type="text/javascript" >
   var designer = new Stimulsoft.Designer.StiDesigner(null, "StiDesigner",
   false);
   designer.renderHtml("content");
</script>
...
```

The specified element must be located on the HTML page on which the report designer is used.

designer.html

```
...
<div id="content"></div>
...
```

Two methods of the **StiReport** object are used to load a report – **loadFile()** and **load()**. They are used as follows:

- loadFile(filePath) – loads a report from the **MRT** file which path is specified in the filePath;
- load(str) – loads a report from the string str, that contains **XML** or **JSON**;
- load(data) – loads a report from the **array data** of the number[] type;
- load(xml) – loads a report from the XML of the **XMLDocument** type;
- load(json) – loads a report from the **JS** object.

For example, use the code below to load a report from file:

designer.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
...
```

The **MRT** format of Stimulsoft Reports is the **JSON** based description of reports. You can use **MRT** files, created in other Stimulsoft Reports designers in the **JSON** based description. Use the code below to save a report to a string:

designer.html

```
...
```

© 2003-2024 Stimulsoft

```
var report = new Stimulsoft.Report.StiReport();
var jsonString = report.saveToJsonString();
...
```

Use the code below to load a report from this string:

designer.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.load(jsonString);
...
```

9.4.3 Creating New Report

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

No action is required to run the designer with a new (empty) report. When the component is loaded, the new report will be created automatically. If you need, you can create a new report object and preload the data for it, or perform some other necessary actions.

designer.html

```
...
var report = new Stimulsoft.Report.StiReport();
//var  newDashboard = Stimulsoft.Report.StiReport.createNewDashboard();

report.reportName = "MyNewReport";
//newDashboard.reportName = "MyDashboard";
...
```

A special **DataSet** object is used for data storage, which has a set of methods for loading data from various formats. The **regData()** method is used to connect the data to the report. The arguments of the method indicate the prepared **DataSet** object.

© 2003-2024 Stimulsoft

designer.html

```
...
report.regData(dataSet);
...
```

Data is added to a special collection of the report object, and is used to build it. Data structure synchronization is used to display the structure of the registered data in the report dictionary. The **synchronize()** method is used for this.

designer.html

```
...
report.dictionary.synchronize();
...
```

You can also create a new report using the main menu of the designer. The **onCreateReport** event is used to load data for a new report, or to perform some other necessary actions. This event will be called when you create a new empty report, or when you create a report using the wizard.

designer.html

```
...
designer.onCreateReport = function (args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readJsonFile("Data/Demo.json");

   args.report.regData(dataSet.dataSetName, "", dataSet);
   args.report.dictionary.synchronize();
}
...
```

Also, you can pre-create a connection to the data source of the selected type, and add it to the collection of connections in the report template. This method is similar to creating a connection in the report dictionary using the designer interface.

designer.html

```
...
var report = new Stimulsoft.Report.StiReport();
//var  newDashboard = Stimulsoft.Report.StiReport.createNewDashboard();

var xmlDataBase = new Stimulsoft.Report.Dictionary.StiXmlDatabase("Demo",
```

© 2003-2024 Stimulsoft

```
"Demo.xsd", "Demo.xml");

report.loadFile("SimpleList.mrt");
//newDashboard.loadFile("Dashboard.mrt");
report.dictionary.databases.clear();
//newDashboard.dictionary.databases.clear();
report.dictionary.databases.add(xmlDataBase);
//newDashboard.dictionary.databases.add(xmlDataBase);
...
```

In this way, you can add the required number of data sources of different types.

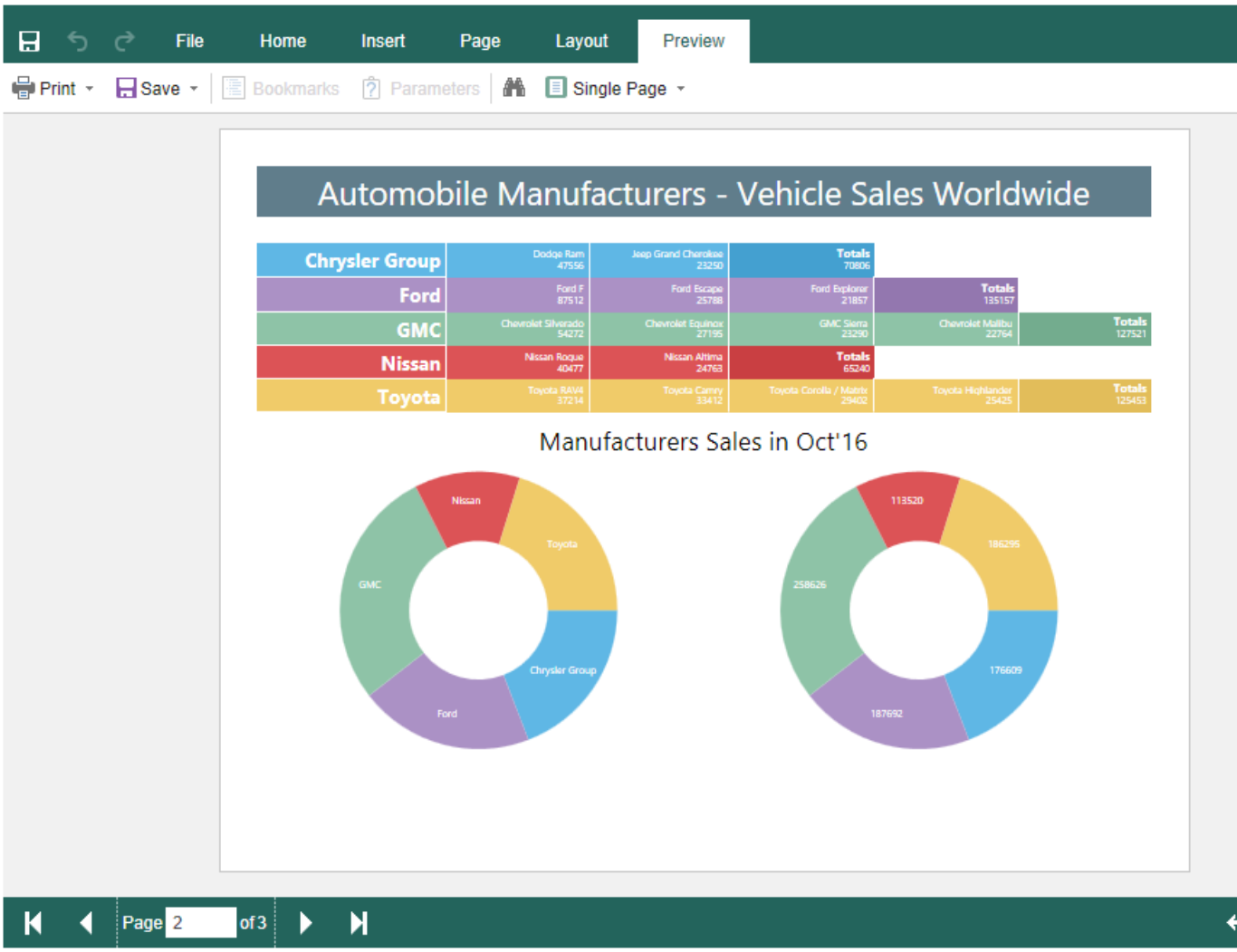
9.4.4 Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The **HTML5 Designer** component provides the ability to preview reports. To preview the report, just go to the appropriate tab in the designer window. The report template will be transferred to the server side, rendered and displayed in the embedded viewer.

© 2003-2024 Stimulsoft



Before previewing the report, it is possible to perform any necessary actions, for example, connect data for the report. To do this, you can use a special **onPreviewReport** event which will be called before previewing the report. The arguments of the event will contain a report to be previewed.

designer.html

```
...
designer.onPreviewReport = function (args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readJsonFile("Data/Demo.json");

   args.report.regData(dataSet.dataSetName, "", dataSet);
}
...
```

© 2003-2024 Stimulsoft

9.4.5 Additional Features of Preview

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

The preview window of the **HTML5 Designer** component has a fully functional interactive **HTML5 Viewer** that can print and export reports, supports working with report parameters, dynamic sorting, interactive reports, collapsing and etc. You do not need any additional settings for the report designer to use these features.

designer.html

```
...
designer.viewer.onBeginExportReport = function (event) {
   switch (event.format) {
      case Stimulsoft.Report.StiExportFormat.Html:
         event.settings.zoom = 2;  // Set zoom to 200%
         break;
   }
   console.log("exporting");
}
...
```

Information

If you do not need any of these additional options to preview the report (for example, exporting or printing a report), you can disable them using the appropriate properties of the **HTML5 Designer** component.

9.4.6 Saving Reports

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

© 2003-2024 Stimulsoft

The **HTML5 Designer** component has two ways of saving the report which are available in the main menu and in the main panel of the designer - **Save Report** and **Save As**. In turn, each of these ways has its own modes and settings.

Saving a report and dashboard on the server side

To save the edited report on the server side, you should set the **onSaveReport** special event which will be called when you select the **Save Report** menu item or click the **Save** button on the main panel of the designer.
An editable report will be passed in the arguments of the event. That report can be saved, for example, in a JSON string and then transferred to the server side.

designer.html

```
...
designer.onSaveReport = function (args) {
   args.preventDefault = false;
   var jsonReport = args.report.saveToJsonString();
}
...
```

Information

Additional details about the `onSaveReport` event handler argument are provided in the Designer Events.

By default, after saving the report, the designer continues working without displaying any messages. If necessary, after saving the report, it is possible to display a dialog box with an error or a text message. The special static function **showError()** is intended for this.

designer.html

```
...
designer.onSaveReport = function (args) {
   args.preventDefault = false;
   Stimulsoft.System.StiError.showError("Some message after saving", true);
}
```

© 2003-2024 Stimulsoft

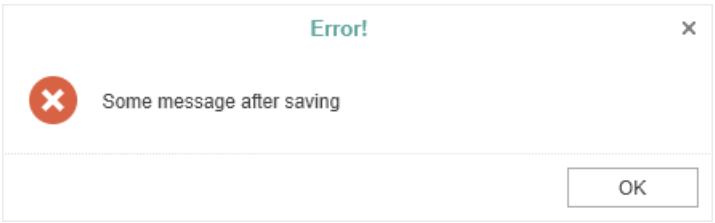
```
...
```

You can get a report name from the designer save dialog or an original report name.

designer.html

```
...
var designer = new Stimulsoft.Designer.StiDesigner(designerOptions,
"StiDesigner", false);
designer.renderHtml("content");

designer.onSaveReport = function (args) {

   //a flag to prevent further processing of the event
   args.preventDefault = false;

   //Report name from the designer save dialog
   var reportName = args.fileName;

   //Original report name from properties
   var reportName = args.report.reportName;
}
...
```

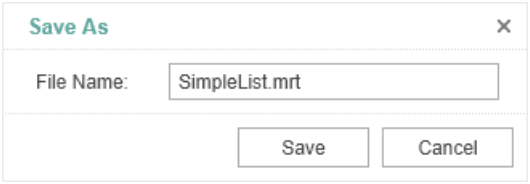
The function takes the error text and a flag as arguments that define the type of the window. The text can contain either a save error message or a warning, or any other message. If **true** is set as the second argument of the function, then an error window will be displayed. If **false** is set, the pop-up window with the error message will be displayed.



Saving reports and dashboards on the client side

No additional designer settings are required to save the edited report on the client side as a file. It is enough to click the **Save As** main menu item. The dialog box will be displayed. In this dialog you can change the name of the report file. The file will be saved to the local disk of the computer.

© 2003-2024 Stimulsoft



The **HTML5 Designer** component provides the ability to change the behavior of the specified save option. The special **onSaveReportAs** designer event is used for this. If you use this event, the report will be saved on the server side. It works similar to the **onSaveReport** event.

designer.html

```
...
designer.onSaveAsReport = function (args) {
   args.report.repotName = "Report";

   // Save the report template
   var jsonReport = args.report.saveToJsonString();
}
...
```

9.4.7 Using Themes

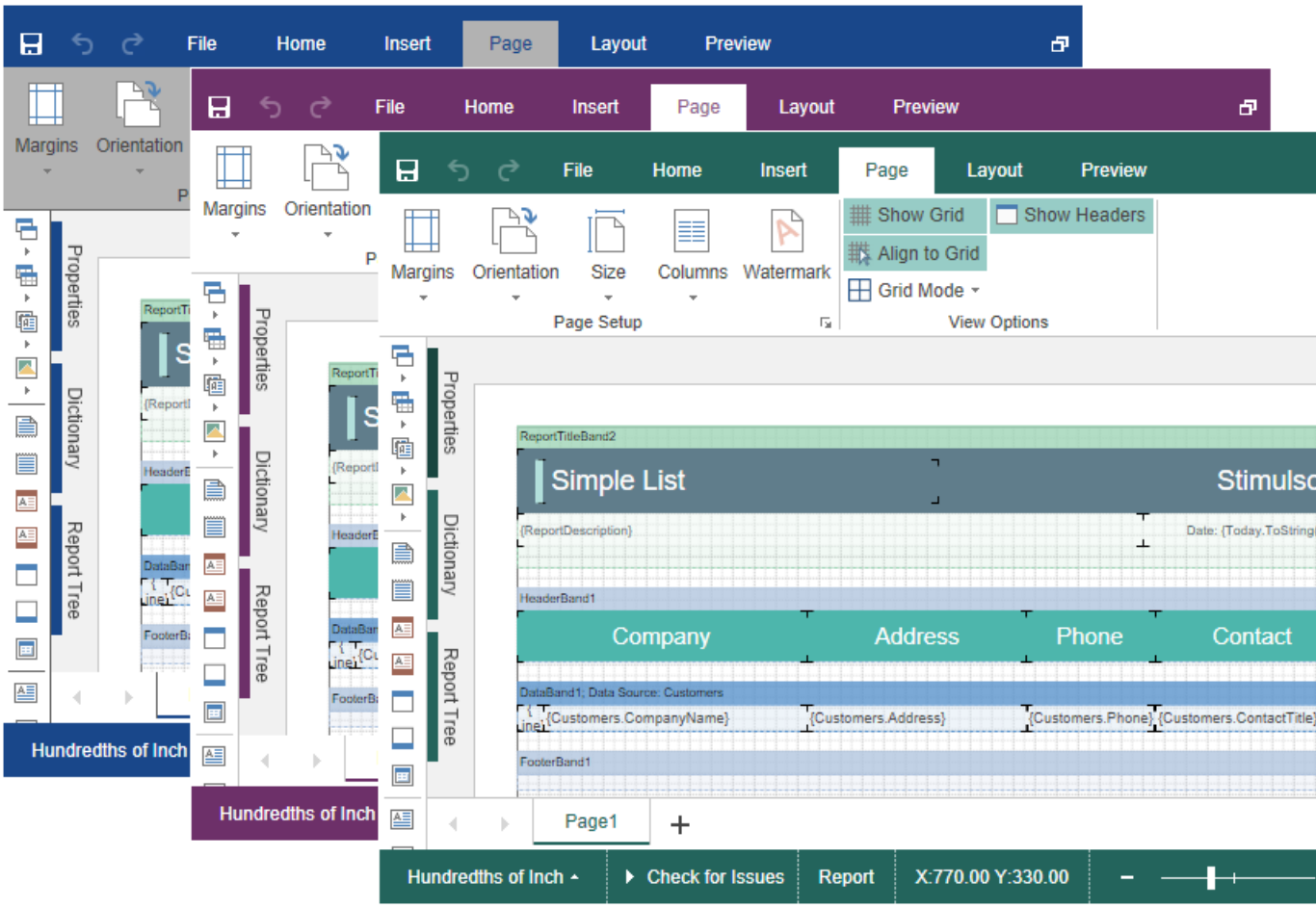
You can change the appearance of visual controls in the **HTML5 Designer** component. You can use the theme component option or the **setTheme()** method for this.

designer.html

```
...
var options = new Stimulsoft.Designer.StiDesignerOptions();
options.appearance.theme =
Stimulsoft.Designer.StiDesignerTheme.Office2022WhiteBlue;
...
designer.setTheme(Stimulsoft.Designer.StiDesignerTheme.Office2022WhiteBlue
);
...
```

There are currently **2 themes** available with different color accents. As a result, **more than 50** variants of the appearance are available. This allows you to customize the appearance of the designer for almost any design of the Web project.

© 2003-2024 Stimulsoft



9.4.8 Designer Events

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

To write interactive applications need to respond to changes in the application. The event system is used for this. It is represented in Stimulsoft Report Designer with the following events:

onPrepareVariables

Asynchronous event is called before filling in the variables in the report at the

© 2003-2024 Stimulsoft

beginning of the report rendering. The event occurs immediately after execution `onPrepareVariables` event of the `StiReport` object. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **PrepareVariables**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

designer.html

```
...
designer.onPrepareVariables = (args, callback) => {
   args.variables[0].value = "Replace value";
}
...
```

onBeginProcessData

Asynchronous event is called before requesting the data for the report. The event occurs immediately after execution `onBeginProcessData` event of the `StiReport` object. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **BeginProcessData**.

© 2003-2024 Stimulsoft

report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
command	a string ID for the current command. Can have values "TestConnection" and "ExecuteQuery" to call the command to test a connection and data acquisition respectively.
database	a string name of the current database.
connection	the name of the current connection to a data source, specified in report template.
headers	an array of the query headers.
withCredentials	an argument is used to provide the ability to set a cookies in the query.
connectionString	a connection string to the data source for a report.
dataSource	the name of the current data source, specified in the report template. It is set only for SQL data sources.
queryString	a string with the SQL query to the database for retrieving data. Used only with the command = "ExecuteQuery".
timeout	an argument is used to provide the ability to define the seconds of the query timeout.
parameters	a list of the query parameters.
escapeQueryParameters	a flag is used to provide the ability to use the escaped part of the request. By default, the value is set to **true**.
pathData	an argument is used to provide the ability to define a path to data file.
tryParseDateTime	a flag is trying to parse data as DateTime.

© 2003-2024 Stimulsoft

relationDirection	an argument is used to provide the ability to change the direction of the relation between data sources.
pathSchema	an argument is used to provide the ability to define a path to XSD files.
firstRowIsHeader	a flag is used to provide the ability to use a first row as data header in Excel data source.
collectionName	a string collection name of the OData data source.
separator	a string separator of the CSV data source.
dataType	an argument is used to provide the ability to set the data type of the GIS data source.
codePage	an argument is used to provide the ability to set a code page of the CSV or DBF data source.

designer.html

```
...
//Replace connection string
designer.onBeginProcessData = (args) => {
   if (args.database == "MySQL")
      args.connectionString = "new connection string";
}
...

//Add a some data
designer.onBeginProcessData = (args, callback) => {
   if (args.database == "MySQL"){
      args.preventDefault = true;
      var result = {
         success: true,
         rows: [
            ["value1", 1, false],
            ["value2", 1, true],
            ["value3", 2, false]
         ],
         columns: [
            "Column1_name",
            "Column2_name",
            "Column3_name"
         ],
         types:[
            "string",
            "int",
            "boolean"
         ]
      }
```

© 2003-2024 Stimulsoft

```
    // https://github.com/stimulsoft/DataAdapters.JS/
    callback(result);
  }
}
...
```

onEndProcessData

Is called after retrieving data for the report. The event occurs immediately after execution `onEndProcessData` event of the `StiReport` object. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **EndProcessData**.
report	a report object **StiReport**.
command	a string ID for the current command. Can have values "TestConnection" and "ExecuteQuery" to call the command to test a connection and data acquisition respectively.
dataSource	the name of the current data source, specified in the report template. It is set only for SQL data sources.
connection	the name of the current connection to a data source, specified in report template.
database	a string name of the current database.
result	a result dataset in a specific JSON format. It has two collections – columns and rows, with descriptions of columns and rows of data sources.

designer.html

```
...
```

© 2003-2024 Stimulsoft

```
designer.onEndProcessData = (args) => {
   if (args.command == "ExecuteQuery" && args.dataSource == "Categories")
      args.result.rows.push(rowData) ;
      // https://github.com/stimulsoft/DataAdapters.JS/
}
...
```

onCreateReport

Asynchronous event Is called after a new report is created. The event handler argument “event” it is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **CreateReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. The default value is **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
isWizardUsed	The flag indicates that a new report is created with the help of the master (the `true` value) or a blank report is created (the `false` value).

designer.html

```
...
designer.onCreateReport = function (args) {
   var report = args.report;

   var database = new
   Stimulsoft.Report.Dictionary.StiJsonDatabase("DemoData", "http://
   localhost/Demo.json");
   report.dictionary.databases.add(database);
   report.dictionary.synchronize();
}
...
```

© 2003-2024 Stimulsoft

onOpenReport

Asynchronous event is called before user click button for opening a report. The event handler argument event is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **OpenReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. The default value is **true**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

designer.html

```
...
//Call custom callback() function for loading template to designer
designer.onOpenReport = (args, callback) => {
   args.async = true;
   args.report = anotherReport;
   callback();
}
...
```

onOpenedReport

Asynchronous event is called before user click button for opening a report before it is assigned to the designer. The event handler argument event is an object with the next fields:

Name	Description

© 2003-2024 Stimulsoft

sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **OpenedReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. The default value is **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

designer.html

```
...
//Add image to report resource when it has been opening
designer.onOpenedReport = (args, callback) => {
   args.async = true;
   var xhr = new XMLHttpRequest();
   xhr.open('GET', "Url to image");
   xhr.onload = function () {
      var imageData = xhr.response;

      var resource = new
      Stimulsoft.Report.Dictionary.StiResource("ImageName");
      resource.content = imageData;
      args.report.dictionary.resources.add(resource);
      callback();
   };
   xhr.send();
}
...
```

onSaveReport

Asynchronous event is called before saving the report. The event handler argument event is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.

© 2003-2024 Stimulsoft

event	a string ID for the current event. By default, the value is set to **SaveReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. The default value is **true**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
fileName	a saving file name.
autoSave	a flag is used to use autosave mode. The default value is **false**.

designer.html

```
...
//Remove report resources before saving
designer.onSaveReport = (args, callback) => {
   var report = args.report.clone();
   report.dictionary.resources.clear();
   args.report = report;
}
...
```

onSaveAsReport

Asynchronous event is called before saving the report if user click **Save As** button. The event handler argument event is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **SaveAsReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. The default value is **false**.

© 2003-2024 Stimulsoft

async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
fileName	a saving file name.
autoSave	a flag is used to use autosave mode. The default value is **false**.

designer.html

```
...
//Stop and redefinition the save method
designer.onSaveAsReport = (args, callback) => {
   args.preventDefault = true;
   var jsonString = args.report.saveToJsonString();
   // save report
}
...
```

onPreviewReport

Asynchronous event is called when going to the report preview tab. The event handler argument event is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default it is **PreviewReport**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. The default value is **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
viewer	a viewer object **StiViewer**.

© 2003-2024 Stimulsoft

designer.html

```
...
designer.onPreviewReport = function (args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readJsonFile("Data/Demo.json");

   args.report.regData(dataSet.dataSetName, "", dataSet);
}
...
```

onExit

Is called before closing the designer. The event handler argument event is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default it is **Exit**.

designer.html

```
...
designer.onExit = function (args) {
   console.log(args.event);
}
...
```

9.4.9 Customizations in Designer

You can add custom controls to the report designer. In this chapter you may find several examples of customizing the report designer.

Adding a button to the **Home** tab.

designer.html

```
...
var designer = new Stimulsoft.Designer.StiDesigner(designerOptions,
"StiDesigner", false);
designer.renderHtml("content");
```

© 2003-2024 Stimulsoft

```
//Example add custom button to home toolbar panel
var homePanel = designer.jsObject.options.homePanel;

//Add buttons group to insert panel. Parameters of GroupBlock(groupName,
groupText) method.
var buttonsGroup = designer.jsObject.GroupBlock("buttonsGroup1",
"Group1");
var buttonsGroupTable = designer.jsObject.GroupBlockInnerTable();
buttonsGroup.container.appendChild(buttonsGroupTable);

//Add big button to buttons group. Parameters of BigButton(name,
groupName, caption, imageName, toolTip, arrow, styles) method.
var customBigButton = designer.jsObject.BigButton("customButton1", null,
"Custom Button", " ", "Tooltip for customButton1");
customBigButton.image.src = "https://www.stimulsoft.com/images/blocks/
ultimate-buttons/logo.png";

buttonsGroupTable.addCell(customBigButton).style.padding = "2px";

//customBigButton onclick event
customBigButton.action = function () {
   alert("customButton was pressed!");
}

//Add buttonsGroup and separator to customPanel
homePanel.firstChild.addCell(buttonsGroup);
homePanel.firstChild.addCell(designer.jsObject.GroupBlockSeparator());
...
```

Also, you can add a button to the top panel, near the **File** menu.

designer.html

```
...
var designer = new Stimulsoft.Designer.StiDesigner(designerOptions,
"StiDesigner", false);
designer.renderHtml("content");

var toolBarRow = designer.jsObject.options.toolBar.firstChild.tr[0];

var customButton = designer.jsObject.ToolButton("customButton1", "Custom
Button");

var buttonCell = document.createElement("td");
buttonCell.className = "stiDesignerToolButtonCell";
buttonCell.appendChild(customButton);

//For example insert button to position 3
toolBarRow.insertBefore(buttonCell, toolBarRow.childNodes[3]);

customButton.action = function () {
   alert("Button clicked!");
}
...
```

© 2003-2024 Stimulsoft

Below is an example of adding a custom panel in the report designer.

designer.html

```
...
var designer = new Stimulsoft.Designer.StiDesigner(designerOptions,
"StiDesigner", false);
designer.renderHtml("content");

var propertiesPanel = designer.jsObject.options.propertiesPanel;

var customPanel = document.createElement("div");
customPanel.jsObject = designer.jsObject;
customPanel.className = "stiDesignerPropertiesPanelInnerContent";
customPanel.style.top = "35px";
customPanel.style.display = "none";
customPanel.style.background = "gray";

propertiesPanel.containers["Custom"] = customPanel;
propertiesPanel.appendChild(customPanel);

var footerTable = propertiesPanel.footer.firstChild;

var tabButton = designer.jsObject.TabButton("CustomTabButton",
"PropertiesGridTabs", "Custom");
tabButton.style.margin = "0 0 0 3px";

tabButton.action = function () {
   if (!this.isSelected) propertiesPanel.showContainer("Custom");
}

designer.jsObject.loc.Panels.Custom = "Custom Panel Name";
...
```

9.4.10 Designer Settings

The HTML5 Designer is configured using properties that are located in the **Stimulsoft.Designer.StiDesignerOptions** class. All properties are split into groups. Some of the groups contain subgroups for ease of use. The following is an example of setting the properties of the viewer.

designer.html

```
...
<script type="text/javascript">
   var report = new Stimulsoft.Report.StiReport();
   report.loadFile("SimpleList.mrt");

   var options = new Stimulsoft.Designer.StiDesignerOptions();
   options.appearance.theme =
   Stimulsoft.Designer.StiDesignerTheme.Office2022WhiteBlue;
```

© 2003-2024 Stimulsoft

```
    options.viewerOptions.appearance.reportDisplayMode =
    Stimulsoft.Report.Export.StiHtmlExportMode.Auto;
    options.toolbar.showFileMenuExit = false;
    options.toolbar.showFileMenuOptions = false;
    options.bands.showChildBand = false;
    options.components.showPanel = false;
    options.appearance.showReportTree = false;
    options.appearance.showTooltips = false;

    var designer = new Stimulsoft.Designer.StiDesigner(options);
    designer.report = report;
</script>
...
```

Main settings (without groups)

Name	Description
`Width`	Sets the width of the component in “px” or “%”.
`Height`	Sets the height of the component in “px” or “%”.

Appearance

Name	Description
`theme`	Specifies the theme of the designer layout. The list of available themes can be found in the `StiDesignerTheme` enumeration. The default value is `Office2022WhiteBlue`.
`defaultUnit`	Sets the units for the size of the report and all its components. • `Stimulsoft.Report.StiReportUnitType.Centimeters` (default value);

© 2003-2024 Stimulsoft

	• `Stimulsoft.Report.StiReportUnitType.HundredthsOfInch;` • `Stimulsoft.Report.StiReportUnitType.Inches;` • `Stimulsoft.Report.StiReportUnitType.Millimeters.`
`zoom`	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the `StiZoomMode` enumeration: • `PageWidth` – when the designer runs, the zoom, necessary to display the report by the page width, will be set; • `PageHeight` – when the designer runs, the zoom, required to display the page height of the report, will be set.
`interfaceType`	Sets the type of interface used for the designer. It can take one of the following values: • `Stimulsoft.Designer.StiInterfaceType.Auto` – the interface type of the designer will be selected automatically depending on the device used (default value); • `Stimulsoft.Designer.StiInterfaceType.Mouse` – forced use of the interface to control the designer with the mouse device; • `Stimulsoft.Designer.StiInterfaceType.Touch` – forced

© 2003-2024 Stimulsoft

	use of the touch interface to control the designer via the touch screen (mobile devices), also in this mode, the interface elements are increased.
showAnimation	Enables animation for various elements of the designer interface. By default, the property is set to true.
showSaveDialog	Enables displaying the dialog to insert a report name when it is saved. The name of the report will be transferred in the parameters of the report designer. By default, the property is set to true.
showTooltips	Enables displaying tooltips for designer controls when the mouse hovers over. By default, the property is set to true.
showTooltipsHelp	Enables displaying links to online documentation in tooltips for designer controls. By default, the property is set to true.
showDialogsHelp	Sets a value which indicates that show or hide the help button in dialogs. By default, the property is set to true.
fullScreenMode	Sets the full screen display mode of the designer. If the property is set to true, the values of the width and height properties are ignored. By default, the property is set to false.
maximizeAfterCreating	Sets a value which indicates that the designer will be maximized after creation. By default, the property is set to false.

© 2003-2024 Stimulsoft

`showLocalization`	Sets a visibility of the localization control of the designer. By default, the property is set to `true`.
`allowChangeWindowTitle`	Allows using a title of the browser window to display the file name of the edited report. By default, the property is set to `true`.
`showPropertiesGrid`	Enables displaying the Property panel of the report designer. By default, the property is set to `true`.
`showReportTree`	Enables displaying the tree of report components. By default, the property is set to `true`.
`propertiesGridPosition`	Sets **Left** or **Right** position of the properties grid in the designer.
`showSystemFonts`	Sets a visibility of the system fonts in the fonts list. By default, the property is set to `true`.
`datePickerFirstDayOfWeek`	Sets the first day of week in the date picker. • `Stimulsoft.Designer.StiFirstDayOfWeek.Auto` - Sets **Monday** or **Sunday** as the first day depending on the browser culture (default value); • `Stimulsoft.Designer.StiFirstDayOfWeek.Monday` - Sets **Monday** as the first day of the week. • `Stimulsoft.Designer.StiFirstDayOfWeek.Sunday` - Sets **Sunday** as the first day of the week.
`wizardTypeRunningAfterLoad`	Calls the Report wizard after starting the report designer. It may have one

© 2003-2024 Stimulsoft

	of the following `StiWizardType` enumeration values: • `None` - runs the report designer without running the report wizard (default value); • `StandardReport` - runs the Standard wizard; • `MasterDetailReport` - runs the Master-Detail wizard; • `LabelReport` - runs the Label wizard; • `InvoicesReport` - runs the Invoice wizard; • `OrdersReport` - runs the Order wizard; • `QuotationReport` - runs the Quote wizard.

Toolbar

Name	Description
`visible`	It enables the display of toolbar in the report designer. The property has the `true` value by default.
`showPreviewButton`	It enables or disables the display of the **Preview** button in the designer toolbar. The property has the `true` value by default.
`showSaveButton`	It enables the display of the **Save** button in the designer toolbar. The property has the `true` value by default.
`showAboutButton`	It enables the display of the **About** button in the designer toolbar. The property has the `false` value by

© 2003-2024 Stimulsoft

	default.
`showFileMenu`	It enables the display of the main menu of the report designer. The property has the `true` value by default.
`showFileMenuNew`	It enables the display of the **New** main menu item. The property has the `true` value by default.
`showFileMenuOpen`	It enables the display of the **Open** main menu item. The property has the `true` value by default.
`showFileMenuSave`	It enables the display of the **Save** main menu item. The property has the `true` value by default.
`showFileMenuSaveAs`	It enables the display of the **Save as** main menu item. The property has the `true` value by default.
`showFileMenuClose`	It enables the display of the **Close** main menu item. The property has the `true` value by default.
`showFileMenuExit`	It enables the display of the **Exit** main menu item. The property has the `true` value by default.
`showFileMenuReportSetup`	It enables the display of the **Report Setup** main menu item. The property has the `true` value by default.
`showFileMenuOptions`	It enables the display of the **Options** main menu item. The property has the `true` value by default.
`showFileMenuInfo`	It enables the display of the **Info** main menu item. The property has the `true` value by default.
`showFileMenuAbout`	It enables the display of the **About** main menu item. The property has the `true` value by default.

© 2003-2024 Stimulsoft

showFileMenuNewReport	It enables or disables the display of the **New Page** main menu item. The property has the true value by default.
showFileMenuNewDashboard	It enables or disables the display of the **New Dashboard** main menu item. The property has the true value by default.
showSetupToolboxButton	It enables or disables the display of the report components side panel settings invoke button. The property has the true value by default.
showNewPageButton	It enables or disables the display of the **New Page** button in the toolbar. The property has the true value by default.
showNewDashboardButton	It enables or disables the display of the **New Dashboard** button in the toolbar. The property has the true value by default.

Bands

Name	Description
showReportTitleBand	It enables the display of the **Report Title** band in the designer components insert menu. The property has the true value by default.
showReportSummaryBand	It enables the display of the **Report Summary** band in the designer components insert menu. The property has the true value by default.
showPageHeaderBand	It enables the display of the **Page**

© 2003-2024 Stimulsoft

	Header band in the designer components insert menu. The property has the `true` value by default.
`showPageFooterBand`	It enables the display of the **Page Footer** band in the designer components insert menu. The property has the `true` value by default.
`showGroupHeaderBand`	It enables the display of the **Group Header** band in the designer components insert menu. The property has the `true` value by default.
`showGroupFooterBand`	It enables the display of the **Group Footer** band in the designer components insert menu. The property has the `true` value by default.
`showHeaderBand`	It enables the display of the **Header** band in the designer components insert menu. The property has the `true` value by default.
`showFooterBand`	It enables the display of the **Footer** band in the designer components insert menu. The property has the `true` value by default.
`showColumnHeaderBand`	It enables the display of the **Column Header** band in the designer components insert menu. The property has the `true` value by default.
`showColumnFooterBand`	It enables the display of the **Column Footer** band in the designer components insert menu. The property has the `true` value by

© 2003-2024 Stimulsoft

	default.
`showDataBand`	It enables the display of the **Data** band in the designer components insert menu. The property has the `true` value by default.
`showHierarchicalBand`	It enables the display of the **Hierarchical** band in the designer components insert menu. The property has the `true` value by default.
`showChildBand`	It enables the display of the **Child** band in the designer components insert menu. The property has the `true` value by default.
`showEmptyBand`	It enables the display of the **Empty** band in the designer components insert menu. The property has the `true` value by default.
`showOverlayBand`	It enables the display of the **Overlay** band in the designer components insert menu. The property has the `true` value by default.
`showTable`	It enables the display of the **Table** component in the designer components insert menu. The property has the `true` value by default.
`showTableOfContents`	It enables the display of the **Table of Contents** band in the designer components insert menu. The property has the `true` value by default.

Cross-Bands

© 2003-2024 Stimulsoft

Name	Description
showCrossTab	It enables the display of the **Cross-Tab** component in the designer components insert menu. The property has the `true` value by default.
showCrossGroupHeaderBand	It enables the display of the **Cross-Group Header** band in the designer components insert menu. The property has the `true` value by default.
showCrossGroupFooterBand	It enables the display of the **Cross-Group Footer** band in the designer components insert menu. The property has the `true` value by default.
showCrossHeaderBand	It enables the display of the **Cross-Header** band in the designer components insert menu. The property has the `true` value by default.
showCrossFooterBand	It enables the display of the **Cross-Footer** band in the designer components insert menu. The property has the `true` value by default.
showCrossDataBand	It enables the display of the **Cross-Data** band in the designer components insert menu. The property has the `true` value by default.

dashboardElements

Name	Description

© 2003-2024 Stimulsoft

`showTableElement`	It enables the display of the **Table** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showCardsElement`	It enables the display of the **Cards** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showChartElement`	It enables the display of the **Chart** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showGaugeElement`	It enables the display of the **Gauge** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showPivotTableElement`	It enables the display of the **Pivot** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showIndicatorElement`	It enables the display of the **Indicator** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showProgressElement`	It enables the display of the **Progress** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showRegionMapElement`	It enables the display of the **Region**

© 2003-2024 Stimulsoft

	Map dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showOnlineMapElement`	It enables the display of the **Online Map** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showImageElement`	It enables the display of the **Image** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showTextElement`	It enables the display of the **Text** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showPanelElement`	It enables the display of the **Panel** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showShapeElement`	It enables the display of the **Shape** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showButtonElement`	It enables the display of the **Button** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showListBoxElement`	It enables the display of the **List Box** dashboard element in the toolbox or

© 2003-2024 Stimulsoft

	the **Insert** tab in the designer. The property has the `true` value by default.
`showComboBoxElement`	It enables the display of the **Combo Box** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showTreeViewElement`	It enables the display of the **Tree View** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showTreeViewBoxElement`	It enables the display of the **Tree View Box** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showDatePickerElement`	It enables the display of the **Date Picker** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.

Components

Name	Description
`showText`	It enables the display of the **Text** component in the designer components insert menu. The property has the `true` value by default.
`showTextInCells`	It enables the display of the **Text in Cells** component in the designer components insert menu. The

© 2003-2024 Stimulsoft

	property has the `true` value by default.
`showRichText`	It enables the display of the **Rich Text** component in the designer components insert menu. The property has the `true` value by default.
`showImage`	It enables the display of the **Image** component in the designer components insert menu. The property has the `true` value by default.
`showBarCode`	It enables the display of the **Bar Code** component in the designer components insert menu. The property has the `true` value by default.
`showShape`	It enables the display of the **Shape** component in the designer components insert menu. The property has the `true` value by default.
`showPanel`	It enables the display of the **Panel** component in the designer components insert menu. The property has the `true` value by default.
`showClone`	It enables the display of the **Clone** component in the designer components insert menu. The property has the `true` value by default.
`showCheckBox`	It enables the display of the **Check Box** component in the designer components insert menu. The property has the `true` value by

© 2003-2024 Stimulsoft

	default.
`showSubReport`	It enables the display of the **Text** component in the designer components insert menu. The property has the `true` value by default.
`showZipCode`	It enables the display of the **Sub-Report** component in the designer components insert menu. The property has the `true` value by default.
`showChart`	It enables the display of the **Chart** component in the designer components insert menu. The property has the `true` value by default.
`showGauge`	It enables the display of the **Gauge** component in the designer components insert menu. The property has the `true` value by default.
`showSparkline`	It enables the display of the **Sparkline** component in the designer components insert menu. The property has the `true` value by default.
`showMathFormula`	It enables the display of the **Math Formula** component in the designer components insert menu. The property has the `true` value by default.
`showMap`	It enables the display of the **Map** component in the designer components insert menu. The property has the `true` value by default.

© 2003-2024 Stimulsoft

Dictionary

Name	Description
showAdaptersInNewConnectionForm	It enables the display of the **Object** category in the new connection creation window. The property has the true value by default.
showDictionary	It enables the display of the report dictionary. The property has the true value by default.
newReportDictionary	Allows you to use aliases in the data dictionary. It can take one of the specified below enumeration values: • StiNewReportDictionary.Auto - defines the mode of using aliases from a saved value in cookies (default value); • StiNewReportDictionary.True - sets the mode of using aliases in the data dictionary; • StiNewReportDictionary.False - disables the mode of using aliases in the data dictionary.
useAliases	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It can take one of the specified below enumeration values: • StiUseAliases.Auto - defines the mode to create or join the data dictionary from a saved value in cookies (default

© 2003-2024 Stimulsoft

	value); • `StiUseAliases.DictionaryNew` - sets the mode to create a new data dictionary when creating a new report; • `StiUseAliases.DictionaryMerge` - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
`showDictionaryContextMenuProperties`	Sets a visibility of the **Properties** item in the dictionary context menu. By default, the property is set to `true`.
`showDictionaryActions`	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.
`dataSourcesPermissions`	It sets available actions on report data sources. It can take one or several values from the `StiDesignerPermissions` enumeration.
`dataConnectionsPermissions`	It sets available actions on connections to report data. It can take one or several values from the `StiDesignerPermissions` enumeration.
`dataColumnsPermissions`	It sets available actions on report data columns. It can take one or several values from the `StiDesignerPermissions` enumeration.
`dataRelationsPermissions`	It sets available actions on report data connections. It can take one or several values from the `StiDesignerPermissions` enumeration.

© 2003-2024 Stimulsoft

businessObjectsPermissions	It sets available actions on report business objects. It can take one or several values from the StiDesignerPermissions enumeration.
variablesPermissions	It sets available actions on report variables. It can take one or several values from the StiDesignerPermissions enumeration.
resourcesPermissions	It sets available actions on sources in report dictionary. It can take one or several values from the StiDesignerPermissions enumeration.
dataTransformationsPermissions	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

In the table below you can see all available values for the StiDesignerPermissions enumeration. They can be set for report dictionary elements.

Name	Description
None	Disables any action on the item of the data dictionary.
Create	It allows you to create definite element of the dictionary.
Delete	It allows you to delete a definite element of the dictionary.
Modify	It allows you to edit a definite element of the dictionary.
View	It allows you to view a definite

© 2003-2024 Stimulsoft

	element of the dictionary.
`ModifyView`	It allows you to edit and view a definite element of the dictionary.
`All`	It allows you to make any actions on the dictionary element.

You can configure the built-in `StiViewer` component used to preview the report. To get access to all of its settings, you should use the `viewerOptions` property, which is an object of the viewer options. All its properties are described in the Viewer settings section.

9.5 Engine

This chapter contains a description of JS Engine functionality.

- Activation
- Connecting Data File
- Connecting SQL Databases
- Localization
- Loading and Saving Report
- Saving Rendered Report
- Getting Access to Pages
- Add custom functions

9.5.1 Activation

YouTube

Watch videos which show how to activate the JS components. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiDesigner** or **StiViewer** components.

index.html

© 2003-2024 Stimulsoft

```
...
function Start() {

   //Activation with using license code
   Stimulsoft.Base.StiLicense.Key = "Your activation code...";

   //Activation with using license file
   Stimulsoft.Base.StiLicense.loadFromFile("license.key");
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

Information

Please note that, due to security policies of the web browser, downloading the license file from the local storage will not be possible.

9.5.2 Connecting Data Files

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Connection parameters to data sources are usually stored in the report template itself. But if necessary, you can use other ways to connect data.

Data Sources from Files

The **DataSet** object is used to store data. It has methods for loading data from various file formats. The **regData()** method is used to connect data in the report, in the arguments of which the prepared **DataSet** object is specified.

The data can be loaded from XML files using XSD schema.

© 2003-2024 Stimulsoft

index.html

```
...
var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
dataSet.readXmlSchemaFile("Demo.xsd");
dataSet.readXmlFile("Demo.xml");

var report = new Stimulsoft.Report.StiReport();
report.regData(dataSet.dataSetName, "", dataSet);
report.dictionary.synchronize();
...
```

And from JSON files:

index.html

```
...
var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
dataSet.readJsonFile("Demo.json");

var report = new Stimulsoft.Report.StiReport();
report.regData(dataSet.dataSetName, "", dataSet);
report.dictionary.synchronize();
...
```

Additional methods for loading data from files

The **DataSet** object has a wide range of methods for loading data:

- readJsonFile(fileName) – loading a JSON file at the specified path;
- readJson(string) – loading JSON data as a string;
- readJson(data) – loading JSON data as a byte array;
- readJson(obj) – using a JavaScript object as data;

- readXmlFile(fileName) – loading an XML file at the specified path;
- readXml(string) – loading XML data as a string;
- readXml(data) – loading XML data as a byte array;

- readXmlSchemaFile(fileName) – loading an XSD file at the specified path;
- readXmlSchema(string) – loading XSD data as a string;
- readXmlSchema(data) – loading XML data as a byte array;

Information

© 2003-2024 Stimulsoft

Loading data schema is optional. If you want to use a data schema, you should add it before loading the XML data.

9.5.3 Connecting SQL Databases

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Samples

See examples on GitHub how to connect to databases.

Since pure JavaScript does not have built-in methods for working with remote databases, this functionality is implemented using server-side code. Therefore, Stimulsoft Reports.JS product contains server data adapters implemented using Node.js, PHP, .NET, .NET Core, Python, Java technologies. The database adapter is a software layer between the DBMS and the client script. The adapter connects to the DBMS and retrieves the necessary data, converting it into JSON. The script running on the server (using the adapter) provides for the exchange of JSON data between the client-side JavaScript application and the server side. To use this mechanism on the client side, you should specify the URL address of the host adapter, which processes requests to a required adapter

Links to examples with ready data adapters, implemented for various platforms: Node.js, PHP, .NET, .NET Core, Python, Java.

It`s easy to use an adapter. You should run an adapter and specify the its address:

index.html

```
...
StiOptions.WebServer.url = "http://localhost:9615";
```

© 2003-2024 Stimulsoft

```
...
```

When requesting data from SQL data sources, the Stimulsoft.Report.Engine sends a POST request to the URL, specified in the option:

index.html

```
...
StiOptions.WebServer.url = "https://localhost/handler.php";
...
```

A JSON object with parameters is passed in the body of the request that use the following structure:

- **command**: two variants are possible - "TestConnection" and "ExecuteQuery";
- **connectionString**: database connection string;
- **queryString**: query string;
- **database**: database type;
- **timeout**: the time of request waiting, specified in the data source;
- **parameters**: an array of parameters as a JSON object {name, value};
- **escapeQueryParameters**: a flag of parameters shielding before requesting.

In response, the Stimulsoft.Report.Engine expects a JSON object with data in the form of the following structure:

- **success**: a flag of successful command execution;
- **notice**: if the flag of command execution has the **false** value, this parameter will contain an error description;
- **rows**: strings array, each element is the array of values, the index is the column number;
- **columns**: an array of column names, index is the column number;
- **types**: an array of column types, the index is the column number. It can take the values "string", "number", "int", "boolean", "array", "datetime".

Request and response sample:

index.html

```
...
request = {
```

© 2003-2024 Stimulsoft

```
   command: "ExecuteQuery",
   connectionString: "Server=myServerAddress;Database=myDataBase;User
   Id=myUsername;Password=myPassword;",
   queryString: "select * from table1",
   database: "MS SQL"
}

response = {
   success: true,

   rows: [
      ["value1", 1, false],
      ["value2", 1, true]
      ["value3", 2, false]
   ],
   columns: [
      "Column1_name",
      "Column2_name",
      "Column3_name"
   ],
   types:[
      "string",
      "int",
      "boolean"
   ]
}
...
```

Custom Data Base

Also, you may register custom data base. To do it you should invoke the following option:

index.html

```
...
Stimulsoft.Report.Dictionary.StiCustomDatabase.registerCustomDatabase(opti
ons);
...
```

The options are the set of properties and the process() function, which will be invoked when requesting data:

- **serviceName**: adapter name which will be displayed in the designer when creating a new connection
- **designerCategory**: the category name to which the adapter will be added in the **New Data Source** menu. By default, the adapter can be found in the `Favorites` category, but you can change this by setting the option to one of the following values: `"Files"`, `"SQL"`, `"NoSQL"`, `"Azure"`, `"Google"`, `"OnlineServices"`,

© 2003-2024 Stimulsoft

`"REST"`.

- **sampleConnectionString**: the sample of a connection string that is inserted in the form of setting up a new connection
- **process**: the function, which will be invoked to prepare and transmit data to the Stimulsoft.Report.Engine

Two arguments are transmitted to the input of the **process()** function: **command** and **callback**.

The **command** argument is the JSON object, where the Stimulsoft.Report.Engine will transfer the following parameters:

- **command**: action, which is being invoked at the moment. Possible values: "**TestConnection**": test the database connection from the new connection creation form "**RetrieveSchema**": retrieving data schema is needed to optimize a request and not only to transfer necessary data set. It is invokes after connection creation "**RetrieveData**": data request.
- **connectionString**: connection string;
- **queryString**: query string;
- **database**: database type;
- **timeout**: the time of request waiting, specified in the data source.

The **callback** argument is the function, which should be invoked to transmit prepared data to the Stimulsoft.Report.Engine. As the callback argument to the functions you must pass a JSON object with the following parameters:

- **success**: the flag of successful command execution;
- **notice**: if the flag of command execution has the **false** value, this parameter should contain an error description;
- **rows**: strings array, each element is the array from values, the index is the column number;
- **columns**: columns name array, the index is the column number;
- **types**: the object where field name is column name and the value is the type of the column {Column_Name : "string"}. The type can take the following values "string", "number", "int", "boolean", "array", "datetime". If the **columns** array will be transmitted, you will be able to transmit types array to the **types**, the index should be column number. It doesn't work for the "**RetrieveSchema**".

If the command = "**RetrieveSchema**", then in addition types you should transmit table names to the **types**.

© 2003-2024 Stimulsoft

The sample of a request and response when receiving a schema:

index.html

```
...
request = {
   command: "RetrieveSchema"
}

response = {
   success: true,

   types:{
      Table1: {
         Column1: "string",
         Column2: "number"
      },
      Table2: {
         Column1: "string"
      }
   }
}
...
```

The sample of a request and response when getting data:

index.html

```
...
request = {
   command: "RetrieveData",
   queryString: "Table1"
}
response = {
   success: true,

   rows: [
      ["value1", 1],
      ["value2", 1]
      ["value3", 2]
   ],
   columns:[
      "Column1",
      "Column2"
   ],
   types:[
      "string",
      "number"
   ]
}
...
```

© 2003-2024 Stimulsoft

The example of adapter registration.

Query Timeout

Also, for SQL data sources used in the report, you can specify the **Query Timeout** in seconds. The value of this property is stored in the report template for each SQL connection separately.

Below is an example of code that you may use to change the connection string for MS SQL, adjust the query, set the query timeout for the already created connection, and data sources in the report.

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("Report.mrt");
report.dictionary.databases.getByName("Connection").connectionString =
"Data Source=server;Integrated Security=True;Initial Catalog=DataBase";
report.dictionary.dataSources.getByName("DataSourceName").sqlCommand =
"select * from Table where Column = 100";
report.dictionary.dataSources.getByName("DataSourceName").commandTimeout =
1000;
...
```

Information

The address to the data adapter must be set before the creation code of the component or the report object, since the value of this option must be known to the report engine before it is initialized.

The following SQL data sources are currently supported - MySQL, MS SQL, PostgreSQL, Firebird, and Oracle. The data adapters are open source and can be modified the way you need.

OData storages

You can also use data for designing reports and dashboards obtained from OData storages. In this case, you can do the authorization using a username, user password, or using a token. Authorization parameters are specified in the connection string to

© 2003-2024 Stimulsoft

the OData storage using the ";" separator.

viewer.html

```
...
//Authorization using a user account
var oDataDatabase = new
Stimulsoft.Report.Dictionary.StiODataDatabase("OData", "OData", "https://
services.odata.org/V4/Northwind/
Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Password;Cli
ent_Id=Your Client ID", false, null);

//Authorization using a user token
var oDataDatabase = new
Stimulsoft.Report.Dictionary.StiODataDatabase("OData", "OData", "https://
services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your token",
false, null);

report.dictionary.databases.add(oDataDatabase);
report.dictionary.synchronize();

//Query with data filter
var productsDataSource =
report.dictionary.dataSources.getByName("Products");
if (productsDataSource != null) productsDataSource.sqlCommand =
"Products?$filter=ProductID eq 2";
...
```

Also, you can specify user HTTP headers for data sources. It can be done in the `onBeginProcessData` event handler. A complete sample is available on our website:

index.html

```
...
// In `onBeginProcessData` event handler add custom HTTP headers
report.onBeginProcessData = function (args) {
   if (
      args.database === "JSON" &&
      args.command === "GetData" &&
      args.pathData && args.pathData.include("/reports/ProtectedDemo.json")
   ) {
      // Add custom header to pass through backend server protection
      args.headers.push({key: "X-Auth-Token", value: "*YOUR TOKEN*"});
   }
};
...
```

© 2003-2024 Stimulsoft

9.5.4 Localization

The **HTML5 Viewer** and **HTML5 Designer** components support full localization of the user interface. The special static method - **addLocalizationFile()** - is used to localize the report viewer interface to the required language. As the arguments of the method, you should specify the path to the localization XML file, and specify whether the localization will automatically load together with the component.

The specified function adds localization to the designer menu:

index.html

```
...
Stimulsoft.Base.Localization.StiLocalization.addLocalizationFile("../
localization/de.xml", true);
...
```

The specified function adds localization to the designer menu, but doesn`t load it automatically:

index.html

```
...
Stimulsoft.Base.Localization.StiLocalization.addLocalizationFile("../
localization/de.xml", false, "Deutsch");
...
```

After the method is called, a link to a localization file will be added to the internal collection. The file will be loaded the first time you select the specified localization in the designer menu. It saves memory and time of components run. In this variant, you should specify localization name for displaying in the designer menu as the third argument. After localization loaded this value will be taken from a localization file.

The specified function adds localization to the designer menu and sets it by default:

index.html

```
...
Stimulsoft.Base.Localization.StiLocalization.setLocalizationFile("../
localization/de.xml");
...
```

© 2003-2024 Stimulsoft

When each the designer loading, this localization will be selected even if earlier another localization was selected.

Information

For the **HTML5 Designer**, when creating a project with pure JavaScript, the necessary localization files should be added using the above code. If the project uses the Node.js technology, then it is enough to specify a folder with localization files. In this case all the files in it will be added automatically:

```
Stimulsoft.System.NodeJs.localizationPath = "locales";
```

9.5.5 Loading and Saving Report

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

Two methods of the **StiReport** object are used to load a report – **loadFile()** and **load()**. They are used as follows:

- loadFile(filePath) – loads a report from the **MRT** file which path is specified in the filePath;
- load(str) – loads a report from the string str, that contains **XML** or **JSON**;
- load(data) – loads a report from the **array data** of the number[] type;
- load(xml) – loads a report from the XML of the **XMLDocument** type;
- load(json) – loads a report from the **JS** object.

For example, use the code below to load a report from file:

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
...
```

© 2003-2024 Stimulsoft

The **MRT** format of Stimulsoft Reports is the **JSON** based description of reports. You can use **MRT** files, created in other Stimulsoft Reports designers in the **JSON** based description. Use the code below to save a report to a string:

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
var jsonString = report.saveToJsonString();
...
```

Use the code below to load a report from this string:

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.load(jsonString);
...
```

9.5.6 Saving Rendered Report

Rendered report can be saved into a **JSON** document for later viewing (data stored within a document):

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.renderAsync(function(){
   var jsonString = report.saveDocumentToJsonString();
});
...
```

9.5.7 Getting Access to Pages

Getting access to pages of a report

The report has a collection of pages, which can be accessed by the following way:

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");

for (var index = 0; index < report.pages.count; index++) {
   alert(report.pages.getByIndex(index).name);
```

© 2003-2024 Stimulsoft

```
}
...
```

Getting access to pages of a rendered report

After rendering the report, a collection of pages is created. It can be accessed by the following way:

index.html

```
...
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
      K      ^     E         EF
  for (var index = 0; index < report.renderedPages.count; index++) {
    alert(report.renderedPages.getByIndex(index).name);
  }
 FX
...
```

9.5.8 Report Events

Information

Since dashboards and reports use the same unified template format - MRT, methods for loading the template and working with data, the word “report” will be used in the documentation text.

In order to write interactive applications it is needed to respond to changes in the application. The event system which is represented in **StiReport** object can be used for this. The following events exist:

onPrepareVariables

Asynchronous event is called before filling in the variables in the report at the beginning of the report rendering. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.

© 2003-2024 Stimulsoft

event	a string ID for the current event. By default, the value is set to **PrepareVariables**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.

viewer.html

```
...
report.onPrepareVariables = (args, callback) => {
   args.variables[0].value = "Replace value";
}
...
```

onBeginProcessData

Asynchronous event is called before requesting the data for the report. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **BeginProcessData**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
command	a string ID for the current command. Can have

© 2003-2024 Stimulsoft

	values "TestConnection" and "ExecuteQuery" to call the command to test a connection and data acquisition respectively.
database	a string name of the current database.
connection	the name of the current connection to a data source, specified in report template.
headers	an array of the query headers.
withCredentials	an argument is used to provide the ability to set a cookies in the query.
connectionString	a connection string to the data source for a report.
dataSource	the name of the current data source, specified in the report template. It is set only for SQL data sources.
queryString	a string with the SQL query to the database for retrieving data. Used only with the command = "ExecuteQuery".
timeout	an argument is used to provide the ability to define the seconds of the query timeout.
parameters	a list of the query parameters.
escapeQueryParameters	a flag is used to provide the ability to use the escaped part of the request. By default, the value is set to **true**.
pathData	an argument is used to provide the ability to define a path to data file.
tryParseDateTime	a flag is trying to parse data as DateTime.
relationDirection	an argument is used to provide the ability to change the direction of the relation between data sources.
pathSchema	an argument is used to provide the ability to define a path to XSD files.
firstRowIsHeader	a flag is used to provide the ability to use a first row as data header in Excel data source.

© 2003-2024 Stimulsoft

collectionName	a string collection name of the OData data source.
separator	a string separator of the CSV data source.
dataType	an argument is used to provide the ability to set the data type of the GIS data source.
codePage	an argument is used to provide the ability to set a code page of the CSV or DBF data source.

viewer.html

```
...
//Replace connection string
report.onBeginProcessData = (args) => {
   if (args.database == "MySQL")
      args.connectionString = "new connection string";
}
...

//Add a some data
report.onBeginProcessData = (args, callback) => {
   if (args.database == "MySQL"){
      args.preventDefault = true;
      var result = {
         success: true,
         rows: [
            ["value1", 1, false],
            ["value2", 1, true],
            ["value3", 2, false]
         ],
         columns: [
            "Column1_name",
            "Column2_name",
            "Column3_name"
         ],
         types:[
            "string",
            "int",
            "boolean"
         ]
      }

      // https://github.com/stimulsoft/DataAdapters.JS/
      callback(result);
   }
}
...
```

onEndProcessData

Is called after retrieving data for the report. The event handler argument “event” is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	a string ID for the current event. By default, the value is set to **EndProcessData**.
report	a report object **StiReport**.
command	a string ID for the current command. Can have values "TestConnection" and "ExecuteQuery" to call the command to test a connection and data acquisition respectively.
dataSource	the name of the current data source, specified in the report template. It is set only for SQL data sources.
connection	the name of the current connection to a data source, specified in report template.
database	a string name of the current database.
result	a result dataset in a specific JSON format. It has two collections – columns and rows, with descriptions of columns and rows of data sources.

viewer.html

```
...
report.onEndProcessData = (args) => {
   if (args.command == "ExecuteQuery" && args.dataSource == "Categories")
      args.result.rows.push(rowData) ;
      // https://github.com/stimulsoft/DataAdapters.JS/
}
...
```

onBeginRender

Is called at the beginning of the report rendering. This is not relevant for the dashboards.

onRendering

Is called at the report is rendering when each report page is created. This is not relevant for the dashboards.

onEndRender

Is called at the ending of the report rendering. This is not relevant for the dashboards.

onExportingRender

Is called before a report export is started.

onExportedRender

Is called after a report export is ended.

onPrinting

Is called when `report.print()` or `report.printToPdf()` method is invoked. This is not relevant for the dashboards.

viewer.html

```
...
//Remove image before report printing
report.onPrintReport = () => {
   var page = report.renderedPages.getByIndex(0);
   var image = page.components.getByName("Image1");
   if (image)
      page.components.remove(image);
}
...
```

© 2003-2024 Stimulsoft

onPrinted

Asynchronous event is called when `report.print()` or `report.printToPdf()` method is invoked after a report is exported to Html or PDF file (depends on method type). This is not relevant for the dashboards. The event handler argument "event" is an object with the next fields:

Name	Description
sender	The identifier of the component, which initiated this event.
event	string ID for the current event. By default, the value is set to **Printed**.
report	a report object **StiReport**.
preventDefault	a flag to prevent further processing of the event. By default, the value is set to **false**.
async	a flag is used to provide the ability to stop the execution of the event before the callback function is executed. By default, the value is set to **false**.
data	export data to the string or byte array representation.

viewer.html

```
...
//Stop to report print and define a custom print method
report.onPrintReport = (args) => {
   args.preventDefault = true;
   var printData = args.data;
   myPrintingMethod(printData);
}
...
```

onRefreshing

Is called after a report is rendered if **report.refreshTime** property is set more than zero. Also, this event is called when **Refresh** button is clicked in the viewer.

© 2003-2024 Stimulsoft

9.5.9 Add custom functions

You can add a custom function to the Dictionary in the report designer when you integrate it into your application. After adding the custom function, you can use this in creating reports and dashboards. Below is the example of adding a function for calculating the sum total.

index.html

```
...
var myFunc = function (value) {
   if (!Stimulsoft.Data.Extensions.ListExt.isList(value))
   return Stimulsoft.Base.Helpers.StiValueHelper.tryToNumber(value);

   return
   Stimulsoft.Data.Functions.Funcs.skipNulls(Stimulsoft.Data.Extensions.Lis
   tExt.toList(value))
   .tryCastToNumber()
   .sum();
};

Stimulsoft.Report.Dictionary.StiFunctions.addFunction("MyCategory",
"MySum", "MySum", "MySum", "", Number, "Return Description", [Object],
["value"], ["Descriptions"], myFunc);
...
```

10 Reports and Dashboards for PHP

The Stimulsoft Reports.PHP and Stimulsoft Dashboards.PHP products are JavaScript and PHP scripts combination, and all the necessary functionality for their interaction. The JavaScript that renders and exports a report and displays dashboards on the client-side. The PHP scripts that connect data sources and control downloading and saving reports and dashboards work on the server side. Also, it allows you to control report variables and parameters of requests.

So as a report is rendered on the client side, it imposes certain restrictions on the automation of rendering and exporting reports. However, it allows you not to worry about server power and saves money on maintenance.

The product contains the report generator and the data analysis engine, components of the viewer and the designer reports, and the set of PHP scripts for work with various SQL data sources.

© 2003-2024 Stimulsoft

10.1 Engine

Stimulsoft Reports.PHP allows you to load, render and export a report in various formats without deploying the viewer and the designer. It allows you not to load capacious scripts on the server side.

A report is rendered and exported on the client side using JavaScript. The server side of PHP contains all you need to work with report files and connections with various SQL data sources. A client is connected with the server using AJAX requests, which send and receive JSON data in a specific format. To make the use of the product convenient, special events and functions are developed both on the JavaScript client side and the PHP service side.

Dashboards
The Stimulsoft Dashboards.PHP analytical panels allow you to load, analyze data, and export a dashboard in various formats without deploying the viewer and the designer.

All the data is analyzed on the client side using JavaScript except for certain SQL operations. The PHP server-side contains all you need to work with dashboard files and connect with various SQL data sources. A user connects with the server using AJAX requests, which send and get JSON data in a certain format. To make the use of the product convenient, special events and functions are developed on both the JavaScript client and the PHP server client.

- Deployment
- Optimization of script loading
- Activation
- Loading and Saving Report
- PHP Events Handler
- Connecting Data Files
- Connecting SQL Data Adapters
- Work with Report Variables
- Printing Report from Code
- Export Report from Code
- Engine Events

10.1.1 Deployment

Samples

© 2003-2024 Stimulsoft

You can find a full code of on the GitHub.

To use the product, you should download the ZIP archive of the product from the Downloads page of our site, unpack it, and copy the contents of the **/PHP** folder to your Web server. This folder is a Web project that contains all the necessary files and resources for the product and samples for working with the viewer and designer.

To install the product in an existing project, you should copy the /vendor folder from the **/PHP** folder to the root directory of the project or use the Composer dependency manager by executing the following console command:

console

```
composer require stimulsoft/reports-php
```

When working with the product, in most cases, it is enough to use only PHP code, which ensures the operation of all the main features. For a more detailed configuration of the product and the use of all features, you must use JavaScript code. Next, two options for using certain features will be described - using **PHP** and **JavaScript** functions. Depending on your needs, you may use either the first or the second option.

To use the reporting tool in a Web project, you should add scripts to the `<head>` block on the PHP page. You can do this using the `StiJavaScript` **PHP** class. In the constructor, you need to specify the type of component to deploy:

index.php

```
<head>
   <?php
      $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
      \StiComponentType::Report);
      $js->renderHtml();
   ?>
</head>
```

© 2003-2024 Stimulsoft

Also, you may connect all the necessary scripts using the `<script>` block. All product scripts are located in the `vendor/stimulsoft/reports-php/public/scripts/` directory:

index.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.js" type="text/javascript"></script>
</head>
```

For dashboards, you need to connect the following package:

console

```
composer require stimulsoft/dashboards-php
```

In the case of using **PHP** code, all necessary scripts will be enabled automatically. No code change is required. If you use **JavaScript** code, you will need to add the appropriate script file that contains everything you need to work with analytical panels:

index.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/dashboards-php/scripts/
   stimulsoft.dashboards.js" type="text/javascript"></script>
</head>
```

Then, you can use **PHP** classes and functions to work with reports, which must be placed inside the `<script>` block. For example, you want to load a report from a file and render it:

© 2003-2024 Stimulsoft

index.php

```
<script type="text/javascript">
   <?php
      $report = new \Stimulsoft\Report\StiReport();
      $report->loadFile('reports/SimpleList.mrt');
      $report->render('onAfterRender');
      $report->renderHtml();
   ?>

   function onAfterRender() {
      alert('Done!');
   }
</script>
```

Information

Stimulsoft Reports.PHP and **Stimulsoft Dashboards.PHP** do not have a native PHP report engine; report rendering and export are performed on the client side using JavaScript code. Therefore, when using **PHP code** to work with components, it is necessary to call the `renderHtml()` function, which will add the appropriate **JavaScript code** to the Web page to perform all the required actions.

If necessary, the same actions can be performed using **JavaScript** functions:

index.php

```
<script type="text/javascript">
   var report = new Stimulsoft.Report.StiReport();
   report.loadFile("reports/SimpleList.mrt");
   report.renderAsync(function() {
      alert("Done!");
   })
</script>
```

Managing the URL for loading JavaScript reporting tool files

By default, all product **.js** files are loaded via URLs relative to the location of the current PHP script. In some cases, this behavior needs to be changed. For this purpose, there are two options for the `StiJavaScript` class. To use an absolute

© 2003-2024 Stimulsoft

path for loading all product scripts, you must set the `useRelativeUrls` option to `true`:

index.php

```
<?php
    $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
\StiComponentType::Report);
    $js->useRelativeUrls = true;
?>
```

To adjust the relative path, the `relativePath` option is provided, for which you need to set a string value that will be used when generating the script loading URL. In this case, the `useRelativeUrls` option must be disabled (the default value is used).

index.php

```
<?php
    $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
\StiComponentType::Report);
    $js->relativePath = '../../';
?>
```

Various alternatives of deployment and optimization are considered in the Optimization of script loading chapter.

10.1.2 Optimization of Scripts Loading

Scripts are quite large due to imposing product functionality. When either first loading a web application or disabled browser caching, the download may take some time, especially if the internet connection is low. We offer two variants of solution this problem: use packed scripts or use a partial functionality and download only what is required.

Packed Scripts

Packed scripts have the same structure as basic scripts, but they have the `*.pack.js` ending in the file name. Such scripts contain a block of packed data as a

© 2003-2024 Stimulsoft

JavaScript variable and a compact unpacker. When loading all scripts, an unpacker automatically unpacks all loaded data and runs a prepared script for execution. The unpacking takes some time, but under certain conditions, for example, when the internet connection is slow time is much more less, than basic scripts load speed.

To use packed scripts you should set the `$js->packed` to `true`. For example:

index.php

```
<head>
   <?php
      $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
      \StiComponentType::Report);
      $js->usePacked = true;
      $js->renderHtml();
   ?>
</head>
```

When using the **JavaScript** code, you should add the ending .pack.js in all Stimulsoft scripts used, for example:

index.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.pack.js" type="text/javascript"></script>
</head>
```

Partial loading of scripts

The `stimulsoft.reports.js` script contains all functionality of rendering and exporting reports. If you want to generate reports, you need only some of these capabilities. There is the loading of necessary parts of the generator, which contain a certain set of capabilities. For example, if maps are not used in your reports, you shouldn't load them. This will accelerate web project load and reduce browser memory consumption.

Information

© 2003-2024 Stimulsoft

This capability is implemented only for the report generator engine, viewer and the designer can't be divided into parts, their scripts will be loaded fully by one block.

To use partial scripts loading you should set the required PHP options in code for the `StiJavaScript` class.

index.php

```
<head>
   <?php
      $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
      \StiComponentType::Report);

      $js->options->reports = false;
      $js->options->reportsChart = true;
      $js->options->reportsExport = true;
      $js->options->reportsImportXlsx = true;
      $js->options->reportsMaps = true;

      $js->renderHtml();
   ?>
</head>
```

When using the **JavaScript** code, you should remove the loading of the stimulsoft.reports.js script from the project, and add one or more scripts from the list below:

index.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.engine.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.export.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.chart.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.maps.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.import.xlsx.js" type="text/javascript"></script>
</head>
```

© 2003-2024 Stimulsoft

Each PHP option of the `StiJavaScript` object controls the loading of a script containing specific functionality. The table contains the whole set of scripts, which you can load separately.

Name	Description
`stimulsoft.reports.engine.js`	It contains only all what you need to load and render a report, work with data. It's necessary for linking.
`stimulsoft.reports.export.js`	It contains algorithms to export a rendered report in various formats – PDF, HTML, Excel, RichText and the others.
`stimulsoft.reports.chart.js`	It contains components for work with all types of charts in a report.
`stimulsoft.reports.maps.js`	It contains components for work with regional and online maps.
`stimulsoft.blockly.editor.js`	It contains visual editor Blockly for creating event scripts in a report. The Event handler is embedded in report engine.
`stimulsoft.reports.import.xlsx.js`	It contains algorithms for work with Excel data sources.

Information

Connection sequence of any Stimulsoft scripts doesn't matter, so as all scripts have additional checks of postponed loading and the run of necessary modules.

10.1.3 Activation

After purchasing the product, you should activate license for used components. There are several ways of license key connection.

Activation using code string

© 2003-2024 Stimulsoft

To activate using a string, you should copy an encrypted license text from your personal account and register it in the event handler. You should place the code until you create and render your report.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setKey('Your activation code...');
   $handler->renderHtml();
?>
```

There is an option to load a license key using JavaScript code on the project page. You should place the code until you create and render your report.

index.php

```
Stimulsoft.Base.StiLicense.Key = "Your activation code...";
```

Activation using a file

To activate using a license file you should download the `license.key` file from your personal account, copy it to the folder of the PHP project, and register it in the event handler.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setFile('license.key');
   $handler->renderHtml();
?>
```

There is an option to load a license key using JavaScript code on the project page. You should place the code until you create and render your report.

index.php

© 2003-2024 Stimulsoft

```
Stimulsoft.Base.StiLicense.loadFromFile("license.key");
```

Protection against license key theft

If you activate license using a string, you can add its code by a certain condition. For example, you want to add a license key for registered users.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   if (!empty($sessionID))
      $handler->license->setKey('Your activation code...');
   $handler->renderHtml();
?>
```

Also, we recommend you change the location and name of the license key file, for example:

license.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setFile('private/a15fc0ef64e6.key');
   $handler->renderHtml();
?>
```

10.1.4 Loading and Saving Reports

Information

The Stimulsoft MRT file is a description of reports with XML or JSON markup. You can use MRT files, created in other Stimulsoft designers.

To load a report using **PHP** code, you can use one of the functions of the `StiReport` object - `loadFile()` or `load()`. Each of the functions takes as input a parameter - the name of the report file:

© 2003-2024 Stimulsoft

Name	Description
`loadFile($filePath, $load = false)`	Loads a report from an MRT file on the client-side, the path to which is specified in the function arguments. If the `$load` parameter is set to `true`, the report file will be loaded on the server side and passed to the client side as a packed `Base64` string.
`load($filePath)`	Loads a report from an XML or JSON string and sends it to the client side as a packed `Base64` string.

For example, you need to load a report from a file on the server side from a private directory and pass it to the client side:

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->loadFile('reports/SimpleList.mrt', true);
   $report->renderHtml();
?>
```

To load a report using **JavaScript** code, you can use one of the functions of the `StiReport` object - `loadFile()` or `load()`.

Name	Description
`loadFile(filePath)`	It loads a report from a MRT file, the path to which is specified in the function arguments.
`load(str)`	It loads a report from the string, which contains the report in XML or JSON format.

© 2003-2024 Stimulsoft

`load(data)`	It loads a report from data array of the `number[]` type.
`load(xml)`	It loads a report from an XML object of the XMLDocument type.
`load(json)`	It loads a report from an JS object.

For example, you need to load a report from a file.

index.php

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("reports/SimpleList.mrt");
```

Additional report formats

Additionally, a report template can be presented in the Stimulsoft MRZ format (a packed report) and the Stimulsoft MRX format (a password-encrypted report). The MRZ format allows you to save files with smaller size, not using other utilities of packaging. The MRX format is encrypted with a password and allows you to hide confidential data, saved in your report.

There are special functions to load a report in the formats mentioned above.

Name	Description
`loadPackedFile(filePath)`	It loads a packed report from a MRZ file, the path to which specified in the function arguments.
`loadPacked(str)`	It loads a packed report from the string, which presented in the Base64 format.
`loadPacked(data)`	It loads a packed report from data array of the `number[]` type.
`loadEncryptedReportFile(filePath, key)`	It loads an encrypted report from a MRX file, the path to which specified in the function arguments. You

© 2003-2024 Stimulsoft

	should specify string encryption key, which set when saving.
`loadEncryptedReport(str, key)`	It loads an encrypted report from the string specified in the Base64 format. You should specify string encryption key as the second argument, which set when saving.
`loadEncryptedReport(data, key)`	It loads an encrypted report from the `number[]` data array type. You should specify the string encryption key as the second argument, which set when saving.

To save a report you can use the following functions.

Name	Description
`saveToJsonString()`	It saves a report as a row in the JSON format.
`savePackedToString()`	It saves a report as a row, presented in the Base64 format.
`savePackedToByteArray()`	It saves a packed report as an array of the `number[]` data type.
`saveEncryptedReportToString(key)`	It saves an encrypted report as a row, presented in the Base64 format. You should specify a string encryption key as argument.
`saveEncryptedReportToByteArray(key)`	It saves an encrypted report as an array of the `number[]` data type. You should specify a string encryption key as argument.

Information

The Stimulsoft Reports.PHP report generator and the Stimulsoft Dshboards.PHP

© 2003-2024 Stimulsoft

dashboards based on the JavaScript platform, and they support saving MRT files in the JSON format only. MRT files in the XML file are supported in the loading mode only. In addition, they will be automatically converted to the JSON format when saving.

Loading and saving documents (rendered reports)

A rendered report can be saved as a document in the JSON format for further view or export. Such document contains rendered report pages, but it doesn't contain initial data and doesn't save the connection to them. For example, you need to save the document in the JSON string.

To load a document using PHP code, you can use one of the functions of the `StiReport` object - `loadDocumentFile()` or `loadDocument()`. Each of the functions takes as input a parameter - the report file name:

Name	Description
`loadDocumentFile($filePath, $load = false)`	Loads a document from an MDC file on the client-side, the path to which is specified in the function arguments. If the `$load` parameter is set to `true`, the document file will be loaded on the server side and passed to the client side as a packed `Base64` string.
`loadDocument($filePath)`	Loads a document from an XML or JSON string and passes it to the client side as a packed `Base64` string.

For example, you need to load a document from a file on the server side from a private directory and pass it to the client side:

index.php

© 2003-2024 Stimulsoft

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->loadDocumentFile('reports/SimpleList.mdc', true);
   $report->renderHtml();
?>
```

Information

So as you always need data to create dashboards, they can't be saved as a document. The Stimulsoft Dashboards.PHP supports saving dashboards as a template only.

The corresponding functions of the `StiReport` object are used to load and save a document using **JavaScript** code. The full list of functions and parameters you may find in the table below.

Sample of document loading from a file:

index.php

```
var report = new Stimulsoft.Report.StiReport();
report.loadDocumentFile("reports/SimpleList.mdc");
```

Here is a sample of loading a report template, building it and saving the rendered document as a JSON string:

index.php

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("reports/SimpleList.mrt");
report.renderAsync(function() {
   var jsonString = report.saveDocumentToJsonString();
});
```

A document can be presented in three different formats as well as report template:

© 2003-2024 Stimulsoft

Stimulsoft MDC (a simple document), Stimulsoft MDZ (a packed document) and Stimulsoft MDX (an encrypted document by a password). A full list of functions to work with all document formats is in the following table.

Name	Description
`loadDocumentFile(filePath)`	It loads a document from a MDC file, the path to which specified in the function arguments.
`loadDocument(str)`	It loads a document from the string, which contains a report in the XML or JSON.
`loadDocument(data)`	It loads a document from data array of the `number[]` type.
`loadDocument(xml)`	It loads a document from an XML object of the XMLDocument type.
`loadDocument(json)`	It loads a document from a JS object.
`loadPackedDocumentFile(filePath)`	It loads a packed document from a MDZ file, the path to it specified in the function arguments.
`loadPackedDocument(str)`	It loads a packed document from the string, presented in the Base64 format.
`loadPackedDocument(data)`	It loads a packed document from data array of the `number[]` type.
`loadEncryptedDocumentFile(filePath, key)`	It loads an encrypted document from a MDX file, the path to it specified in the function arguments. You should specify the string encryption key set when saving as the second argument.
`loadEncryptedDocument(str, key)`	It loads an encrypted document from the string presented in the Base64 format. You should specify the string encryption key set when saving as the second argument.

© 2003-2024 Stimulsoft

`loadEncryptedDocument(data, key)`	It loads an encrypted document from data array of the `number[]` type. You should specify the string encryption key set when saving as the second argument.
`saveDocumentToJsonString()`	It saves a document as the string in the JSON format.
`savePackedDocumentToString()`	It saves a packed document as the string presented in the Base64 format.
`savePackedDocumentToByteArray()`	It saves a packed document as data array of the `number[]` type.
`saveEncryptedDocumentToString(key)`	It saves an encrypted document as the string presented in the Base64 format. You should specify the string encryption key as the second argument.
`saveEncryptedDocumentToByteArray(key)`	It saves an encrypted document as data array of the `number[]` type. You should specify the string encryption key as the second argument.

Information

When saving a document from the viewer menu, a file is saved in the JSON format too, and it has the MDC expansion for a standard document, the MDZ for a packed document, the MDX for an encrypted document.

10.1.5 Rendering a Report

To build a loaded report, you should call the `render()` function on the report object. For example, you want to build a report before exporting it.

index.php

```
<?php
```

© 2003-2024 Stimulsoft

```
    $report = new \Stimulsoft\Report\StiReport();
    $report->loadFile('SimpleList.mrt');
    $report->render();
    $report->exportDocument(\Stimulsoft\StiExportFormat::Pdf);
    $report->renderHtml();
?>
```

If you need to perform any actions after rendering a report using **JavaScript** code, you should pass the name of the **JavaScript** function as a parameter. For example, after building a report, you need to display the following message:

index.php

```
<?php
    $report = new \Stimulsoft\Report\StiReport();
    $report->loadFile('SimpleList.mrt');
    $report->render('onAfterRender');
    $report->renderHtml();
?>

function onAfterRender() {
    alert("The report rendering is completed.");
}
```

If you need to perform any actions with the report before it is built using **JavaScript** code, you should specify the name of the **JavaScript** function for the `onBeforeRender` event. The action type, and the report itself will be passed in the function arguments. For example, before building a report, you need to register JSON data:

index.php

```
<?php
    $report = new \Stimulsoft\Report\StiReport();
    $report->onBeforeRender = 'onBeforeRender';
    $report->render();
    $report->renderHtml();
?>

function onBeforeRender(args) {
    var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
    dataSet.readJsonFile("Demo.json");

    var report = args.report;
```

© 2003-2024 Stimulsoft

```
    report.regData(dataSet.dataSetName, "", dataSet);
}
```

10.1.6 PHP Events Handler

Samples

You can find a full code on the GitHub.

The report generator, the viewer, and the report designer can invoke events on the client-side, send them to the PHP server for next processing, and take a prepared report from the PHP server. Everything you need to work with the events you may find in the `StiHandler` class, which you need to add to the `<script>` block before the rest of the components:

index.php

```
<?php
    $handler = new \Stimulsoft\StiHandler();
    $handler->renderHtml();
?>
```

If necessary, you can set the server script file of the event handler in the options and the timeout for waiting for a response:

index.php

```
<?php
    $handler = new \Stimulsoft\StiHandler();
    $handler->options->url = 'handler.php';
    $handler->options->timeout = 30;
    $handler->renderHtml();
?>
```

After performing the `renderHtml()` function, the **JavaScript** code necessary for working with events will be added to the **PHP** page. To transfer data to the server

© 2003-2024 Stimulsoft

side, you should enable the processing of the required event. Set the corresponding property of the component to `true`:

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = true;
   $report->renderHtml();
?>
```

It is possible to define a **JavaScript** function for the event if it is necessary to process the parameters passed to the server side before passing them. In this case, all passed parameters will be contained in the function arguments, and if necessary, the values of the arguments can be changed:

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = 'onBeginProcessData';
   $report->renderHtml();
?>

function onBeginProcessData(args) {

}
```

In the event handler on the PHP server-side, in the event of the same name in arguments, you can read all available parameters, transferred from the report generator, check them and correct something, if needed. For example, in this case the `onBeginProcessData` event will be invoked in the `handler.php` file.

handler.php

```
$handler->onBeginProcessData = function ($args) {
   ...

   return \Stimulsoft\StiResult::success();
};
```

© 2003-2024 Stimulsoft

After the event processing on the PHP server-side, the handler should return a response to the client-side about the result of the event. Two static functions are provided for it: the `StiResult::success()` is the event made successfully, and the `StiResult::error()` is an error occurred during the execution. You control this process and make decision about the response you need. You can transfer a string message, which will be displayed on the client-side after the event as the function parameter, for example.

handler.php

```
$handler->onBeginProcessData = function ($args) {
   ...

   // You can send a successful result
   return \Stimulsoft\StiResult::success();

   // You can send an informational message
   //return \Stimulsoft\StiResult::success('Some warning or other useful
   information.');
   // You can send an error message
   //return \Stimulsoft\StiResult::error('A message about some error.');
};
```

Information

There is an option to automatically send information about an error to the client-side if critical errors occur on the PHP server-side. If the specified functionality is not required, it can be disabled by setting false to the `$registerErrorHandlers` parameter in the server-side event handler constructor:

```
$handler = new \Stimulsoft\StiHandler(false);
```

Data encryption, transmitted to the PHP server-side

To avoid theft of transmitted data to intruders, we recommend you to use HTTPS protocol, mostly that's not enough. In addition, by default all transmitted data goes through a special coding algorithm and transmitted to the server in encrypted form. It allows you to hide your confidential data, such as log in and password in the

© 2003-2024 Stimulsoft

connection string from curious users, who work with your application.

However, if it's not required or if you need to display the input data of a request to debug an application, the encryption disable capability is envisaged. To do this, just set the `$encryptData` property to false in the event handler:

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->encryptData = false;
   $handler->renderHtml();
?>
```

Transferring GET parameter values to the PHP event handler

You can automatically transfer all the values of the GET request parameters to the event handler, where their values can be accessed in all events. To enable this feature, simply set the `passQueryParameters` option to `true`, and all GET request parameters will be passed with each request to the event handler. By default, this option is disabled.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->options->passQueryParameters = true;
   $handler->renderHtml();
?>
```

10.1.7 Connecting Data Files

As a rule, connection parameters to data source are saved in report template. To work with file data source, such as XML, JSON, Excel, CSV you don't need to make any additional actions. So as all algorithms are in the report generator script. If necessary, you can use other methods to connect data using JavaScript functions of the reporting tool. Use the `onBeforeRender` event of the report object. Data can be loaded directly into the `DataSet` special object, used to save them. It contains functions for data loading from XML/XSD and JSON file formats. After file loading, you should invoke the `regData()` function of the report object to connect data. In the function arguments the prepared `DataSet` object is specified.

© 2003-2024 Stimulsoft

Sample of data loading from an XML file using XSD scheme.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeforeRender = 'onBeforeRender';
   $report->renderHtml();
?>

function onBeforeRender(args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readXmlSchemaFile("Demo.xsd");
   dataSet.readXmlFile("Demo.xml");

   var report = args.report;
   report.regData(dataSet.dataSetName, "", dataSet);
   report.dictionary.synchronize();
}
```

Information

Data scheme loading is not essential. If you want to use data scheme, you should add it before XML data loading.

Data loading from a JSON file sample.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeforeRender = 'onBeforeRender';
   $report->renderHtml();
?>

function onBeforeRender(args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readJsonFile("Demo.json");

   var report = args.report;
   report.regData(dataSet.dataSetName, "", dataSet);
   report.dictionary.synchronize();
}
```

© 2003-2024 Stimulsoft

There are the `readXml()` and the `readJson()` functions apart from the `readXmlFile()` and the `readJsonFile()`. They take data as a string or an object.

Information

The `report.dictionary.synchronize()` function is needed to synchronize connected data with data dictionary of report template. So as, when invoking this function, report dictionary created on the basis of data structure, loaded in the `DataSet`. The synchronization function is not required, if dictionary created in advance and its structure matches connected data.

Events of work with data

There is a capability to subscribe to events of work with data for more flexible work with file data.

Data loading event

To view and change parameters of file data connection before load them, you should define the `onBeginProcessData` event of a report object or the viewer or the designer component.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = 'onBeginProcessData';
   $report->renderHtml();
?>

function onBeginProcessData(args) {
   var pathData = args.pathData;
}
```

In arguments of the event the information about connection to a file data source – name and type of connection in a report template and the path to the data file will

© 2003-2024 Stimulsoft

be transferred. You can find a detailed description of available values of arguments in the Engine events chapter.

You can change the path to the data file. In this case, after the event completed, the report generator will request a file by a new path, specified in the arguments. For example, you need to change the path to the JSON data file for a specified connection.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = 'onBeginProcessData';
   $report->renderHtml();
?>

function onBeginProcessData(args) {
   if (args.connection == "MyJsonConnection")
      args.pathData = "Data/Demo.json";
}
```

Information

The `onBeginProcessData` event will be invoked twice for an XML data source: first time to read an XSD scheme, second time to read an XML data file.

Data connection event

To view or correct loaded data before connect them and render a report you should define the `onEndProcessData` for a report object or the viewer or the designer component.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onEndProcessData = 'onEndProcessData';
   $report->renderHtml();
?>

function onEndProcessData(args) {
   var dataSet = args.dataSet;
```

© 2003-2024 Stimulsoft

```
}
```

In arguments of the event information about connection to a file data source – name and type of connection, saved in a report template and prepared DataSet object, which contains tables and data rows, received from the file source. You can find a detailed description of available values of the arguments in the Engine events chapter.

Loading data files on the PHP server-side

Sometimes you have to control loading of a data file on the server-side or for example create data array using a code. To do it you should specify the path to a PHP script or the service, which contains logic of getting data instead of specifying the path to the data file. The simplest PHP script of data loading will look like this.

json.php

```
<?php
   echo file_get_contents('Data/Demo.json');
?>
```

In this case, you should specify URL address as the path to data in a report template to this file, for example.

File Data Source

```
https://localhost/data/json.php
```

Using variables in data files

There is an ability, which allows you to use variables as expressions (and use expressions) when specifying the path to a file data source. A variable or an expression is specified in braces. You can use several expressions in any place of the path to a data file. For example.

File Data Source

© 2003-2024 Stimulsoft

```
https://localhost/data/{VariableJsonFileName}.json
https://localhost/data/json.php?id={VariableId}
https://localhost/{VariableCategory}/{VariableId}
```

This way, one data source can be converted to the REST syntax. It will allow you not to create several the same data sources to get similar data and also in conjunction with the server PHP logic and the report generator events to make the data source more flexible.

Using the OData

Also, you can use data to create reports, received from the OData storages. In this case you should authorize using a user's name, user's password or a token. Authorization parameters are specified in the string of connection to a OData storage with the help of the ";" separator.

index.php

```
// Authorization using a user account
var oDataDatabase = new
Stimulsoft.Report.Dictionary.StiODataDatabase("OData", "OData", "https://
services.odata.org/V4/Northwind/
Northwind.svc;AddressBearer=adress;UserName=UserName;Password=Password;Cli
ent_Id=Your Client ID", false, null);

// Authorization using a user token
var oDataDatabase = new
Stimulsoft.Report.Dictionary.StiODataDatabase("OData", "OData", "https://
services.odata.org/V4/Northwind/Northwind.svc;Token=Enter your token",
false, null);

report.dictionary.databases.add(oDataDatabase);
report.dictionary.synchronize();

// Query with data filter
var productsDataSource =
report.dictionary.dataSources.getByName("Products");
if (productsDataSource != null) productsDataSource.sqlCommand =
"Products?$filter=ProductID eq 2";
```

10.1.8 Connecting SQL Data Adapters

Samples

You can find a full code on the GitHub.

© 2003-2024 Stimulsoft

To render reports the report generator allows you to use data from various SQL sources. So as pure JavaScript doesn't have embedded methods to work with remote database, this functionality is implemented using a server PHP code. Special events of work with data on the side of JavaScript client and PHP server are intended for this.

Loading data event

To work with SQL data source you should define the `onBeginProcessData` event for a report object or the viewer or the designer component. The event will be invoked when designing a report as soon as you need to connect to used SQL data sources. In arguments of the event all necessary parameters of connection to a SQL data source and SQL parameters of a request too. See the PHP Event Handler section for a detailed description of how events work.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProceddData = true;
   $report->renderHtml;
?>
```

If it is necessary to process the parameters passed to the server side before passing them, you may define a JavaScript function for the event:

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProceddData = 'onBeginProcessData';
   $report->renderHtml;
?>

function onBeginProcessData(args) {

}
```

© 2003-2024 Stimulsoft

The event arguments will contain all necessary parameters for connecting to the SQL source, as well as SQL query parameters.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = 'onBeginProcessData';
   $report->renderHtml();
?>

function onBeginProcessData(args) {
   var connectionString = args.connectionString;
   var queryString = args.queryString;
}
```

You can find a detailed description of available values of arguments in the Engine events chapter.

All parameters of connection to a data source can be changed. For example, you need to set a new connection string for specified connection and set a new SQL query for a specified data source.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = 'onBeginProcessData';
   $report->renderHtml();
?>

function onBeginProcessData(args) {
   // Change the connection string
   if (args.connection == "MyConnectionName")
      args.connectionString =
      "Server=localhost;Database=test;uid=root;password=******;";

   // Change the SQL query
   if (args.dataSource == "MyDataSource")
      args.queryString = "SELECT * FROM MyTable";
}
```

© 2003-2024 Stimulsoft

You can set parameters of an SQL query for each SQL data source, which saved in the data source as collection. This collection is also transferred in arguments of the event. It is an array from the objects, which contain parameters name, its type and value.

index.php

```
args.parameters = [
   {
      name: "ParameterString",
      type: 752,
      typeName: "Text",
      value: "Text value"
   },
   {
      name: "ParameterInt",
      type: 3,
      typeName: "Int32",
      value: 20
   }
];
```

You can change values of a request parameters and the type of a new value should correspond to the type of a changed parameter.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeginProcessData = 'onBeginProcessData';
   $report->renderHtml();
?>

function onBeginProcessData(args) {
   // Change the SQL query parameters with the required values
   if (args.parameters)
      args.parameters["Parameter1"] = "TableName";
}
```

Information

All types of an SQL query will correspond to available types of database tables, to

© 2003-2024 Stimulsoft

which connection is made. You can find more detailed information about an SQL query parameters in the Using parameters in an SQL query chapter.

To change the connection parameters on the PHP server side, you should additionally define the `onBeginProcessData` event of the same name in the event handler on the server side and assign a new value to a specific parameter. All other actions will be performed automatically. For example, for the specified connection, set a new connection string, and for the specified data source, set a new SQL query:

handler.php

```
$handler->onBeginProcessData = function ($args) {
   // Change the connection string
   if ($args->connection == "MyConnectionName")
      $args->connectionString =
      "Server=localhost;Database=test;uid=root;password=******;";

   // Change the SQL query
   if ($args->dataSource == "MyDataSource")
      $args->queryString = "SELECT * FROM MyTable";

   return \Stimulsoft\StiResult::success();
};
```

This way in the `onBeginProcessData` event, you can find the type of database, connection name, and data source name. In addition, you can find and if needed correct the connection string and a SQL query for getting data. The changed values will not be transferred to the client-side, so in this event you can use such confidential data on the PHP server-side of as login, password in connection string, names, and prefixes of tables etc.

In addition, in this event you can set parameter values of a SQL query. You can find more detailed information about a SQL query in this section in the Using parameters in a SQL query chapter.

Connection data event

To view or correct loaded data before connect them and render a report you should define the `onEndProcessData` event for a report object or the viewer or the

© 2003-2024 Stimulsoft

designer component.

index.php

```
<?php
   $report-> = new \Stimulsoft\Report\StiReport();
   $report->onEndProcessData = 'onEndProcessData';
   $report->renderHtml();
?>

function onEndProcessData(args) {
   var result = args.result;
}
```

In arguments of the event all necessary parameters of connection to a SQL data source will be transferred and the result of a request execution, which contains tables, types and data strings, received from the SQL source. You can find a detailed description of available values of arguments in the Engine Events chapter.

An object of data of an executed SQL query has the following structure.

index.php

```
args.result = {
   count: 3,
   columns: ["id", "username", "phone"],
   types: ["int", "string", "string"],
   rows: [
      length: 2,
      ["1", "Mario Pontes", "555-6874"],
      ["2", "Helen Bennett", "555-2376"]
   ]
}
```

Available properties of the data object are specified in this table.

Name	Description
count	The total number of columns of a SQL data source table.
columns	The names of columns of a SQL data

© 2003-2024 Stimulsoft

	source table.
`types`	The types of columns of a SQL data source table, cast to known types for the report generator.
`rows`	Data strings of a SQL data source are an array from all table rows.

All data of the result of a SQL query execution can be corrected, in this case after the event completed, a report will be designed with changed data already.

To view or correct loaded data on the PHP server-side before sending them to the side of JavaScript client you should define the `onEndProcessData` event in the event handler file.

handler.php

```
$handler->onEndProcessData = function ($args) {
   $result = $args->result;

   return \Stimulsoft\StiResult::success();
};
```

In arguments of the event all necessary parameters of connection to a SQL data source will be transferred and the result of a request execution, which contains tables, types and data strings, received from a SQL source. You can find a detailed description of available values of arguments in the Engine events chapter.

An object of data of an executed SQL query has the following structure.

handler.php

```
$args->result->count = 3;
$args->result->columns = array("id", "username", "phone");
$args->result->types = array("int", "string", "string");
$args->result->rows = array(
   array("1", "Mario Pontes", "555-6874"),
   array("2", "Helen Bennett", "555-2376")
);
```

© 2003-2024 Stimulsoft

Available properties of the data object are specified in the table.

Name	Description
`count`	The total number of columns of a SQL data source table.
`columns`	The names of columns of a SQL data source table.
`types`	The types of columns of a SQL data source, cast to known types for the report generator.
`rows`	Data rows of a SQL data source are an array from arrays of all table arrays.

All data of the result of a SQL query execution can be corrected, in this case after the event completed, changed data will be transferred to the client-side and used to design a report.

Using parameters in an SQL query

If needed, you can use parameters in a SQL query. To do it in a data source you should add parameters to a special collection and set an appropriate type and value for each parameter by default. After this, you can use parameters in a SQL query by the following way.

SQL Data Source

```
SELECT * FROM @Parameter1 WHERE UserID = @Parameter2
```

All values of the parameters will be transferred to the PHP server-side as a separate collection and they can be checked and changed before a SQL query completed. To get access to values of parameters, you should use the `$args->parameters` in the `onBeginProcessData` event, for example.

© 2003-2024 Stimulsoft

handler.php

```
$args->parameters['Parameter1']->value = 'TableName';
$args->parameters['Parameter2']->value = 10;
```

Information

New values of parameters in this collection will not transferred to the client-side, so you can assign confidential data as values on the PHP server-side.

If several data source are used in a report, you should check it before assign parameters, otherwise, an error of PHP script will occur, if one of the parameters will be absent in a current data source. For example, two data source are created, both of them has one common parameter and one parameter only in the second data source.

handler.php

```
$args->parameters['Parameter1']->value = 'TableName';

if ($args->dataSource == 'DataSource2')
   $args->parameters['Parameter2']->value = 10;
```

Using variable as SQL parameter

There is an ability, which allows you to use a variable as SQL parameter. To do it you should set the **Allow using as SQL parameter** property in the editor of report variable. After it can be used in any SQL query. The syntax will be the same as when using parameters in data source.

Information

This variable will be transferred to the collection of parameters only if it is used in a request. The parameters from data source collection are always transferred, even if they are not used in a query.

© 2003-2024 Stimulsoft

Escaping parameter values

All values of parameters will be automatically escaped to exclude SQL injections and compromise the security of query execution. If you don't need to escape and you control the security of safety of parameter values, automatic escaping can be disabled. To do this, set the `$escapeQueryParameters` property to `false` in the event handler:

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->escapeQueryParameters = false;
   $handler->renderHtml();
?>
```

After you set the specified property, the use of parameters is unsafe. You need to strictly control the values before executing SQL queries.

Information

Escaping is applied only for parameters of SQL queries, and for the variables used as parameters. If a variable is used as an expression i.e in braces, for example {VariableName}, escaping will not be applied in any case. You can find a detailed description of variable work in the Work with report variables chapter.

10.1.9 Work with Report Variables

The report generator allows you to use variables in expressions, requests, filters and other report elements. There is an ability, which allows you preview and change values of variables from a code before you render a report.

Access to values of variables from a code

The report generator allows you easily to get access to report variables via the data dictionary. To do it you should use the `getByName()` function for the collection of

© 2003-2024 Stimulsoft

variables. To change values of a variable you should assign it in the value property and you don't need to check the set type of a variable and the conversion of types will be done automatically. For example you need to change string and integer value of specified variables.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onBeforeRender = 'onBeforeRender';
   $report->renderHtml();
?>

function onBeforeRender(args) {
   var report = args.report;

   var variableString =
   report.dictionary.variables.getByName("VariableString");
   variableString.value = "Text value";

   var variableInt = report.dictionary.variables.getByName("VariableInt");
   variableInt.value = "20";
}
```

The event of preparing values of variables

To get access to values of variables before designing a report, you should define the `onPrepareVariables` event for a report object or the viewer or the designer component.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->onPrepareVariables = 'onPrepareVariables';
   $report->renderHtml();
?>

function onPrepareVariables(args) {
   var variables = args.variables;
}
```

In arguments of the event, the collection of report variables with their values will be transferred. If a variable is initialized as an expression, a calculated value of the

© 2003-2024 Stimulsoft

expression will be transferred to the collection. You can find a detailed description of values of arguments in the Engine Events chapter.

The collection of variables is an array of objects, which contain the name of a variable, its type and value. For example.

index.php

```
args.variables = [
   {
      name: "VariableString",
      type: "String",
      value: "Text value"
   },
   {
      name: "VariableInt",
      type: "Int32",
      value: 20
   }
];
```

You can change values of variables and the type of a new value should correspond to the type of a changed variable.

To get access to values of report values on the PHP server-side you should add the invoke of event handler on the client-side, and define the `onPrepareVariables` event with the same name in the file of event handler. You can find a detailed description of work of events in the PHP events handler chapter.

index.php

```
<?php
   $report-> = new \Stimulsoft\Report\StiReport();
   $report->onPrepareVariables = true;
   $report->renderHtml();
?>
```

In arguments of the event the collection of report variables, which is an associative array where the name of a variable is a key and the object, which contains the type of a variable and its value. For example.

© 2003-2024 Stimulsoft

handler.php

```
$handler->onPrepareVariables = function ($args) {
   $args->variables['VariableString']->value = 'Text value from Server-
   Side';

   $args->variables['VariableInt']->value = 123;
   $type = $args->variables['VariableInt']->type; // 'Int32'

   return \Stimulsoft\StiResult::success();
};
```

You can change values of variables and the type of a new value should correspond to the type of a changed variable. Also, you can create a new report variable, if needed. A new collection, which contains only the variables with changed values and new variables will be transferred to the client-side.

To change the value of a simple variable you should change a value in the collection of variables. A value should be of the same type as an original. The values of the DataTime type are transferred as a string value in the "YYYY-MM-dd HH-mm-ss": format.

handler.php

```
$args->variables['VariableString']->value = 'Value from Server-Side';
$args->variables['VariableInt']->value = 123;
$args->variables['VariableDecimal']->value = 123.456;
$args->variables['VariableDateTime']->value = '2021-03-20 22:00:00';
```

To change a value of the Range type variable, you should use the nested `value->from` and `value->to` values of the variable you select in the collection. The format of each of two values is similar to a simple variable.

handler.php

```
$args->variables['VariableStringRange']->value->from = 'Aaa';
$args->variables['VariableStringRange']->value->to = 'Zzz';
```

© 2003-2024 Stimulsoft

To change the values of the List type variable, you should use a value from the list by its index. You can set values of all the list as a prepared array.

handler.php

```
$args->variables['VariableStringList']->value[0] = 'Test';
$args->variables['VariableStringList']->value = ['1', '2', '2'];
```

To create a new variable, which is not defined in a report you should assign a prepared associative array in the `['value' => 'New Value']` format to the collection of variables using a new variable's name. After, you can use the variable to render a report, it means the variable will not be saved in the report template.

handler.php

```
$args->variables['NewVariable'] = ['value' => 'New Value'];
```

Information

If a variable is used in a SQL query as an expression, i.e. it is typed in braces, for example the `{VariableName}` , its value will not be automatically escaped. You should keep an eye on the safety of values or use a variable as parameter of query, for example `@VariableName`. You can find a detailed description of work of parameters in the Connecting SQL data adapters chapter.

Report variables transferred in a URL query

The report generator has an ability, which allows you automatically to assign values of variables, transferred in the URL query. To do this, set the `$passQueryParametersToReport` property of the event handler to `true`:

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->passQueryParametersToReport = true;
```

© 2003-2024 Stimulsoft

```
    $handler->renderHtml();
?>
```

The report generator will make all other actions automatically. If there is a variable in a report, its value will be changed to a value from the URL query. If there is no such variable in a report, it will be created to render the report. It means, a new variable will not be saved in the report template. The names of variables, transferred in the URL query are not case sensitive.

Information

All values of variables will be assigned before the `onPrepareVariables` event invoked. This way in this event, you can additionally control the set values and correct them, if needed.

10.1.10 Printing Report from Code

In the report generator you can print a report from a code. To do it you should use a special method the `print()` for a report object.

index.php

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.renderAsync(function() {
    report.print();
});
```

Information

Print a report doesn't call its automatic rendering, so you should call in advance the `renderAsync()` method, and it will render a report. This method is not required for ready documents (rendered reports).

When printing a report you can specify a range of pages for printing. The special

© 2003-2024 Stimulsoft

`StiPagesRange` class is intended for this and you can set the following parameters as constructor arguments.

Name	Description
`rangeType`	The type of set range. The following values are available: `Stimulsoft.Report.StiRangeType.All` – is print of all pages, other parameters of the constructor are not taken into account (the value by default); `Stimulsoft.Report.StiRangeType.Pages` – is print of a specified in the second argument of pages range argument; `Stimulsoft.Report.StiRangeType.CurrentPage` – is print of the current page, you should specify the current number of the page in the third argument.
`pageRanges`	The range in a string representation is the numbers of pages separated by commas or via hyphen. It is used together with the `Stimulsoft.Report.StiRangeType.Pages` variant.
`currentPage`	The current number of a page. It is used together with the `Stimulsoft.Report.StiRangeType.CurrentPage` variant.

Samples of setting print range.

index.php

```
$pageRange = new Stimulsoft\Report\StiPagesRange(Stimulsoft\Report
```

© 2003-2024 Stimulsoft

```
\StiRangeType::All);
$report->print($pageRange);

$pageRange = new Stimulsoft\Report\StiPagesRange(Stimulsoft\Report
\StiRangeType::Pages, '1,3-8');
$report->print($pageRange);

$pageRange = new Stimulsoft\Report\StiPagesRange(Stimulsoft\Report
\StiRangeType::CurrentPage, '', 5);
$report->print($pageRange);
```

10.1.11 Export Report from Code

In the report generator, you can export a report from a code to the following formats:

- Document (Report Snapshot)
- PDF
- XPS
- PowerPoint
- HTML / HTML5
- Text
- Word / Excel
- ODT / ODC
- CSV
- SVG

You can use the `exportDocumentAsync()` method for a report object to export.

index.php

```
<?php
   $report = new \Stimulsoft\Report\StiReport();
   $report->loadFile('SimpleList.mrt');
   $report->render();
   $report->exportDocument(\Stimulsoft\StiExportFormat::Pdf);
   $report->renderHtml();
?>
```

Information

© 2003-2024 Stimulsoft

Report export doesn't invoke its automatic rendering, so for a loaded report template you should invoke the `render()` method, which will render a report. This method is not required for ready documents (rendered reports).

As arguments to the `exportDocument()` function, you must specify the required export format from the `StiExportFormat` enumeration.

Name	Description
`StiExportFormat::Document`	Saving a document (a rendered report).
`StiExportFormat::Pdf`	Saving in the Adobe PDF format.
`StiExportFormat::Xps`	Saving in the XPS (XML Paper Specification) format.
`StiExportFormat::Ppt2007`	Saving in the Microsoft PowerPoint format.
`StiExportFormat::Html`	Saving in the HTML format.
`StiExportFormat::Html5`	Saving in HTML5 format using SVG markup elements.
`StiExportFormat::Text`	Saving in the text format.
`StiExportFormat::Word2007`	Saving in the Microsoft Word format.
`StiExportFormat::Excel2007`	Saving in the Microsoft Excel format.
`StiExportFormat::Odt`	Saving in the OpenDocument Text format.
`StiExportFormat::Ods`	Saving in the OpenDocument Spreadsheet format.
`StiExportFormat::Csv`	Saving in the CSV (Comma-Separated Values) data format.
`StiExportFormat::ImageSvg`	Saving in the image of the SVG format.

© 2003-2024 Stimulsoft

After exporting the report, the resulting data stream will be passed to the web browser for download as a file. The file name and MIME data type will be determined automatically.

10.1.12 Engine Events

The report generator supports the events, which give an ability to make necessary operations before definite actions both on the JavaScript client-side and on the PHP server-side. You don't need any additional settings for work of events on the client-side. However, for work of events on the server-side you should add a special event handler to the page. You can find a detailed description of connection and the PHP event handler setting in the PHP Events Handler chapter.

The report generator supports the following events:

- onBeforeRender
- onBeginProcessData
- onEndProcessData
- onPrepareVariables

onBeforeRender

The event is called before the report is rendered. See the list below with the event handler arguments on the JavaScript client side:

Name	Description
event	The identifier of the current event is "BeforeRender".
sender	The identifier of the component that initialized this event can take the following values: • Report
report	The current report object.

onBeginProcessData

© 2003-2024 Stimulsoft

This event is invoked before data request, which are needed for rendering a report. The list of the event handler arguments on the JavaScript client-side is presented in the table below.

Name	Description
`event`	The identifier of the current event has the "`BeginProcessData`" value.
`sender`	The identifier of the component, which initialized this event can take the following values: • `Report` • `Viewer` • `Designer`
`report`	Current report object
`command`	The identificator of the current command can take the following values: • `TestConnection` - a connection check is performed; • `ExecuteQuery` - data query is performed from the specified SQL source. • `GetSchema` - the XSD scheme is read from a file source. • `GetData` - data is read from a file source.
`connection`	The name of the current connection to a data source, specified in report template.
`connectionString`	Connection string to the SQL data source.
`queryString`	The SQL query for getting data. It is used only with the `ExecuteQuery`

© 2003-2024 Stimulsoft

	command.
`database`	The name of the current database. It can take the following values: • `XML` • `JSON` • `Excel` • `CSV` • `MySQL` • `MS SQL` • `PostgreSQL` • `Firebird` • `Oracle`
`pathData`	The path to the data source file, specified in the report template. It's set only for XML and JSON data sources.
`pathSchema`	The path to the data scheme file, specified in the report template. It is set only for XML data source.
`dataSource`	The name of the current data source, specified in the report template. It is set only for SQL data sources.
`parameters`	The collection of parameters and their values, specified in an SQL data source.
`preventDefault`	This flag is an ability to stop further event processing by the report generator. The `false` value is set by default.

In the table below, you can find the list of event handler arguments on the PHP server-side.

© 2003-2024 Stimulsoft

Name	Description
sender	The identificator of the component, which initialized this event can take the following values: • StiComponentType::Report • StiComponentType::Viewer • StiComponentType::Designer
command	The identificator of the current command can take the following values: • StiDataCommand::TestConnection - the connection is checked; • StiDataCommand::ExecuteQuery - the data is required from a specified SQL source.
connection	The name of the current connection to a data source, specified in report template.
connectionString	Connection string to the SQL data source.
queryString	The SQL query for getting data. It is used only with the StiDataCommand::ExecuteQuery command.
database	The name of the current database. It can take the following values: • StiDatabaseType::MySQL • StiDatabaseType::MS SQL • StiDatabaseType::PostgreSQL

© 2003-2024 Stimulsoft

	• StiDatabaseType::Firebird • StiDatabaseType::Oracle
dataSource	The name of the current data source, specified in report template.

You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL data adapters chapters.

onEndProcessData

The event is invoked after data loading before rendering a report. In the table below you can find the list of the event handler arguments on the side of JavaScript client.

Name	Description
event	The identificator of the current event has the "EndProcessData" value.
sender	The identificator of the component, which initialized this event can take the following values: • Report • Viewer • Designer
report	Current report object.
command	The identificator of the current command can take the following values: • ExecuteQuery - data is obtained from the specified SQL source. • GetData - data is obtained from a file source.
connection	The name of the current connection to a data source, specified in report table.

© 2003-2024 Stimulsoft

`database`	The name of the current database. It can take the following values: • `XML` • `JSON` • `Excel` • `CSV` • `MySQL` • `MS SQL` • `PostgreSQL` • `Firebird` • `Oracle`
`dataSource`	The name of the current data source, specified in report template. It is set only for SQL data sources.
`dataSet`	The prepared `Stimulsoft.System.Data.DataSet` object contains tables and data rows, received from a file source.
`result`	The collection of columns, their types and data rows, received from an SQL source.

In the table below, you can find the list of the event handler arguments on the PHP server-side.

Name	Description
`sender`	The identificator of the component, which initialized this event can take the following values: • `StiComponentType::Report` • `StiComponentType::Viewer`

© 2003-2024 Stimulsoft

	• StiComponentType::Designer
command	The identificator of the current command can take the following values: • StiDataCommand::TestConnection - the connection is checked; • StiDataCommand::ExecuteQuery - data is queried from the specified SQL source.
connection	The name of the current connection to a data source, specified in report template.
database	The name of the current database. It can take the following values: • StiDatabaseType::MySQL • StiDatabaseType::MS SQL • StiDatabaseType::PostgreSQL • StiDatabaseType::Firebird • StiDatabaseType::Oracle
dataSource	The name of the current data source, specified in report template. It is set only for SQL data sources.
result	The collection of columns, their types and data rows, received from an SQL source.

You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL data adapters chapters.

onPrepareVariables

© 2003-2024 Stimulsoft

This event is invoked before rendering a report after preparing the report variables. In the table below, you can see the list of the event handler arguments on the JavaScript client-side:

Name	Description
`event`	The identificator of the current event has the "`PrepareVariables`" variable.
`sender`	The identificator of the component, which initialized this event can take the following values: • `Report` • `Viewer` • `Designer`
`report`	Current report object.
`variables`	The collection of report variables and their values.
`preventDefault`	This flag gives an ability to stop the further event handler by the report generator. The `false` value is set by default.

In the table below you can find the list of the event handler arguments on the PHP server-side:

Name	Description
`sender`	The identificator of the component, which initialized this event can take the following values: • `StiComponentType::Report` • `StiComponentType::Viewer`

© 2003-2024 Stimulsoft

	• `StiComponentType::Designer`
`variables`	The collection of report variables and their values.

You can find a detailed description and samples of using in the Work with report variables chapter.

10.2 HTML5 Viewer

The report viewer is the HTML5 (`StiViewer`) component, intended for viewing, print and export reports to the window of the browser on any computer with any set operating system. The viewer supports a lot of themes, animated interface, bookmarks, interactive reports, edited report elements on a page, full screen mode, search and other necessary features for viewing reports.

The viewer can view both report template and a rendered report. In case of using report template, the viewer will render it automatically using the JavaScript report generator. You can find detailed information about the work of the generator in the Engine section.

The viewer uses the AJAX technology to make all actions (loading and rendering a report, flipping and zooming pages, interactivity in reports, print, export, etc) and it allows you not to reload the entire page and you can accelerate your work. The JavaScript, which is applied to report rendering, it allows you to use almost any server-side of low productivity.

Information

Taking into account that, the identical unified format of MRT template is used for dashboards and reports, the methods for loading the template and work with data, a report word will be used in the documentation text.

Deployment

Work with Bookmarks

© 2003-2024 Stimulsoft

- Activation
- Dynamic Sorting, Collapsing, and Drill-Down
- Showing Reports and Dashboards
- Editing Report
- Localization
- Sending Report By Email
- Printing Report
- Calling Designer from Viewer
- Report Export
- Appearance
- Viewing Modes
- Viewer Events
- Work with Report Variables
- Viewer Settings

10.2.1 Deployment

Samples

You can find a full code of on the GitHub.

To use the product, you should download the ZIP archive of the product from the Downloads page of our site, unpack it, and copy the contents of the **/PHP** folder to your Web server. This folder is a Web project that contains all the necessary files and resources for the product and samples for working with the viewer and designer.

To install the product in an existing project, you should copy the /vendor folder from the **/PHP** folder to the root directory of the project or use the Composer dependency manager by executing the following console command:

console

```
composer require stimulsoft/reports-php
```

When working with the product, in most cases, it is enough to use only PHP code, which ensures the operation of all the main features. For a more detailed configuration of the product and the use of all features, you must use JavaScript

© 2003-2024 Stimulsoft

code. Next, two options for using certain features will be described - using **PHP** and **JavaScript** functions. Depending on your needs, you may use either the first or the second option.

To use the HTML5 Viewer (`StiViewer`) component you should add specified scripts and styles to the `<head>` block on the PHP page. You can do this using the `StiJavaScript` **PHP** class. In the constructor, you need to specify the type of component to deploy:

viewer.php

```
<head>
   <?php
      $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
      \StiComponentType::Viewer);
      $js->renderHtml();
   ?>
</head>
```

Also, you may connect all the necessary scripts using the `<script>` block. All product scripts are located in the `vendor/stimulsoft/reports-php/public/scripts/` directory:

viewer.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/stimulsoft.viewer.js"
    type="text/javascript"></script>
</head>
```

For dashboards, you need to connect the following package:

console

```
composer require stimulsoft/dashboards-php
```

© 2003-2024 Stimulsoft

To view dashboards in the viewer, in addition you should add an appropriate script file, which contains all what is needed for work with dashboards.

index.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/dashboards-php/scripts/
   stimulsoft.dashboards.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/stimulsoft.viewer.js"
    type="text/javascript"></script>
</head>
```

Information

You can find possible variants of deployment optimizations in the Optimization of script loading chapter.

After that, you can use **PHP** classes and functions to work with the viewer, which must be placed inside the `<script>` block. To display the viewer on the page, you should create an object of the HTML5 Viewer component (`StiViewer`), a visual part of the component will be deployed where the following script will be invoked.

viewer.php

```
<script type="text/javascript">
   <?php
      $viewer = new \Stimulsoft\Viewer\StiViewer();
      $viewer->renderHtml();
   ?>
</script>
```

The viewer constructor arguments

The constructor of the `StiViewer` object can take optional arguments as input that affect its operation. There are two arguments in total:

© 2003-2024 Stimulsoft

viewer.php

```
<script type="text/javascript">
   <?php
      $viewer = new \Stimulsoft\Viewer\StiViewer($options, $viewerId);
      $viewer->renderHtml();
   ?>
</script>
```

Name	Description
`options`	This is a set of options found in the `Stimulsoft.Viewer.StiViewerOptions` class. The options are divided into categories and contain all what you need for setting behavior and appearance of the viewer. You can find a detailed description of categories and options in the Viewer settings chapter.
`viewerId`	String identificator of the HTML element of the viewer. The "`StiViewer`" value is used by default.

The viewer display variants

There are two options for displaying the viewer: immediately at the site of the script and inside the specified HTML element of the page. The second option can be useful if, for example, you need to create and execute view scripts in advance, display the viewer on a button by clicking, display the viewer on a dynamically built page, and in other cases.

To render the viewer, you need to call the `renderhtml()` special function on the created object:

viewer.php

```
<script type="text/javascript">
   <?php
      $viewer = new \Stimulsoft\Viewer\StiViewer();
```

© 2003-2024 Stimulsoft

```
    $viewer->renderHtml();
  ?>
</script>
```

When invoking `renderhtml()` function, visual part of the viewer will be rendered in the HTML page, where this function is invoked. If needed, the viewer can be displayed inside of the defined HTML element. In this case, you should transfer ID of this element or link to this element as function argument.

viewer.php

```
<script type="text/javascript">
  <?php
    $viewer = new \Stimulsoft\Viewer\StiViewer();
    $viewer->renderHtml('viewerContent');
  ?>
</script>

<body>
  <div id="viewerContent"></div>
</body>
```

10.2.2 Activation

After purchasing the product, you should activate license for used components. There are several ways of license key connection.

Activation using code string

To activate using a string, you should copy an encrypted license text from your personal account and register it in the event handler. You should place the code until you create and render your report.

index.php

```
<?php
  $handler = new \Stimulsoft\StiHandler();
  $handler->license->setKey('Your activation code...');
  $handler->renderHtml();
?>
```

© 2003-2024 Stimulsoft

There is an option to load a license key using JavaScript code on the project page. You should place the code until you create and render your report.

index.php

```
Stimulsoft.Base.StiLicense.Key = "Your activation code...";
```

Activation using a file

To activate using a license file you should download the `license.key` file from your personal account, copy it to the folder of the PHP project, and register it in the event handler.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setFile('license.key');
   $handler->renderHtml();
?>
```

There is an option to load a license key using JavaScript code on the project page. You should place the code until you create and render your report.

index.php

```
Stimulsoft.Base.StiLicense.loadFromFile("license.key");
```

Protection against license key theft

If you activate license using a string, you can add its code by a certain condition. For example, you want to add a license key for registered users.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   if (!empty($sessionID))
```

© 2003-2024 Stimulsoft

```
    $handler->license->setKey('Your activation code...');
  $handler->renderHtml();
?>
```

Also, we recommend you change the location and name of the license key file, for example:

license.php

```
<?php
  $handler = new \Stimulsoft\StiHandler();
  $handler->license->setFile('private/a15fc0ef64e6.key');
  $handler->renderHtml();
?>
```

10.2.3 Showing Reports and Dashboards

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `report.render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer.

To display a report in the viewer you should create the `StiReport` object, load a report template to it, and assign a received object to the viewer. All other actions will be done automatically, the viewer will render a report and display the first page.

viewer.php

```
<?php
  $report = new \Stimulsoft\Report\StiReport();
  $report->loadFile('reports/SimpleList.mrt');

  $viewer = new \Stimulsoft\Viewer\StiViewer();
  $viewer->report = $report;
  $viewer->renderHtml();
?>
```

© 2003-2024 Stimulsoft

The viewer can render and display both report templates and documents (rendered reports). You can find more detailed information with various report formats and documents in the Loading and Saving Reports chapter.

10.2.4 Localization

The viewer supports full localization of its interface. To localize the interface into the language you need, you should set the required file name for the `localization` option of the viewer:

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->localization = 'de.xml';

   $viewer = new \Stimulsoft\Viewer\StiViewer($options);
   $viewer->renderHtml();
?>
```

All available localization XML files are located in the resources of the installed product package. If necessary, you may download the localization file from any other location. You need to specify the full path to the XML file for the `localization` option:

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->localization = '/resources/loc/de.xml';

   $viewer = new \Stimulsoft\Viewer\StiViewer($options);
   $viewer->renderHtml();
?>
```

10.2.5 Printing Report

There are several variants of report print in the viewer. Each of them has its features, advantages and disadvantages.

Print to PDF

Print will be done using report export to PDF format. The advantages include excellent accuracy of location and report elements print in comparison with other

© 2003-2024 Stimulsoft

print variants. The disadvantages include a set plug in the browser to view PDF files (modern browsers have an embedded tool for viewing and printing PDF files).

Print with Preview
A report will be printed in a separate pop-up browser to HTML format. A report can be previewed and sent it to the printer or copy to another place as a text or an HTML code. The disadvantages include cross browser capability when printing, a lack of need for setting special plug ins. A disadvantage is a relatively low accuracy of report elements location, due to features of HTML formatting implementation.

Print without Preview
A report will be printed directly to the printer without preview. After you, select this menu item, the system print dialog is displayed. Therefore, as print in this mode is carried out in HTML format, print quality is similar to report print quality with preview.

Information

A report is printed using embedded methods of a used browser. That's why the presentation of the dialog window may differ in various operating systems and browsers. Also, a browser allow you to control print settings from the JavaScript code, so you should make required settings in the dialog window.

Settings
When selecting report print the menu with the selection of print variant on the viewer panel. The component has the ability to force set the required print mode. To do it you should set the `printDestination` property to one of the specified values below from the `StiPrintDestination` enumeration.

Name	Description
`StiPrintDestination::Default`	When selecting print the menu with available variants of print will be displayed (a value of a property by default)
`StiPrintDestination::Pdf`	Print to PDF format.

© 2003-2024 Stimulsoft

`StiPrintDestination::Direct`	Print to HTML format directly to the printer, the system print dialog will be displayed.
`StiPrintDestination::WithPreview`	Print to HTML format with preview in a pop-up window.

For example, you should set print mode only in PDF format.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->printDestination = \Stimulsoft\Viewer
   \StiPrintDestination::Pdf;
?>
```

The viewer has an ability to fully disable report print, if it's not required. To do it you should set the `false` value for the `ShowPrintButton` property.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->showPrintButton = false;
?>
```

Print events

If you need to make some actions before report print, you can use the `onPrintReport` event. In arguments of the event, the type of report print and a report will be transferred, sent to print.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onPrintReport = 'onPrintReport';
   $viewer->renderHtml();
```

© 2003-2024 Stimulsoft

```
?>

function onPrintReport(args) {
   var printAction = args.printAction;
   var report = args.report;
}
```

To make the print work on the PHP server-side, you should add the invoke of the event handler on the client-side and define the `onPrintReport` event in the file of event handler with the same name on the server-side. You can find a detailed description of event work in the PHP Events Handler chapter.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onPrintReport = true;
   $viewer->renderHtml();
?>
```

handler.php

```
$handler->onPrintReport = function ($args) {
   $fileName = $args->fileName;
   $printAction = $args->printAction;

   return \Stimulsoft\StiResult::success();
};
```

You can find a detailed description of available argument values in the Viewer Events chapter.

Printing from a Code

There is an ability, which allows you to print a report from a code not using the viewer functions. You can find a detailed description of this functionality in the Printing Report from Code chapter.

© 2003-2024 Stimulsoft

10.2.6 Report Export

The viewer allows you to export a displayed report to PDF, PowerPoint, HTML, Text, Word, ODT Excel, ODC, CSV, SVG formats. A dashboard can be converted to PDF, Excel, HTML formats. To make the export work, additional settings of the viewer are not required.

Begin Export event

If you need to make some actions before export a report, you can use the `onBeginExportReport` event. The event will be invoked after the dialog export window with settings displayed. In arguments of the event the type of report export, a report and all settings of selected export will be transferred.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onBeginExportReport = 'onBeginExportReport';
   $viewer->renderHtml();
?>

function onBeginExportReport(args) {
   var format = args.format;
   var fileName = args.fileName;
   var settings = args.settings;
}
```

The `format` property contains the type of selected export in the menu. It can take one of the values of the `Stimulsoft.Report.StiExportFormat` enumeration. You can find a detailed description of available values in the Export Report from Code chapter.

The `fileName` property contains file's name to save after export finished. The `settings` property contains all available settings for the current export type. Export settings and the name of a saved file can be changed, new values will be used automatically to export a report. For example, you need to change quality and image resolution for PDF export.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
```

© 2003-2024 Stimulsoft

```
   $viewer->onBeginExportReport = 'onBeginExportReport';
   $viewer->renderHtml();
?>

function onBeginExportReport(args) {
   if (args.format == Stimulsoft.Report.StiExportFormat.Pdf) {
      args.settings.imageQuality = 0.90;
      args.settings.imageResolution = 200;
   }
}
```

To get access to export settings on the PHP server-side you should add the invoke of the event handler on the client-side and define the `onBeginExportReport` event of the same name in the file of the event handler. You can find a detailed description of the event work in the PHP Events Handler chapter.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onBeginExportReport = true;
   $viewer->renderHtml();
?>
```

handler.php

```
$handler->onBeginExportReport = function ($args) {
   $fileName = $args->fileName;
   $format = $args->format;
   $settings = $args->settings;

   return \Stimulsoft\StiResult::success();
};
```

Export settings and the name of a saved file can be changed to export a report. For example, you need to add a prefix for the file name and change quality and image resolution for PDF export.

handler.php

```
$handler->onBeginExportReport = function ($args) {
```

© 2003-2024 Stimulsoft

```
    $args->fileName = 'PHP-' . $args->fileName;

    if ($args->format == \Stimulsoft\StiExportFormat::Pdf) {
        $args->settings->imageQuality = 0.90;
        $args->settings->imageResolution = 200;
    }

    return \Stimulsoft\StiResult::success();
};
```

You can find a detailed description of available argument values in the Viewer Events chapter.

End Export event

If you need to make some actions after export a report before its saving, you should use the `onEndExportReport` event. In arguments of the event the type of report export and name and byte data of an exported file.

viewer.php

```
<?php
    $viewer = new \Stimulsoft\Viewer\StiViewer();
    $viewer->onEndExportReport = 'onEndExportReport';
    $viewer->renderHtml();
?>

function onEndExportReport(args) {
    var format = args.format;
    var fileName = args.fileName;
    var data = args.data;
}
```

If you need, the name of a saved file or byte data can be changed, new values will be automatically used when saving a file.

By default, when exporting a file is saved to the computer local disk. To transfer an exported file to the PHP server-side you should add the invoke of events on the client-side in the `onEndExportReport` event, and on the server-side you should define the `onEndExportReport` event of the same name in the file of event handler. You can find a detailed description of event work in the PHP Events Handler

© 2003-2024 Stimulsoft

chapter.

viewer.php

```
viewer.onEndExportReport = function (args) {

   // Calling the server-side handler
   Stimulsoft.Helper.process(args);
}
```

handler.php

```
$handler->onEndExportReport = function ($args) {
   $fileName = $args->fileName;
   $format = $args->format;
   $data = $args->data;

   return \Stimulsoft\StiResult::success();
};
```

In this case, after exporting, the obtained byte data will be transferred to the PHP server-side. Saving on the local computer will be disabled. For example, you want to save all exported files on the server side to the `reports` folder, and display a message after successfully saving the file:

handler.php

```
$handler->onEndExportReport = function ($args) {
   $reportName = $args->fileName . '.' . $args->fileExtension;
   file_put_contents('reports/' . $reportName, base64_decode($args->data));

   return \Stimulsoft\StiResult::success("The exported report is saved
   successfully as $reportName");
};
```

Information

Byte data is transferred to the server-side in the `Base64` coding, so before saving you should convert them into original byte stream using, for example, the `base64_decode()` standard PHP function.

© 2003-2024 Stimulsoft

You can find a detailed description of available argument values in the Viewer Events chapter.

Export Settings

Sometimes you should disable not used report export formats and leave only used. It allows you to load interface and simplify using the viewer. To disable not used formats of exports you should set the `false` value for corresponding properties of the viewer, for example.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->exports->showExportToDocument = false;
   $options->exports->showExportToWord2007 = false;
   $options->exports->showExportToCsv = false;
?>
```

Also, if needed, you can completely remove the display of dialog export windows, export will always be carried out with settings by default. In this case, settings can be controlled in export event. To disable dialog windows you should set the `false` value for the `showExportDialog` property.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->exports->showExportDialog = false;
?>
```

A full list of available options is in the Viewer Settings chapter.

Export Report from Code

© 2003-2024 Stimulsoft

In addition, report export is available with the help of a code. The `exportDocument()` method of report object is intended for this. You can find a detailed description in the Export Report from Code chapter.

10.2.7 Viewing Modes

The viewer allows you to set various modes of interface display and report pages. In addition, you can control the display on mobile devices.

Scroll Bars

There are two modes of report display in the viewer: with scroll bars and without them. Viewing mode without scroll bars is set by default. To enable viewing mode with scroll bars you should set the `true` value for the `scrollbarsMode` property.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->appearance->scrollbarsMode = true;
?>
```

In the first mode (without scroll bars) the viewer displays the entire page or report. If sizes are specified by width and height, the viewer will crop the page that exceeded page boundaries. In the second mode, in comparison with the first, when the page will not be cropped if it's outside the viewer dimension. Instead of this scroll bars will appear with the help of them you can view a page or a report completely.

Information

In the mode of report viewing with scroll bars, you should set the viewer height or the height with 650 pixels will be set by default.

Full screen mode

There is the mode of full screen display of a report and dashboard in the viewer. Standard viewing mode is enabled by default, the viewer has specified sizes in the settings. To enable full screen mode of viewing you should set the true value for

© 2003-2024 Stimulsoft

the `fullScreenMode` property.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->appearance->fullScreenMode = true;
?>
```

Also, to enable or disable the full screen mode, you can use the corresponding button on the viewer control panel.

Display report pages

There are three modes of report display: single page, the entire report as a ribbon and table display of report pages. To control modes you should use the `viewMode` property, which takes one of the following values.

Name	Description
`StiWebViewMode::SinglePage`	One page, selected on the control panel is displayed.
`StiWebViewMode::Continuous`	All report pages are displayed as a ribbon.
`StiWebViewMode::MultiplePages`	All report pages are displayed as a table.

For example, you need to set the mode of display all pages as a ribbon.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->viewMode = \Stimulsoft\Viewer
   \StiWebViewMode::Continuous;
?>
```

© 2003-2024 Stimulsoft

Mobile mode

The viewer supports the work both with a simple computer, touch screens and mobile devices. To control interface modes you should use the `interfaceType` property, which takes one of the following values.

Name	Description
`StiInterfaceType::Auto`	The viewer interface will be selected automatically depending on the device you use (the value by default).
`StiInterfaceType::Mouse`	Forced using of standard interface to control the viewer using a computer mouse.
`StiInterfaceType::Touch`	Forced using of the `Touch` interface to control the viewer using touch screen of a monitor. In this mode interface elements of the viewer have enlarged sizes for convenient control.
`StiInterfaceType::Mobile`	Forced using of the `Mobile` interface to control the viewer using smartphone screen. In this mode, the viewer interface is simplified and adapted to control using a mobile device.

For example, you need to completely disable mobile display mode.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->appearance->interfaceType = \Stimulsoft\Viewer
   \StiInterfaceType::Mouse;
?>
```

© 2003-2024 Stimulsoft

10.2.8 Work with Report Variables

To work with report variables in the viewer you should use the support of special control panel. To add a parameter to the panel you should define the variable, which is requested from a user. When viewing a report in the viewer, this variable will be automatically added to the parameter panel. All types of report variables are supported (simple variables, date and time, range, lists, etc).

Control of variables on the parameter panel

To make some actions before assign parameters you should use the `onInteraction` event, which will be invoked when making interactive actions of the viewer. In arguments of the event the type of action and the collection of variables and their values, located on the parameter panel will be transferred. The type of action in this case will have the `Variables` value.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onInteraction = 'onInteraction';
   $viewer->renderHtml();
?>

function onInteraction(args) {
   if (args.action == "Variables") {
      var variables = args.variables;
   }
}
```

The collection of variables is the object, which contains all variables of the parameter panel and their values.

viewer.php

```
var variables = {
   VariableString: "Text value",
   VariableInt: 20
}
```

You can change values of variables, and the type of a new value should correspond to the type of the changed variable. You can find a detailed description of available

© 2003-2024 Stimulsoft

argument values in the Viewer Events chapter.

Setting the parameter panel

If the work with variables in the viewer is not required, you can completely disable this feature. To do that you should use the `showParametersButton` property, which requires the setting of the `false` value.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->showParametersButton = false;
?>
```

Information

When such configuration of the viewer is carried out, the parameter panel will not be displayed even if there are parameters in a displayed report.

Control of all report variables

If you need to control all report variables, you should use the `onPrepareVariables` event, which will be invoked before rendering a report.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onPrepareVariables = 'onPrepareVariables';
   $viewer->renderHtml();
?>

function onPrepareVariables(args) {
   var variables = args.variables;
}
```

You can control all variables both on the client-side and on the PHP server-side. You can find a detailed description in the Work with Report Variables in the engine

© 2003-2024 Stimulsoft

section.

10.2.9 Work with Bookmarks

There is the support of report bookmarks in the viewer. When displaying such report on the left side of the page, the panel with bookmarks will be displayed. When selecting a bookmark for your report, the viewer will automatically go to the page you need, and the report element with a bookmark will be highlighted.

Setting bookmarks

The width of the bookmark panel is 180 pixels by default. The viewer allows you to change this value. To do it you should use the `bookmarksTreeWidth` property, and its value is specified in pixels.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->appearance->bookmarksTreeWidth = 300;
?>
```

If the work with report bookmarks is not required, you can completely disable this feature. To do that, you should use the `showBookmarksButton` property, which needes to be set in the `false` value.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->showBookmarksButton = false;
?>
```

Information

In this case, report bookmarks will not be displayed, even if they are present in a displayed report. This property doesn't exert influence over print and export a report.

© 2003-2024 Stimulsoft

When printing a report with bookmarks, the bookmarks tree will be hidden. If apart from the report it's required to print bookmarks too, you should set the `bookmarksPrint` property in the `true` value.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->appearance->bookmarksPrint = true;
?>
```

10.2.10 Dynamic Sorting, Collapsing, and Drill-Down

Apart from the variables, which values can be set on the parameter viewer. The viewer supports other kinds of interactivity, which add convenience and functionality when using the report generator. They are sorting, collapsing and detailing.

Sorting
Dynamic sorting is an ability, which allows you to change sorting direction in a rendered report. To do it, you should click on the component, which had a set dynamic sorting. Dynamic sorting is carried out in the following ways: **Ascending**, **Descending**. Each time when clicking on a component, the direction is reversed.

Multilevel sorting in a report is allowed. To do that, you should hold the **Ctrl** button and step by step click on sorted report components. To reset sorting you can click on any sorted component not holding the **Ctrl** button.

Collapsing
A report with dynamic collapsing is an interactive report, where definite blocks can collapse/expand their content when clicking on block header. Report elements, which can be collapsed/ expanded, they are marked with a special icon with a **[-]** or **[+]** sign.

Drill-Down
When detailing data, the panel of drill down with tabs of drilled reports will be displayed under the viewer toolbar. There is an ability, which allows you to close not required detailing pages at the moment.

The viewer interactivity event

© 2003-2024 Stimulsoft

To make dynamic sorting, collapsing and drill-down reports work you don't need any additional settings of the viewer. To make some actions before sorting, collapsing or drilling down a report you should use the `onInteraction` event, which will be invoked when interactive actions of the viewer are made. There is a definite type of action for each kind of the viewer interactivity.

Name	Description
`Sorting`	The action happens when using the sorting of columns.
`DrillDown`	The action happens when using the detailing of columns.
`Collapsing`	The action happens when using the collapsing of report blocks.
`Variables`	The action happens when using variables on the parameter panel. You can find a detailed description in the Work with Report Variables chapter.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onInteraction = 'onInteraction';
   $viewer->renderHtml();
?>

function onInteraction(args) {
   switch (args.action) {
      case "Sorting":
         break;

      case "DrillDown":
         break;

      case "Collapsing":
         break;
   }
}
```

In arguments, corresponding collections of the `sortingParameters`,

© 2003-2024 Stimulsoft

`collapsingParameters` and `drillDownParameters` parameters are transferred. They contain data in a special format, needed for the current interactive action.

viewer.php

```
var sortingParameters = {
   ComponentName: "Text10;false",
   DataBand: "DataBand1;DESC;CompanyName"
};

var collapsingParameters = {
   CollapsingStates: {
      GroupHeaderBand1: {
         keys: [1],
         values: [false]
   },
   ComponentName: "GroupHeaderBand1"
};

var drillDownParameters = [
   {
      ComponentIndex: "1"
      DrillDownMode: null
      ElementIndex: "6"
      PageGuid: "b916d048d3f446dc97c356d4ff47f48f"
      PageIndex: "0"
      ReportFile: null
   }
];
```

If needed, the values of parameter collection can be corrected, at the same time saving structure and order of transferred values. You can find a detailed description of available argument values in the Viewer Events chapter.

10.2.11 Editing Report

The viewer has an ability to edit elements of a rendered report, such as text fields and checkboxes. To make editing possible, you should mark the components you need as edited in report template. You don't need any additional settings of the viewer. After a report displayed in the viewer, you should click the corresponding button on the viewer toolbar to start editing.

Information

Edited values will be applied when printing or exporting a report and the original

© 2003-2024 Stimulsoft

report is intact. When the viewer is reloaded all values will be restored to the original.

10.2.12 Sending Report By Email

There is an ability to send a report by Email in the viewer. To activate this capability, you should set the `showSendEmailButton` event of the viewer in the `true` value and add the `onEmailReport` event handler.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->showSendEmailButton = true;

   $viewer = new \Stimulsoft\Viewer\StiViewer($options);
   $viewer->onEmailReport = true;
   $viewer->renderHtml();
?>
```

If necessary, you may add a call to a JavaScript event, in the arguments of which you can find the required data for sending an email, get the type of report export, get the report itself, get the report export settings and change them. You can find a detailed description of available argument values in the Viewer Events chapter.

When sending a report by Email the menu of attachment format selection, which corresponds to the menu of report export selection is displayed. When you select a format, the dialog of input sending parameters will be displayed, among them: email receiver, theme, and text of a Email. After the sending is confirmed the `onEmailReport` event mentioned above will be invoked, where you can check and correct data, input in this form:

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onEmailReport = 'onEmailReport';
   $viewer->renderHtml();
?>

function onEmailReport(args) {
   args.settings.subject = "Invoice: " + args.settings.subject;
```

© 2003-2024 Stimulsoft

```
}
```

Pure JavaScript doesn't have the work of functions with Email. There are functions for this capability, intended for sending on the PHP server-side. To use them, you should add the invoke of event handler on the client-side in the `onEmailReport` event. Also you should define the `onEmailReport` event of the same name on the server-side in the file of event handler. You can find a detailed description in the PHP Events Handler chapter.

viewer.php

```
<?php
   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onEmailReport = true;
   $viewer->renderHtml();
?>
```

handler.php

```
$handler->onEmailReport = function ($args) {
   $format = $args->format;
   $fileName = $args->fileName;
   $settings = $args->settings;

   return \Stimulsoft\StiResult::success();
};
```

To send Email on the PHP server-side, in arguments of the event you should transfer data, such as log in, account's password, from which the sending will be made and server settings: its address, port and other parameters. The `settings` property in arguments of the event is intended for this. It is an object of the `StiEmailSettings` class, which contains all necessary properties to set Email sending. You can find a detailed description of available settings in the Viewer Events chapter.

The sample of setting minimum required settings of the mail server to send a report.

handler.php

© 2003-2024 Stimulsoft

```
$handler->onEmailReport = function ($args) {
   $args->settings->from = 'mail.sender@stimulsoft.com';
   $args->settings->host = 'smtp.stimulsoft.com';
   $args->settings->port = 456;
   $args->settings->login = '********';
   $args->settings->password = '********';

   return \Stimulsoft\StiResult::success();
};
```

Sample of checking and changing theme of an Email before sending.

handler.php

```
$handler->onEmailReport = function ($args) {
   if (strlen($args->settings->subject) == 0)
      $args->settings->subject = $args->formatName . ' report '.$args-
      >settings->attachmentName;

   return \Stimulsoft\StiResult::success();
};
```

Sample of adding addresses of additional an Email receivers with a report.

handler.php

```
$handler->onEmailReport = function ($args) {
   $args->settings->cc[] = 'extra_recipient_one@stimulsoft.com';
   $args->settings->bcc[] = 'hidden_recipient_one@stimulsoft.com';
   $args->settings->bcc[] = 'hidden_recipient_two@stimulsoft.com John
   Smith';

   return \Stimulsoft\StiResult::success();
};
```

Email sending settings

The viewer allows you to set values by default for forms of Email sending. The `defaultEmailAddress`, `defaultEmailSubject` and `defaultEmailMessage` properties are intended for this.

© 2003-2024 Stimulsoft

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->showSendEmailButton = true;

   $options->email->defaultEmailAddress =
   'recipient_address@stimulsoft.com';
   $options->email->defaultEmailSubject = 'New Invoice';
   $options->email->defaultEmailMessage = 'Please check the new invoice in
   the attachment';
?>
```

10.2.13 Calling Designer from Viewer

The viewer has an ability to invoke the report designer. The **Design** button located on the viewer toolbox panel is intended for this. This button is disabled by default. To use this ability you should set the `showDesignButton` property in the `true` value and define the `onDesignReport` event.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->showDesignButton = true;

   $viewer = new \Stimulsoft\Viewer\StiViewer();
   $viewer->onDesignReport = 'onDesignReport';
   $viewer->renderHtml();
?>

function onDesignReport(args) {
   window.open("designer.php?fileName=" + args.fileName);
}
```

You can find a detailed description of available argument values in the Viewer Events chapter.

Information

The viewer doesn't run the designer, it just invokes the specified event and transfers it as arguments of file name and a viewed report. In the event you can redirect to the PHP page, where the report designer is located.

© 2003-2024 Stimulsoft

10.2.14 Appearance

The viewer has an ability to change themes of visual controls. To do this, set the `theme` property in the component options:

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->appearance->theme = \Stimulsoft\Viewer
   \StiViewerTheme::Office2022WhiteBlue;
?>
```

There are currently **8 themes** available with different color accents. As a result, **more than 60** variants of the appearance are available. This allows you to customize the appearance of the viewer for almost any design of the Web project.

Additional settings

By default, the viewer has only upper toolbox, where there are all elements of report control. If needed, the toolbox can be divided into upper and bottom. The print menu, report export, buttons of work with parameters and bookmarks will be on the upper panel. The bottom panel will contain elements for switching between report pages and the menu of zoom control. To enable a specified mode, you should use the `displayMode` property, which can have the following values.

Name	Description
`StiToolbarDisplayMode::Simple`	Simple display mode, all controls are located on one control panel (the value by default).
`StiToolbarDisplayMode::Separated`	Original is the Separate display mode, the control panel is divided into the upper one - interaction with a report and the bottom - interaction with pages.

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();
   $options->toolbar->displayMode = \Stimulsoft\Viewer
   \StiToolbarDisplayMode::Simple;
   $options->appearance->scrollbarsMode = true;
?>
```

Additionally it is possible to set design parameters of the main viewer elements. For example, you can change font and color of the viewer toolbar captions, set the viewer background, set color of page borders, etc. Below you can see the list of available properties, which change the viewer appearance and their values by default.

viewer.php

```
<?
   $options = new \Stimulsoft\Viewer\StiViewerOptions();

   $options->appearance->backgroundColor = 'white';
   $options->appearance->pageBorderColor = 'red';
   $options->appearance->showPageShadow = false;

   $options->toolbar->backgroundColor = 'aqua';
   $options->toolbar->borderColor = 'darkgreen';
   $options->toolbar->fontColor = 'white';
   $options->toolbar->fontFamily = 'Arial';
?>
```

One of the standard constants for HTML colors is specified as a value for color, or a color code in RGB format, for example, `'#ff2020'`.

10.2.15 Viewer Events

The viewer supports the events, which give an ability to make necessary operations before definite actions both on the JavaScript client-side and on the PHP server-side. To make events work on the client-side, you don't need any additional settings. To make events work on the server-side, you should add a special event handler to the page. You can find a detailed description of connection and setting the PHP event handler in the PHP Events Handler chapter.

The viewer supports the following events:

- onBeginProcessData

© 2003-2024 Stimulsoft

- onEndProcessData
- onPrepareVariables
- onPrintReport
- onBeginExportReport
- onEndExportReport
- onInteraction
- onEmailReport
- onDesignReport

onBeginProcessData
The event is invoked before data request, which needed to render a report. The list of arguments of the event is in the Engine Events chapter. You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL Data Adapters chapters.

onEndProcessData
The event is invoked after loading data before rendering a report. The list of the event arguments are in the Engine Events chapter. You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL Data Adapters chapters.

onPrepareVariables
The event is invoked before rendering a report after preparing report variables. The list of the event arguments is in the Engine Events chapter. You can find a detailed description and samples of using in the Work with Report Variables chapter.

onPrintReport
The event is invoked before printing a report. In the table below, you can find the list of the event handler arguments on the JavaScript client-side.

Name	Description

© 2003-2024 Stimulsoft

`event`	The identifier of the current event has the "`PrintReport`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer` • `Designer`
`report`	Current report object.
`printAction`	The type of report print. It can take the following values: • `StiPrintAction::PrintPdf` - Print to PDF format; • `StiPrintAction::PrintWithoutPreview` - Print to HTML format, directly to the printer. The system print dialog will be displayed; • `StiPrintAction::PrintWithPreview` - Print to HTML format with preview in a pop-up window.
`fileName`	Report file's name for saving.
`preventDefault`	This flag gives an ability to stop further event processing by the viewer. The `false` value is set by default.

In the table below you can find the list of the event handler arguments on the PHP server-side.

Name	Description
`sender`	The identifier of the component, which initiated this event can take the following values:

© 2003-2024 Stimulsoft

	• StiComponentType::Viewer • StiComponentType::Designer
printAction	Report print type. It can take the following values: • StiPrintAction::PrintPdf - Print to PDF format; • StiPrintAction::PrintWithoutPreview - Print to HTML format directly to the printer. System print dialog will be displayed; • StiPrintAction::PrintWithPreview - Print to HTML format with preview in a pop-up window.
fileName	Report file name for saving.

You can find a detailed description and samples of using in the Printing Report chapters.

onBeginExportReport

The event is invoked before exporting a report after the dialog of export settings. In the table below, you can see the list of the event handler argument on the JavaScript client-side.

Name	Description
event	The identifier of the current event has the "BeginExportReport" value.
sender	The identifier of the component, which initiated this event can take the following values: • Viewer • Designer

© 2003-2024 Stimulsoft

action	The action, which initiated export event can take the following values: • Stimulsoft.Viewer.StiExportAction.ExportReport • Stimulsoft.Viewer.StiExportAction.SendEmail
report	Current report object.
format	Selected report export format. It can take the following values: • Stimulsoft.Report.StiExportFormat.Document • Stimulsoft.Report.StiExportFormat.Pdf • Stimulsoft.Report.StiExportFormat.Xps • Stimulsoft.Report.StiExportFormat.Ppt2007 • Stimulsoft.Report.StiExportFormat.Html • Stimulsoft.Report.StiExportFormat.Html5 • Stimulsoft.Report.StiExportFormat.Text • Stimulsoft.Report.StiExportFormat.Word2007 • Stimulsoft.Report.StiExportFormat.Excel2007 • Stimulsoft.Report.StiExportFormat.Odt • Stimulsoft.Report.StiExportFormat.Ods • Stimulsoft.Report.StiExportFormat.Csv • Stimulsoft.Report.StiExportFormat.ImageSvg
formatName	Name of a selected report export format corresponds to the name of the constants in format list.
settings	Settings of a selected export format. List of available properties will depend on a selected export type.
fileName	Report file name for saving after export completed.
openAfterExport	The flag indicates that a report will be exported in a new browser tab (the true value) or after export completed file saving will be invoked (the false value).
preventDefault	This flag gives an ability to stop furher event processing by the viewer. The false value is set by default.

In the table below, you can see the list of event handler argument on the PHP

© 2003-2024 Stimulsoft

server-side.

Name	Description
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer` • `Designer`
`action`	The action, which invoked export event can take the following values: • `StiExportAction::ExportReport` • `StiExportAction::SendEmail`
`format`	Selected report export format. It can take the following values: • `StiExportFormat::Document` • `StiExportFormat::Pdf` • `StiExportFormat::Xps` • `StiExportFormat::Ppt2007` • `StiExportFormat::Html` • `StiExportFormat::Html5` • `StiExportFormat::Text` • `StiExportFormat::Word2007` • `StiExportFormat::Excel2007` • `StiExportFormat::Odt` • `StiExportFormat::Ods` • `StiExportFormat::Csv`

© 2003-2024 Stimulsoft

	• `StiExportFormat::ImageSvg`
`formatName`	Name of a selected report export corresponds to the name of constants in the format list.
`fileName`	Report file name for saving after export completed.
`settings`	Settings of a selected export format. List of available properties will depend on a selected export type.

You can find a detailed description and samples of using in the Report Export chapter.

onEndExportReport

The event is invoked after exporting a report till its saving as a file. In the table below, you can see the list of the event handler argument on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`EndExportReport`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer` • `Designer`
`report`	Current report object.
`format`	Selected report export format. It can take the following values: • `Stimulsoft.Report.StiExportFormat.Document`

© 2003-2024 Stimulsoft

	• Stimulsoft.Report.StiExportFormat.Pdf • Stimulsoft.Report.StiExportFormat.Xps • Stimulsoft.Report.StiExportFormat.Ppt2007 • Stimulsoft.Report.StiExportFormat.Html • Stimulsoft.Report.StiExportFormat.Html5 • Stimulsoft.Report.StiExportFormat.Text • Stimulsoft.Report.StiExportFormat.Word2007 • Stimulsoft.Report.StiExportFormat.Excel2007 • Stimulsoft.Report.StiExportFormat.Odt • Stimulsoft.Report.StiExportFormat.Ods • Stimulsoft.Report.StiExportFormat.Csv • Stimulsoft.Report.StiExportFormat.ImageSvg
formatName	Name of a selected report export corresponds to the name of constants in the format list.
data	Byte data of an exported report, prepared for saving to a file.
fileName	Report file name for saving after export completed.
openAfterExport	The flag indicates that a report will be exported in a new browser tab (the true value) or after export completed file saving will be invoked

© 2003-2024 Stimulsoft

	(the `false` value)
`preventDefault`	This flag gives an ability to stop further event processing by the viewer. The `false` value is set by default.

In the table below, you can see the list of the event handler argument on the PHP server-side.

Name	Description
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer` • `Designer`
`format`	Selected report export format. It can take the following values: • `StiExportFormat::Document` • `StiExportFormat::Pdf` • `StiExportFormat::Xps` • `StiExportFormat::Ppt2007` • `StiExportFormat::Html` • `StiExportFormat::Html5` • `StiExportFormat::Text` • `StiExportFormat::Word2007` • `StiExportFormat::Excel2007` • `StiExportFormat::Odt` • `StiExportFormat::Ods`

© 2003-2024 Stimulsoft

	• `StiExportFormat::Csv` • `StiExportFormat::ImageSvg`
`formatName`	Name of a selected report export corresponds to the name of constants in the format list.
`data`	Byte data of an exported report, prepared for saving to a file.
`fileName`	Report file name for saving after export completed.
`fileExtension`	Report file extension for saving after the export completed corresponds to the type of a selected format.

You can find a detailed description and samples of using in the Report Export chapter.

onInteraction

The event is invoked while interactive action of the viewer (dynamic sorting, collapsing, drill-down, applying of parameters) until processing values by the report generator. In the table below, you can see the list of the event handler argument on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`Interaction`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer` • `Designer`
`report`	Current report object.

© 2003-2024 Stimulsoft

`action`	The identifier of the current interactive action can take the following values: • `Sorting` - This action happens when using sorting columns; • `DrillDown` - This action happens when using sorting columns; • `Collapsing` - This action happens when using of collapsing report blocks.
`variables`	The collection of report variables and their values, set in the parameter panel.
`sortingParameters`	The collection of parameters, needed for dynamic report sorting.
`collapsingParameters`	The collection of parameters needed for dynamic report elements collapsing.
`drillDownParameters`	The collection of parameters, needed for report drilldown.

You can find a detailed description and samples of using in the Dynamic Sorting, Collapsing, and Drill-Down chapter.

onEmailReport

The event is invoked after exporting a report before sending it by Email. In the table below, you can see the list of the event handler argument on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`EmailReport`" value.
`sender`	The identifier of the component,

© 2003-2024 Stimulsoft

	which initiated this event can take the following values: • `Viewer` • `Designer`
`report`	Current report object.
`format`	Selected report export format. It can take the following values: • `Stimulsoft.Report.StiExportFormat.Document` • `Stimulsoft.Report.StiExportFormat.Pdf` • `Stimulsoft.Report.StiExportFormat.Xps` • `Stimulsoft.Report.StiExportFormat.Ppt2007` • `Stimulsoft.Report.StiExportFormat.Html` • `Stimulsoft.Report.StiExportFormat.Html5` • `Stimulsoft.Report.StiExportFormat.Text` • `Stimulsoft.Report.StiExportFormat.Word2007` • `Stimulsoft.Report.StiExportFormat.Excel2007` • `Stimulsoft.Report.StiExportFormat.Odt` • `Stimulsoft.Report.StiExportFormat.Ods` • `Stimulsoft.Report.StiExportFormat.Csv` • `Stimulsoft.Report.StiExportFormat.ImageSvg`
`formatName`	Name of a selected report export corresponds to the name of

© 2003-2024 Stimulsoft

	constants in the list of formats.
`data`	Byte data of an exported report, prepared for sending by Email.
`fileName`	Report file name for sending by Email.
`settings`	The object, which contains parameters, filled in the viewer dialog of sending a report by Email. The description of all parameters is in a separate table below. All the parameters must be filled (they can be checked and filled on the PHP server-side).

In the table below you can see the list of parameters of sending Email on the JavaScript client-side.

Name	Description
`email`	The Email address, which will receive an exported report.
`subject`	Email theme.
`message`	Email text.

In the table below, you can see the list of the event handler argument on the PHP server-side.

Name	Description
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer` • `Designer`

© 2003-2024 Stimulsoft

format	Selected report export format. It can take the following values: • StiExportFormat::Document • StiExportFormat::Pdf • StiExportFormat::Xps • StiExportFormat::Ppt2007 • StiExportFormat::Html • StiExportFormat::Html5 • StiExportFormat::Text • StiExportFormat::Word2007 • StiExportFormat::Excel2007 • StiExportFormat::Odt • StiExportFormat::Ods • StiExportFormat::Csv • StiExportFormat::ImageSvg
formatName	Name of a selected report export corresponds to the name of constants in the list of formats.
data	Byte data of an exported report, prepared for sending by Email.
fileName	Report file name for sending by Email.
settings	The object, which contains parameters, filled in the viewer dialog of sending a report by Email. The description of all parameters is in a separate table below.

© 2003-2024 Stimulsoft

In the table below you can see the list of parameters of sending Email on the PHP server-side:

Name	Description
`from`	Email address of a Email sender.
`name`	Name and surname of a Email sender.
`to`	The email address which will receive an exported report is transferred from the viewer dialog window.
`subject`	Email theme is transferred from the viewer dialog window.
`message`	Email text is transferred from the viewer dialog window.
`attachmentName`	Report name in attachment. Report file name is used by default.
`charset`	The coding, used for Email text. The "UTF-8" is used by default.
`host`	The SMPT server is mandatory to fill in.
`port`	The SMPT port server. The 465 is used by default.
`secure`	The type of connection encryption with mail server, the "ssl" can be used (by default) or "tls" encryption.
`login`	The log in of connection to the mail server is mandatory to fill in.
`password`	Password for connection to the mail server is mandatory to fill in.
`cc`	The CC array (Carbon Copy) addresses of secondary Email recipients.

© 2003-2024 Stimulsoft

`bcc`	The BCC array (Blind Carbon Copy) of addresses of hidden Email recipients.

You can find a detailed description and samples of using in the Sending Report by Email chapter.

onDesignReport

The event is invoked when clicking the **Design** button in the viewer panel. In the table below you can see the list of the event handler argument on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`DesignReport`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Viewer`
`report`	Current report object.
`fileName`	Report file name for sending and loading to the designer.

You can find a detailed description and samples of using in the Calling Designer from the Viewer chapter.

10.2.16 Viewer Settings

The viewer is set using the properties from the `StiViewerOptions` class. All the properties are divided into the groups for comfortable using. All classes and enumerations of the viewer are located in the `\Stimulsoft\Viewer` namespace. To set the viewer, you should create the class of options, set required properties, and transfer an object of options as the first argument to the viewer constructor.

© 2003-2024 Stimulsoft

viewer.php

```
<?php
   $options = new \Stimulsoft\Viewer\StiViewerOptions();

   $options->appearance->theme = \Stimulsoft\Viewer
   \StiViewerTheme::Office2022WhiteBlue;
   $options->appearance->reportDisplayMode = \Stimulsoft\Viewer
   \StiHtmlExportMode::FromReport;
   $options->width = '1000px';
   $options->height = '1000px';
   $options->toolbar->displayMode = \Stimulsoft\Viewer
   \StiToolbarDisplayMode::Separated;
   $options->appearance->fullScreenMode = true;
   $options->appearance->scrollbarsMode = true;
   $options->exports->ShowExportToWord2007 = false;
   $options->exports->showExportToCsv = false;

   $viewer = new \Stimulsoft\Viewer\StiViewer($options);
   $viewer->renderHtml();
?>
```

Main settings (without groups)

Name	Description
width	Sets the width of the component in “px” or “%”. The "100%" value is set by default.
height	It sets height of a component in "px" or "%". The "100%" value is set by default for standard mode and "650px" for the mode of display with scroll bars.
localization	Sets the selected localization of the component. By default, the English localization is set. It is built into the component.

Appearance

© 2003-2024 Stimulsoft

Name	Description
theme	Specifies the theme of the viewers' layout. The list of available themes can be found in the StiViewerTheme enumeration. The default value is Office2022WhiteBlue.
backgroundColor	Sets the background color of the viewer. By default. it is set to 'white'.
pageBorderColor	Sets the border color of the viewer. By default, it is set to 'gray'.
rightToLeft	Sets the **Right to Left** mode for viewer controls. By default, the property is set to false.
fullScreenMode	It sets the full screen mode of the viewer display. If the property is set in the true value, the values of **width** and **height** properties are ignored. The false value is set by dedault.
scrollbarsMode	Sets the preview mode with scrollbars. By default, the property is set to false.
openLinksWindow	Sets the target window to open links contained in the report. By default, it is set to '_blank' (new window). It can have one of the standard values '_blank', '_self', '_top', as well as the name of the window or frame.
showTooltips	Enables displaying tips for the viewer controls when the mouse hovers over. By default, the property is set to true.

© 2003-2024 Stimulsoft

showTooltipsHelp	Sets a value which indicates that show or hide the help link in tooltips. By default, the property is set to true.
showDialogsHelp	Sets a value which indicates that show or hide the help button in dialogs. By default, the property is set to true.
pageAlignment	Sets the position of the report page in the viewer window: • StiContentAlignment::DefaultValue - page alignment is determined from the template settings (value is set by default); • StiContentAlignment::Left – the page will be aligned left; • StiContentAlignment::Center – the page will be centered (set by default); • StiContentAlignment::Right – the page will be aligned right.
showPageShadow	Enables displaying shadow for report pages. By default, the property is set to true.
bookmarksPrint	Enables printing of report bookmarks (besides the report itself). By default, the property is set to false.
bookmarksTreeWidth	Sets the width of the bookmarks panel in pixels. By default, the width is 180 pixels.
parametersPanelPosition	It sets location of the panel parameters in the viewer: • StiParametersPanelPosition::FromReport - the location of the panel is determined from

© 2003-2024 Stimulsoft

	the template settings (value is set by default); • `StiParametersPanelPosition::Top` - the panel is located upper report page; • `StiParametersPanelPosition::Left` - the panel is located to the left from report page.
`parametersPanelMaxHeight`	It sets max height of the parameter panel in pixels. The 300 value is set by default.
`parametersPanelColumnsCount`	It sets the number of columns in the parameter panel. The 2 value is set by default.
`parametersPanelDateFormat`	It sets date and time format for the variables, which are displayed in the parameter panel. The `String.empty` value is set by default.
`parametersPanelSortDataItems`	It sets or disables the sorting variable values mode. The option is set in the `true` value by default, i.e variable values are sorted.
`interfaceType`	It sets the type of the viewer interface. The following values can be used: • `StiInterfaceType::Auto` – the type of the viewer interface will be selected automatically depending on the device you use (value is set by default); • `StiInterfaceType::Mouse` – forced using of standard interface to control the viewer using a computer mouse; • `StiInterfaceType::Touch` – forced using the `Touch`

© 2003-2024 Stimulsoft

	interface to control the viewer using touch screen of a monitor. In this mode, the viewer interface elements have enlarged sizes for comfortable control; • `StiInterfaceType::Mobile` – forced using the `Mobile` interface to control the viewer using smartphone screen. In this mode, the viewer interface has a simplified appearance to control using a mobile device.
`allowMobileMode`	Enables or disables displaying a report or dashboard in the mobile mode. If the option is set to `false`, then the mobile view will not be used. If the option is set to `true`, the mobile view mode will be used when opening the viewer on mobile devices. By default, the option is set to `true`.
`chartRenderType`	It sets the type of chart drawing in a report: • `StiChartRenderType::AnimatedVector` – charts will be drawn in the vector mode with animation (value is set by default); • `StiChartRenderType::Vector` – charts will be drawn as a vector image without animation.
`reportDisplayMode`	It sets the export mode to display report pages. It can take one of the following values: • `StiHtmlExportMode::FromReport` - the export mode of the report elements is defined from

© 2003-2024 Stimulsoft

	report template settings - Div or Table (value is set by default); • `StiHtmlExportMode::Table` – report elements are exported using HTML tables; • `StiHtmlExportMode::Div` – report elements are exported using DIV markup.
`datePickerFirstDayOfWeek`	It gives an ability to set the first day of the week for the **Date Picker** tool: • `StiFirstDayOfWeek::Auto` - Monday or Sunday will be set as the first day of the week depending on browser culture (value is set by default); • `StiFirstDayOfWeek::Monday` - Monday will be set as the first day of the week; • `StiFirstDayOfWeek::Sunday` - Sunday will be set as the first day of the week.
`datePickerIncludeCurrentDayForRanges`	It gives an ability to include or not the current day into the range of the **Date Picker** element values. By default, the option is set in the `false` value i.e. the current day is not included into the range of the element values.
`allowTouchZoom`	It gives an ability change the viewer zoom by touching. By default, the option is set in the `true` value.
`combineReportPages`	It allows you to combine processed pages of report template into one template or present each page of the template as a separate tab in the viewer. By default, the option is set in the `false` value i.e. each page of report template will be presented as

© 2003-2024 Stimulsoft

	a separate tab in the viewer.

Toolbar

Name	Description
`visible`	It allows you to display or not to display the viewer toolbar. By default, the `true` value is set.
`displayMode`	It sets the display of the viewer toolbar. It can take one of the enumeration values below: • `StiToolbarDisplayMode::Simple` – simple display mode, all elements of control are located in one control panel (value is set by default); • `StiToolbarDisplayMode::Separated` – separated display mode, toolbar is divided into upper and bottom.
`backgroundColor`	It allows you to change the color of toolbar. The `'transparent'` value is set by default.
`borderColor`	It allows you to change toolbar border color. The `'transparent'` value is set by default.
`fontColor`	It gives an ability to change the font of all elements in the toolbar and in all menus of this panel. The `'transparent'` value is set by default.
`fontFamily`	It allows you to change the font for all elements in the toolbar and in all menus of this panel. By default, the `'Arial'` value is set.

© 2003-2024 Stimulsoft

`alignment`	It sets the alignment of elements in the control panel: • `StiContentAlignment::Default` – alignment of elements depends on the `RightToLeft` option (value is set by default); • `StiContentAlignment::Left` – all elements will be aligned to the left side of the toolbar; • `StiContentAlignment::Center` – all elements will be aligned to the center of the toolbar; • `StiContentAlignment::Right` – all elements will be aligned to the right side of the toolbar.
`showButtonCaptions`	It enables or disables the display of the viewer toolbar buttons text. By default, the property is set to `true`.
`showPrintButton`	It allows you to display or not to display the **Print** button. By default, the property is set to `true`.
`showOpenButton`	It enables the display of the **Open** button in the viewer toolbar when viewing reports or dashboards. By default, the property is set to `true`.
`showSaveButton`	It enables the display of the **Save** button in the toolbar when viewing reports or dashboards. By default, the property is set to `true`.
`showSendEmailButton`	It allows you to display or not to display the **Send Email** button in the toolbar. By default, the `false` value is set. Also, you should add the `onEmailReport` event handler.
`showFindButton`	It allows you to display or not to display the **Find** button in the

© 2003-2024 Stimulsoft

	toolbar. By default, the property is set to `true`.
`showBookmarksButton`	It allows you to display or not to display the **Bookmarks** button in the toolbar. If this button is not displayed, the bookmark panel, the bookmark panel will not be displayed in a report. By default, the property is set to `true`.
`showParametersButton`	It allows you to display or not to display the **Parameters** button in the toolbar. If this button is not displayed, the parameter panel will not be displayed in a report. By default, the property is set to `true`.
`showResourcesButton`	It allows you to display or not to display the **Resources** button in the toolbar. If this button is not displayed, the resources panel will not be displayed in a report. By default, the property is set to `true`.
`showEditorButton`	It allows you to display or not to display the **Editor** button in the toolbar. If this button is not displayed, you won't be able to change edited data. By default, the property is set to `true`.
`showFullScreenButton`	It enables the display of the **Full Screen** button in the viewer toolbar when viewing reports or dashboards. By default, the property is set to `true`.
`showRefreshButton`	It allows you to display or not to display the **Refresh** button in the viewer toolbar when viewing dashboards. By default, the property is set to `true`.

© 2003-2024 Stimulsoft

showFirstPageButton	It allows you to display or not to display the **First Page** button in the toolbar. By default, the property is set to `true`.
showPreviousPageButton	It allows you to display or not to display the **Previous Page** in the toolbar.By default, the property is set to `true`.
showCurrentPageControl	It allows you to display or not to display an indicator of the current page in the toolbar.By default, the property is set to `true`.
showNextPageButton	It allows you to display or not to display the **Next Page** button in the toolbar. By default, the property is set to `true`.
showLastPageButton	It allows you to display or not to display the **Last Page** button in the toolbar. By default, the property is set to `true`.
showZoomButton	It allows you to display or not to display the **Zoom** selection button in the toolbar. By default, the property is set to `true`.
showViewModeButton	It allows you to display or not to display the report pages display modes button. By default, the property is set to `true`.
showDesignButton	It enables the display of the **Design** button in the viewer toolbar when viewing reports or dashboards. By default, the property is set to `false`.
showAboutButton	It allows you to display or not to display the **About** button. By default, the property is set to `true`.

© 2003-2024 Stimulsoft

showPinToolbarButton	It allows you to display or not to display the **Pin** button in the mobile mode of report viewing. By default, the property is set to `true`.
printDestination	It sets the report print mode. It can take one of the enumeration values below: • `StiPrintDestination::Default` – the menu with the selection print mode will be displayed (value is set by default); • `StiPrintDestination::Pdf` – print will be made in PDF format; • `StiPrintDestination::Direct` – print will be made in HTML format directly to the printer. System print dialog will be displayed; • `StiPrintDestination::PopupWindow` – print will be made in HTML format via the pop-up window of report preview.
viewMode	It sets the report pages display mode: • `StiWebViewMode::OnePage` – one page selected in the toolbar is displayed (value is set by default); • `StiWebViewMode::Continuous` – all report pages are displayed as a ribbon; • `StiWebViewMode::MultiplePages` – all report pages are displayed as a table.
zoom	It allows you to set the scale of report pages when loading the

© 2003-2024 Stimulsoft

	viewer. 100 percent by default. Max value is 500 percent. • `StiZoomMode::PageWidth` – report pages scale by page width; • `StiZoomMode::PageHeight` – report pages scale by page height.
`menuAnimation`	It allows you to enable or disable the animation of display and closing various menus in the viewer. By default, the property is set to `true`.
`showMenuMode`	It sets the mode opening of various menus in the viewer when hovering or clicking. • `StiShowMenuMode::Click` – click-to-open menu mode (value is set by default); • `StiShowMenuMode::Hover` – hover-to-open menu mode.
`autoHide`	It sets the mode of automatic collapsing the toolbar when viewing a report in the mobile mode. By default, the property is set to `true`.

Exports

Name	Description
`storeExportSettings`	It allows you to save export settings in cookies. The `true` value is set by default.
`showExportDialog`	It allows you to display or not to display the **Export Settings** menu. If the menu is hidden, the export will be made with values by default. The `true` value is set by default.

© 2003-2024 Stimulsoft

showExportToDocument	It allows you to display or not to display the **Document File** element in the **Save** menu. The `true` value is set by default.
showExportToPdf	It enables the display of the **Adobe PDF File** export menu item when viewing reports and the **Adobe PDF** when viewing dashboards. The property has the `true` value by default.
showExportToXps	It enables the display of the **XPS File** export menu item. The property has the `true` value by default.
showExportToPowerPoint	It enables the display of the **Microsoft PowerPoint 2007/2010 File** export menu item. The property has the `true` value by default.
showExportToHtml	It allows you to display or not to display the **HTML File** element in the **Export Settings** menu and the **HTML** when viewing dashboards.. The `true` value is set by default.
showExportToHtml5	It allows you to display or not to display the **HTML5 File** element in the **Export Settings** menu. The `true` value is set by default.
showExportToText	It enables the display of the the **Text File** export menu item. The property has the `true` value by default.
showExportToWord2007	It allows you to display or not to display the **Microsoft Word 2007/2010 File** element in the **Save** menu. The `true` value is set by default.
showExportToOpenDocumentWriter	It enables the display of the **OpenDocument Writer File** export

© 2003-2024 Stimulsoft

	menu item. The property has the `true` value by default.
`showExportToExcel2007`	It enables the display of the menu **Microsoft Excel 2007/2010 File** export menu item when viewing reports and the **Microsoft Excel** item when viewing dashboards. The property has the `true` value by default.
`showExportToOpenDocumentCalc`	It enables the display of the **OpenDocument Calc File** export menu item. The property has the `true` value by default.
`showExportToCsv`	It enables the display of the **CSV File** export menu item. The property has the `true` value by default.
`showExportToJson`	It enables the display of the **JSON File** export menu item. The property has the `true` value by default.
`showExportToImageSvg`	It enables the display of the **Image** export menu item. The property has the `true` value by default.

Email

Name	Description
`showEmailDialog`	It enables the display of the parameter dialog window of report sending by Email. If the dialog window is disabled, the sending by Email will be done with the settings specified by default `onEmailReport`. The `true` value is set by default.
`showExportDialog`	It enables the display of the

© 2003-2024 Stimulsoft

	parameter dialog window when sending an Email. If the property has the false value, the export will be done with specified by default settings. The `true` value is set by default.
`defaultEmailAddress`	It sets an Email recipient by default, i.e. the address which will receive a Email with an attached report. The value is not set by default.
`defaultEmailSubject`	It sets the theme (header) of an Email by default. The value is not set by default.
`defaultEmailMessage`	It sets a message (text) of an Email by default. The value is not set by default.

10.3 HTML5 Designer

The report designer is an HTML5 component (`StiDesigner`) intended for developing reports in a browser window on any computer with any set operating system. The designer supports a lot of themes of design, animated interface and there are a lot of components to render a report of almost any complexity.

The designer has the convenient mode of report preview in the embedded viewer. This accelerates report development. The work the embedded viewer is almost similar to the work of a separate component. This nuance is considered more detailed in the HTML5 Viewer section.

The designer uses the AJAX technology to make all actions (loading and rendering a report, the work with components and their settings, report preview, print, export, etc.). It allows you to get rid of reloading of the entire system and increase the speed of work. The JavaScript technology applied to render reports, allows you to use almost any server-side of low performance.

Information

Due to the fact that an identical unified MRT template, the methods for loading a

© 2003-2024 Stimulsoft

template and work with data are used for dashboards and reports, in the documentation text the word «a report» will be used.

- Deployment
- Activation
- Creating and Editing Reports
- Saving Reports
- Localization
- Preview
- Appearance
- Designer Events
- Designer Settings

10.3.1 Deployment

Samples

You can find a full code of on the GitHub.

To use the product, you should download the ZIP archive of the product from the Downloads page of our site, unpack it, and copy the contents of the **/PHP** folder to your Web server. This folder is a Web project that contains all the necessary files and resources for the product and samples for working with the viewer and designer.

To install the product in an existing project, you should copy the /vendor folder from the **/PHP** folder to the root directory of the project or use the Composer dependency manager by executing the following console command:

console

```
composer require stimulsoft/reports-php
```

When working with the product, in most cases, it is enough to use only PHP code, which ensures the operation of all the main features. For a more detailed

© 2003-2024 Stimulsoft

configuration of the product and the use of all features, you must use JavaScript code. Next, two options for using certain features will be described - using **PHP** and **JavaScript** functions. Depending on your needs, you may use either the first or the second option.

To use the HTML5 component the designer (`StiDesigner`) in the project you should add specified scripts and styles to the `<head>` block on the PHP page. You can do this using the `StiJavaScript` **PHP** class. In the constructor, you need to specify the type of component to deploy:

designer.php

```
<head>
   <?php
      $js = new \Stimulsoft\StiJavaScript(\Stimulsoft
      \StiComponentType::Designer);
      $js->renderHtml();
   ?>
</head>
```

To edit dashboards in the designer, in addition you should add a corresponding script file, which contains all what you need to work with dashboards.

designer.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/stimulsoft.viewer.js"
    type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.designer.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.blockly.editor.js" type="text/javascript"></script>
</head>
```

For dashboards, you need to connect the following package:

console

```
composer require stimulsoft/dashboards-php
```

© 2003-2024 Stimulsoft

In the case of using **PHP** code, all necessary scripts will be included automatically, no code change is required. If you use **JavaScript** code, you will need to add the appropriate script file containing everything you need to work with dashboards:

designer.php

```
<head>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.reports.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/dashboards-php/scripts/
   stimulsoft.dashboards.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.viewer.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.designer.js" type="text/javascript"></script>
   <script src="vendor/stimulsoft/reports-php/scripts/
   stimulsoft.blockly.editor.js" type="text/javascript"></script>
</head>
```

Information

You can find possible optimization variants of deployment in the Optimization of Scripts Loading chapter.

After that, you can use JavaScript functions to work with the designer. To display the designer on the page, you should create an object of the HTML5 Designer component (`StiDesigner`) a visual part of the component will be deployed in this place, where this script will be called.

designer.php

```
<script type="text/javascript">
   <?php
      $designer = new \Stimulsoft\Designer\StiDesigner();
      $designer->renderHtml();
   ?>
</script>
```

© 2003-2024 Stimulsoft

Arguments of the designer constructor

The `StiDesigner` object is created with the help of the `Stimulsoft.Designer.StiDesigner()` constructor, which can take optional arguments as input, which exert influence over its work. There are three arguments.

designer.php

```
<script type="text/javascript">
   <?php
      $designer = new \Stimulsoft\Designer\StiDesigner($options,
      $designerId);
      $designer->renderHtml();
   ?>
</script>
```

Name	Description
`options`	It is the set of options found in the `Stimulsoft.Designer.StiDesignerOptions` class. The options are divided into categories and contains all what you need to set the behavior and appearance of the designer. You can find a detailed description of categories and options in the Designer Settings chapter.
`designerId`	String identifier of the HTML element of the designer. The "`StiDesigner`" value is used by default.

The designer display variants

There are two variants of the designer display: immediately at the location of the script and deferred at the specified HTML element of the page. The second variant can be useful if, for example, you need to create and execute the designer scripts in advance, display the designer by clicking a button, display the designer on a dynamically rendering page and in other cases.

© 2003-2024 Stimulsoft

To display designer, you should invoke the `renderHtml()` function of the created object.

designer.php

```
<script type="text/javascript">
   <?php
      $designer = new \Stimulsoft\Designer\StiDesigner();
      $designer->renderHtml();
   ?>
</script>
```

When invoking `renderHtml()` function, a visual part of the designer will be rendered in an HTML page, where this function is invoked. If needed, you can display the designer inside a definite HTML element. In this case, you should transfer ID of this element or the link to the element as function argument.

designer.php

```
<script type="text/javascript">
   <?php
      $designer = new \Stimulsoft\Designer\StiDesigner();
      $designer->renderHtml('designerContent');
   ?>
</script>

<body>
   <div id="designerContent"></div>
</body>
```

10.3.2 Activation

After purchasing the product, you should activate license for used components. There are several ways of license key connection.

Activation using code string

To activate using a string, you should copy an encrypted license text from your personal account and register it in the event handler. You should place the code until you create and render your report.

© 2003-2024 Stimulsoft

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setKey('Your activation code...');
   $handler->renderHtml();
?>
```

There is an option to load a license key using JavaScript code on the project page. You should place the code until you create and render your report.

index.php

```
Stimulsoft.Base.StiLicense.Key = "Your activation code...";
```

Activation using a file

To activate using a license file you should download the `license.key` file from your personal account, copy it to the folder of the PHP project, and register it in the event handler.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setFile('license.key');
   $handler->renderHtml();
?>
```

There is an option to load a license key using JavaScript code on the project page. You should place the code until you create and render your report.

index.php

```
Stimulsoft.Base.StiLicense.loadFromFile("license.key");
```

© 2003-2024 Stimulsoft

Protection against license key theft

If you activate license using a string, you can add its code by a certain condition. For example, you want to add a license key for registered users.

index.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   if (!empty($sessionID))
      $handler->license->setKey('Your activation code...');
   $handler->renderHtml();
?>
```

Also, we recommend you change the location and name of the license key file, for example:

license.php

```
<?php
   $handler = new \Stimulsoft\StiHandler();
   $handler->license->setFile('private/a15fc0ef64e6.key');
   $handler->renderHtml();
?>
```

10.3.3 Creating and Editing Reports

To run the designer without a report you don't need to make any actions. To load a component, the main menu of the designer will be displayed. If you need to run the designer with a new report, you can create a new object of the `StiReport` report and assign it to the designer. In addition, you can preload data for a new report or make some other necessary actions.

To edit a report in the designer, you should create the `StiReport` object, load a report template to it and assign the resulting object to the designer. All other actions will be made automatically, the designer will display the first page of the template.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
```

© 2003-2024 Stimulsoft

```
   $report = new \Stimulsoft\Report\StiReport();
   $report.loadFile('reports/SimpleList.mrt');
   $designer->report = $report;
   $designer->renderHtml();
?>
```

The designer can work with standard, packed, and encrypted templates. You can find a detailed description of work with various report formats in the Loading and Saving Reports chapter.

Report creation event

A new report can be created using the main menu of the designer. To preload data for a new report or make some other necessary actions with a new report, you should use the `onCreateReport` event. This event will be invoked when creating a new blank report from the main menu or report creation using the master. For example, when creating a new report you need to connect data to it and synchronize them with data dictionary.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onCreateReport = 'onCreateReport';
   $designer->renderHtml();
?>

function onCreateReport(args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readJsonFile("Data/Demo.json");

   args.report.regData(dataSet.dataSetName, "", dataSet);
   args.report.dictionary.synchronize();
}
```

If needed, you can control the process of creation a new report on the PHP server-side. To do it you should add the invoke of the event handler on the client-side and define the `onCreateReport` event of the same name in the event handler file. You can find a detailed description of work of events in the PHP Events Handler chapter.

designer.php

© 2003-2024 Stimulsoft

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onCreateReport = true;
   $designer->renderHtml();
?>
```

handler.php

```
$handler->onCreateReport = function ($args) {
   $report = $args->report;

   return \Stimulsoft\StiResult::success();
};
```

In arguments of the event a report object created in the designer will be transferred. If needed, you can change it or load a new report. After the event completed, this report will be automatically loaded to the designer to edit. For example, you need to preload a prepared template with embedded connections to data when creating a new report in the designer.

handler.php

```
$handler->onCreateReport = function ($args) {
   $args->report = file_get_contents('reports/NewTemplateWithData.mrt');

   return \Stimulsoft\StiResult::success();
};
```

You can find a detailed description of available values of arguments in the Designer Events chapter.

10.3.4 Saving Reports

There are two variants of report saving in the designer, which are available in the main menu and the main panel of the designer: **Save** and **Save as**. At the same time, each of these saving variants has its modes and settings.

Report saving on the JavaScript client-side

© 2003-2024 Stimulsoft

When clicking the **Save** button, a report template file is saved using browser features and you don't need any settings for it. If you need to save a report using your own methods, you should use the `onSaveReport` event. In arguments of the event, report file name and a report will be transferred. A report can be saved, for example, in a JSON string and transfer it to server-side using your own methods.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onSaveReport = 'onSaveReport';
   $designer->renderHtml();
?>

function onSaveReport(args) {
   var fileName = args.fileName;
   var report = args.report;

   var jsonReport = args.report.saveToJsonString();
}
```

If the event is defined after the event completed, the designer continues to work without any additional actions and messages. If needed after saving a report you can display the dialog window with an error or a text message. To do it you should use the `StiError.showError()` function. It's up to you to define the need of display an error message.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onSaveReport = 'onSaveReport';
   $designer->renderHtml();
?>

function onSaveReport(args) {
   var report = args.report;

   // Error message
   Stimulsoft.System.StiError.showError("An error occurred while saving the
   report.");

   // Info message
   Stimulsoft.System.StiError.showError("The report was saved
   successfully.", true, true);
}
```

© 2003-2024 Stimulsoft

When clicking the **Save As** button, the dialog window with report file name request will be displayed. After that report template file will be saved using browser features. If you need to save a report using your own methods, you should use the `onSaveAsReport` event. In arguments of the event report file name and a report will be transferred.

The work of this event is similar to the `onSaveReport` event work, apart from that after the event completed automatic saving report template on a computer using browser features. To stop this action, you should set the `preventDefault` property in the `true` value in arguments of the event and in this case, automatic saving will not be done.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onSaveAsReport = 'onSaveAsReport';
   $designer->renderHtml();
?>

function onSaveAsReport(args) {
   args.preventDefault = true;
}
```

If needed, you can get access to an original report name or report name from the saving dialog by the following way.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onSaveAsReport = 'onSaveAsReport';
   $designer->renderHtml();
?>

function onSaveAsReport(args) {
   // Report name from the designer save dialog
   var reportName1 = args.fileName;

   // Original report name from properties
   var reportName2 = args.report.reportName;
```

© 2003-2024 Stimulsoft

```
}
```

You can find a detailed description of available values of arguments in the Designer Events section.

Report saving on the PHP server-side

To save a report on the PHP server-side, you should add the invoke of the event handler on the client-side, define the `onSaveReport` event of the same name on the server-side in the event handler file. You can find a detailed description of work of events in the PHP Events Handler chapter.

designer.php

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onSaveAsReport = true;
   $designer->renderHtml();
?>
```

handler.php

```
$handler->onSaveReport = function ($args) {
   $fileName = $args->fileName;
   $report = $args->report;
   $reportJson = $args->reportJson;

   return \Stimulsoft\StiResult::success();
};
```

In arguments of the event report file name and a report as an object and as a JSON string will be transferred. To save a report you can use standard PHP functions. For example, you need to save an edited report as a file in the specified directory.

handler.php

```
$handler->onSaveReport = function ($args) {
   $fileName = $args->fileName;
   $reportJson = $args->reportJson;
```

© 2003-2024 Stimulsoft

```
    file_put_contents('reports/'.$fileName.'.mrt', $reportJson);

    return \Stimulsoft\StiResult::success();
};
```

You can find a detailed description of available values of arguments in the Designer Events chapter.

Information

The work of the `onSaveAsReport` event is implemented in the same way on the PHP server-side. All arguments of the event have the same names and values.

10.3.5 Localization

The Designer supports full localization of its interface. To localize the interface into the required language, you should set the required file name for the `localization` option of the viewer:

designer.php

```
<?php
    $options = new \Stimulsoft\Designer\StiDesignerOptions();
    $options->localization = 'de.xml';

    $designer = new \Stimulsoft\Designer\StiDesigner($options);
    $designer->renderHtml();
?>
```

All available localization XML files you may find in the resources of the installed product package. If necessary, you may download the localization file from any other location. For this, you need to specify the full path to the desired XML file for the `localization` option:

designer.php

```
<?php
    $options = new \Stimulsoft\Designer\StiDesignerOptions();
```

© 2003-2024 Stimulsoft

```
    $options->localization = '/resources/loc/de.xml';

    $designer = new \Stimulsoft\Designer\StiDesigner($options);
    $designer->renderHtml();
?>
```

The designer has the functionality to select the required localization of the interface using a special menu on the toolbar. By default, the English (built-in) localization is added to this menu and set using the `localization` property. To add additional localizations to the menu, use the special `addLocalization()` function in the designer options. The localization file or the full path to this file is specified as a parameter:

designer.php

```
<?php
    $options = new \Stimulsoft\Designer\StiDesignerOptions();
    $options->addLocalization("fr.xml");
    $options->addLocalization("it.xml");
    $options->addLocalization("pl.xml");

    $options->localization = 'de.xml';

    $designer = new \Stimulsoft\Designer\StiDesigner($options);
    $designer->renderHtml();
?>
```

10.3.6 Preview

There is the preview of an edited report mode. To use it, you should go to the corresponding tab in the designer window. Report template will be rendered and displayed in the embedded viewer.

Preview event

You can make some necessary actions, for example, connect data for a report before view a report. To do it, you should use the `onPreviewReport` event, which will be invoked before viewing a report. In arguments of the event, the report intended for viewing will be transferred.

designer.php

© 2003-2024 Stimulsoft

```
<?php
   $designer = new \Stimulsoft\Designer\StiDesigner();
   $designer->onPreviewReport = 'onPreviewReport';
   $designer->renderHtml();
?>

function onPreviewReport(args) {
   var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
   dataSet.readJsonFile("Data/Demo.json");

   args.report.regData(dataSet.dataSetName, "", dataSet);
}
```

Additional features

The report preview in the designer is a full-featured interactive viewer, which can print and export a report and supports the work with report parameters. All available interactive actions are supported, such as dynamic sorting, drill-down, collapsing. To use the specified features you don't need any additional settings of the report designer.

If needed, you can use any events of the viewer. To do it you should refer to the embedded `StiViewer` object in the report designer and subscribe to the event you need. This option is only available when using **JavaScript** code, for example:

designer.php

```
designer.viewer.onBeginExportReport = function (event) {
   switch (event.format) {
      case Stimulsoft.Report.StiExportFormat.Html:
         event.settings.zoom = 2; // Set zoom to 200%
         break;
   }
}
```

Information

Refer to the embedded viewer is possible only after display of the designer component on the page, i.e. after the `designer.renderHtml()` method invoked or the viewer object will not be created.

© 2003-2024 Stimulsoft

You can find the list of available events of the embedded viewer and the description of using variants in the Viewer Events chapter.

10.3.7 Appearance

The designer has the ability, which allows you to change design themes of visual controls. To do this, you should set the `theme` property in the component options:

designer.php

```
<?php
   $options = new \Stimulsoft\Designer\StiDesignerOptions();
   $options->appearance->theme = \Stimulsoft\Designer
   \StiDesignerTheme::Office2022WhiteBlue;
?>
```

There are currently **2 themes** available with different color accents. As a result, more than **50 themes** are available. This allows you to customize the appearance of the designer for almost any design of the Web project.

10.3.8 Designer Events

The designer supports the events, which allow you to make necessary operations before definite actions both on the JavaScript client-side and on the PHP server-side. To make the events work on the client-side you don't need any additional settings. To make the events work on the server-side, you should add a special event handler to the page. You can find a detailed description and settings of the PHP event handler in the PHP Events Handler.

The designer supports the following events:

- onBeginProcessData
- onEndProcessData
- onPrepareVariables
- onCreateReport
- onOpenReport
- onSaveReport
- onSaveAsReport
- onPreviewReport

© 2003-2024 Stimulsoft

- onExit

onBeginProcessData
The event is invoked before data request, which are needed to render a report. You can find the list of arguments of the event in the Engine Events chapter. You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL Data Adapters chapters.

onEndProcessData
The event is invoked after loading data before rendering a report. You can find the list of arguments of the event in the Engine Events chapter. You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL Data Adapters chapters.

onPrepareVariables
The event is invoked before rendering a report after preparing report variables. You can find the list of arguments of the event in the Engine Events chapter. You can find a detailed description and samples of using in the Connecting Data Files and Connecting SQL Data Adapters chapters.

onCreateReport
The event is invoked after creation a new report in the designer. In the table below you can see the list of arguments of the event handler on the JavaScript client-side.

Name	Description
event	The identifier of the current event has the "CreateReport" value.
sender	The identifier of the component, which initiated this event can take the following values: • Designer
report	Current report object.

© 2003-2024 Stimulsoft

isWizardUsed	The flag indicates that a new report is created with the help of the master (the true value) or a blank report is created (the false value).

In the table below, you can see the list of the event handler arguments on the PHP server-side.

Name	Description
sender	The identifier of the component, which initiated this event can take the following values: • StiComponentType::Designer
report	The current report presented as an object.
isWizardUsed	The flag indicates that a new report is created with the help of the master (the true value) or a blank report is created (the false value).

onOpenReport

The event is invoked before opening a report from the designer menu. In the table below, you can see the list of the event handler arguments on the JavaScript client-side.

Name	Description
event	The identifier of the current event has the "OpenReport" value.
sender	The identifier of the component, which initiated this event can take the following values: • Designer
preventDefault	This flag allows you to stop further event processing by the designer.

© 2003-2024 Stimulsoft

	The `true` value is set by default.

onSaveReport

The event is invoked when saving a report in the designer. In the table below, you can see the list of the event handler arguments on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`SaveReport`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Designer`
`report`	Current report object.
`fileName`	Report file name for saving.
`preventDefault`	This flag allows you to stop further event processing by the designer. The `true` value is set by default.

In the table below, you can see the list of the event handler arguments on the PHP server-side.

Name	Description
`sender`	The identifier of the component, which initiated this event can take the following values: • `StiComponentType::Designer`
`report`	The current report presented as an object.
`reportJson`	The current report presented as a JSON string.
`fileName`	Report file name for saving

© 2003-2024 Stimulsoft

onSaveAsReport

The event is invoked when saving a report in the designer with a preliminary input of the file name. In the table below, you can see the list of the event handler arguments on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`SaveAsReport`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Designer`
`report`	Current report object.
`fileName`	Report file name for saving.
`preventDefault`	This flag allows you to stop further event processing by the designer. The `true` value is set by default.

In the table below, you can see the list of the event handler arguments on the PHP server-side.

Name	Description
`sender`	The identifier of the component, which initiated this event can take the following values: • `StiComponentType::Designer`
`report`	The current report presented as an object.
`reportJson`	The current report presented as a JSON string.
`fileName`	Report file name for saving.

© 2003-2024 Stimulsoft

onPreviewReport

The event is invoked when going to the report view tab. In the table below, you can see the list of the event handler arguments on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`PreviewReport`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Designer`
`report`	Current report object.
`fileName`	Report file name for saving.
`preventDefault`	This flag allows you to stop further event processing by the designer. The `true` value is set by default.

onExit

The event is invoked when clicking the **Exit** button in the main menu of the designer. In the table below, you can see the list of the event handler arguments on the JavaScript client-side.

Name	Description
`event`	The identifier of the current event has the "`Exit`" value.
`sender`	The identifier of the component, which initiated this event can take the following values: • `Designer`

© 2003-2024 Stimulsoft

10.3.9 Designer Settings

The designer is set using the properties found in the `Stimulsoft.Designer.StiDesignerOptions` class. All properties are divided into groups for comfortable using. All designer classes and enums you may find in the `\Stimulsoft\Designer` namespace. To set the designer you should create the class of options, set required properties and transfer an object of options to the designer constructor as the first argument.

designer.php

```
<?php
   $options = new \Stimulsoft\Designer\StiDesignerOptions();

   $options->appearance->theme = \Stimulsoft\Designer
   \StiDesignerTheme::Office2022WhiteBlue;
   $options->viewerOptions->appearance->reportDisplayMode = \Stimulsoft
   \Designer\StiHtmlExportMode::FromReport;
   $options->toolbar->showFileMenuExit = false;
   $options->toolbar->showFileMenuOptions = false;
   $options->bands->showChildBand = false;
   $options->components->showPanel = false;
   $options->appearance->showReportTree = false;
   $options->appearance->showTooltips = false;

   $designer = new \Stimulsoft\Designer\StiDesigner($options);
?>
```

Main settings (without groups)

Name	Description
`Width`	It sets component width in "px or "%". The 100 % value is set by default.
`Height`	It sets component height in "px" or "%". The "800px" value is set by default.
`localization`	Sets the selected localization of the component. By default, the English localization is set. It is built into the component.

© 2003-2024 Stimulsoft

Appearance

Name	Description
`theme`	Specifies the theme of the designer layout. The list of available themes can be found in the `StiDesignerTheme` enumeration. The default value is `Office2022WhiteBlue`.
`defaultUnit`	It sets the units of measure of sizes for a report and all its components: • `StiReportUnitType::Centimeters` (value is set by default); • `StiReportUnitType::HundredthsOfInch`; • `StiReportUnitType::Inches`; • `StiReportUnitType::Millimeters`.
`zoom`	Sets the zoom for displaying report pages. The default setting is 100 percent. It can take one of the following values of the `StiZoomMode` enumeration: • `StiZoomMode::PageWidth` – when the designer runs, the zoom, necessary to display the report by the page width, will be set; • `StiZoomMode::PageHeight` – when the designer runs, the zoom, required to display the page height of the report, will be set.

© 2003-2024 Stimulsoft

`interfaceType`	It sets the type of the designer interface. The following values can be used: • `StiInterfaceType::Auto` – the type of the designer interface will be selected automatically depending on the device you use (the value by default); • `StiInterfaceType::Mouse` – the forced using **Mouse** interface for controlling the designer using a computer mouse; • `StiInterfaceType::Touch` – the forced using of the **Touch** interface to control the designer using touch screen of a monitor. In this mode, the designer interface elements have enlarged sizes for comfortable control.
`showAnimation`	It allows you to enable or disable the animation of display and closing various menus in the designer. The `true` value is set by default.
`showSaveDialog`	It enables the display of the input dialog of report name when its saving. Report name will be transferred in the designer parameters. The `true` value is set by default.
`showTooltips`	It enables the display of prompts for the designer controls when hovering. The `true` value is set by default.
`showTooltipsHelp`	It enables the display of the link to the online documentation in prompts for the designer controls.

© 2003-2024 Stimulsoft

	The `true` value is set by default.
`showDialogsHelp`	It allows you to display or not to display the reference invoke button in various menus. The `true` value is set by default.
`fullScreenMode`	It sets the full screen mode of the designer display. If the property is set in the `true` value, the values of the width and height properties are ignored. The `false` value is set by default.
`maximizeAfterCreating`	It allows you to set max size of the report designer. The `false` value is set by default.
`showLocalization`	It allows you to display or not to display the localization control in the report designer. The `true` value is set by default.
`allowChangeWindowTitle`	It allows you to use the header of browser window to display file name of edited report. The `true` value is set by default.
`showPropertiesGrid`	It enables the display of the property panel in the report designer. The `true` value is set by default.
`showReportTree`	It enables the display of report components tree. The `true` value is set by default.
`propertiesGridPosition`	It allows you to define the position of the property panel to the left or to the right: • `StiPropertiesGridPosition::Left` – the property panel will be displayed on the left (value is set by default); • `StiPropertiesGridPositio`

© 2003-2024 Stimulsoft

	`n::Right` – the property panel will be displayed on the right.
`showSystemFonts`	It allows you to display or not to display system fonts in the list of fonts. The property has the `true` value by default, i.e. system fonts are displayed in the list of fonts.
`datePickerFirstDayOfWeek`	It allows you to set the first day of the week for the **Date picker** element: • `StiFirstDayOfWeek::Auto` - Monday or Sunday will be set as the first day of the week depending on browser culture (value is set by default); • `StiFirstDayOfWeek::Monday` - Monday will be set as the first day of the week; • `StiFirstDayOfWeek::Sunday` - Sunday will be set as the first day of the week.
`undoMaxLevel`	It sets max depth of report changes cancel when its editing. It exerts influence over memory usage. 6 changes are set by default.
`wizardTypeRunningAfterLoad`	It allows you to call the master of report creation after the report designer is run. It can take one of the specified below enumeration values: • `StiWizardType::None` - the report designer will be run without the invoke of report creation master (value is set by default); • `StiWizardType::StandardReport` - the report designer will be run with the invoke of standard report creation master;

© 2003-2024 Stimulsoft

	• `StiWizardType::MasterDetailReport` - the report designer will be run with the invoke of master-detail report creation master; • `StiWizardType::LabelReport` - the report designer will be run with the invoke of the report creation master with labels; • `StiWizardType::InvoicesReport` - the report designer will be run with the invoke of the invoice creation master; • `StiWizardType::OrdersReport` - the report designer will be run with the invoke of the order creation master; • `StiWizardType::QuotationReport` - the report designer will be run with the invoke of the quota creation master.
`allowWordWrapTextEditors`	It enables or disables line break in editors of text in the designer. The property has the `true` value by default.

Toolbar

Name	Description
`visible`	It enables the display of toolbar in the report designer. The property has the `true` value by default.
`showPreviewButton`	It enables or disables the display of the **Preview** button in the designer toolbar. The property has the `true`

© 2003-2024 Stimulsoft

	value by default.
showSaveButton	It enables the display of the **Save** button in the designer toolbar. The property has the true value by default.
showAboutButton	It enables the display of the **About** button in the designer toolbar. The property has the false value by default.
showFileMenu	It enables the display of the main menu of the report designer. The property has the true value by default.
showFileMenuNew	It enables the display of the **New** main menu item. The property has the true value by default.
showFileMenuOpen	It enables the display of the **Open** main menu item. The property has the true value by default.
showFileMenuSave	It enables the display of the **Save** main menu item. The property has the true value by default.
showFileMenuSaveAs	It enables the display of the **Save as** main menu item. The property has the true value by default.
showFileMenuClose	It enables the display of the **Close** main menu item. The property has the true value by default.
showFileMenuExit	It enables the display of the **Exit** main menu item. The property has the true value by default.
showFileMenuReportSetup	It enables the display of the **Report Setup** main menu item. The property has the true value by default.
showFileMenuOptions	It enables the display of the **Options**

© 2003-2024 Stimulsoft

	main menu item. The property has the `true` value by default.
`showFileMenuInfo`	It enables the display of the **Info** main menu item. The property has the `true` value by default.
`showFileMenuAbout`	It enables the display of the **About** main menu item. The property has the `true` value by default.
`showFileMenuNewReport`	It enables or disables the display of the **New Page** main menu item. The property has the `true` value by default.
`showFileMenuNewDashboard`	It enables or disables the display of the **New Dashboard** main menu item. The property has the `true` value by default.
`showSetupToolboxButton`	It enables or disables the display of the report components side panel settings invoke button. The property has the `true` value by default.
`showNewPageButton`	It enables or disables the display of the **New Page** button in the toolbar. The property has the `true` value by default.
`showNewDashboardButton`	It enables or disables the display of the **New Dashboard** button in the toolbar. The property has the `true` value by default.

Bands

Name	Description
`showReportTitleBand`	It enables the display of the **Report Title** band in the designer

© 2003-2024 Stimulsoft

	components insert menu. The property has the `true` value by default.
`showReportSummaryBand`	It enables the display of the **Report Summary** band in the designer components insert menu. The property has the `true` value by default.
`showPageHeaderBand`	It enables the display of the **Page Header** band in the designer components insert menu. The property has the `true` value by default.
`showPageFooterBand`	It enables the display of the **Page Footer** band in the designer components insert menu. The property has the `true` value by default.
`showGroupHeaderBand`	It enables the display of the **Group Header** band in the designer components insert menu. The property has the `true` value by default.
`showGroupFooterBand`	It enables the display of the **Group Footer** band in the designer components insert menu. The property has the `true` value by default.
`showHeaderBand`	It enables the display of the **Header** band in the designer components insert menu. The property has the `true` value by default.
`showFooterBand`	It enables the display of the **Footer** band in the designer components insert menu. The property has the `true` value by default.

© 2003-2024 Stimulsoft

showColumnHeaderBand	It enables the display of the **Column Header** band in the designer components insert menu. The property has the true value by default.
showColumnFooterBand	It enables the display of the **Column Footer** band in the designer components insert menu. The property has the true value by default.
showDataBand	It enables the display of the **Data** band in the designer components insert menu. The property has the true value by default.
showHierarchicalBand	It enables the display of the **Hierarchical** band in the designer components insert menu. The property has the true value by default.
showChildBand	It enables the display of the **Child** band in the designer components insert menu. The property has the true value by default.
showEmptyBand	It enables the display of the **Empty** band in the designer components insert menu. The property has the true value by default.
showOverlayBand	It enables the display of the **Overlay** band in the designer components insert menu. The property has the true value by default.
showTable	It enables the display of the **Table** component in the designer components insert menu. The property has the true value by default.

© 2003-2024 Stimulsoft

showTableOfContents	It enables the display of the **Table of Contents** band in the designer components insert menu. The property has the `true` value by default.

Cross-Bands

Name	Description
showCrossTab	It enables the display of the **Cross-Tab** component in the designer components insert menu. The property has the `true` value by default.
showCrossGroupHeaderBand	It enables the display of the **Cross-Group Header** band in the designer components insert menu. The property has the `true` value by default.
showCrossGroupFooterBand	It enables the display of the **Cross-Group Footer** band in the designer components insert menu. The property has the `true` value by default.
showCrossHeaderBand	It enables the display of the **Cross-Header** band in the designer components insert menu. The property has the `true` value by default.
showCrossFooterBand	It enables the display of the **Cross-Footer** band in the designer components insert menu. The property has the `true` value by default.
showCrossDataBand	It enables the display of the **Cross-**

© 2003-2024 Stimulsoft

	Data band in the designer components insert menu. The property has the `true` value by default.

dashboardElements

Name	Description
`showTableElement`	It enables the display of the **Table** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showCardsElement`	It enables the display of the **Cards** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showChartElement`	It enables the display of the **Chart** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showGaugeElement`	It enables the display of the **Gauge** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showPivotTableElement`	It enables the display of the **Pivot** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showIndicatorElement`	It enables the display of the **Indicator** dashboard element in the

© 2003-2024 Stimulsoft

	toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showProgressElement`	It enables the display of the **Progress** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showRegionMapElement`	It enables the display of the **Region Map** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showOnlineMapElement`	It enables the display of the **Online Map** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showImageElement`	It enables the display of the **Image** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showTextElement`	It enables the display of the **Text** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showPanelElement`	It enables the display of the **Panel** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showShapeElement`	It enables the display of the **Shape** dashboard element in the toolbox or the **Insert** tab in the designer. The

© 2003-2024 Stimulsoft

	property has the `true` value by default.
`showButtonElement`	It enables the display of the **Button** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showListBoxElement`	It enables the display of the **List Box** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showComboBoxElement`	It enables the display of the **Combo Box** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showTreeViewElement`	It enables the display of the **Tree View** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showTreeViewBoxElement`	It enables the display of the **Tree View Box** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.
`showDatePickerElement`	It enables the display of the **Date Picker** dashboard element in the toolbox or the **Insert** tab in the designer. The property has the `true` value by default.

Components

© 2003-2024 Stimulsoft

Name	Description
showText	It enables the display of the **Text** component in the designer components insert menu. The property has the true value by default.
showTextInCells	It enables the display of the **Text in Cells** component in the designer components insert menu. The property has the true value by default.
showRichText	It enables the display of the **Rich Text** component in the designer components insert menu. The property has the true value by default.
showImage	It enables the display of the **Image** component in the designer components insert menu. The property has the true value by default.
showBarCode	It enables the display of the **Bar Code** component in the designer components insert menu. The property has the true value by default.
showShape	It enables the display of the **Shape** component in the designer components insert menu. The property has the true value by default.
showPanel	It enables the display of the **Panel** component in the designer components insert menu. The property has the true value by default.

© 2003-2024 Stimulsoft

showClone	It enables the display of the **Clone** component in the designer components insert menu. The property has the `true` value by default.
showCheckBox	It enables the display of the **Check Box** component in the designer components insert menu. The property has the `true` value by default.
showSubReport	It enables the display of the **Text** component in the designer components insert menu. The property has the `true` value by default.
showZipCode	It enables the display of the **Sub-Report** component in the designer components insert menu. The property has the `true` value by default.
showChart	It enables the display of the **Chart** component in the designer components insert menu. The property has the `true` value by default.
showGauge	It enables the display of the **Gauge** component in the designer components insert menu. The property has the `true` value by default.
showSparkline	It enables the display of the **Sparkline** component in the designer components insert menu. The property has the `true` value by default.
showMathFormula	It enables the display of the **Math**

© 2003-2024 Stimulsoft

	Formula component in the designer components insert menu. The property has the `true` value by default.
`showMap`	It enables the display of the **Map** component in the designer components insert menu. The property has the `true` value by default.

Dictionary

Name	Description
`showAdaptersInNewConnectionForm`	It enables the display of the **Object** category in the new connection creation window. The property has the `true` value by default.
`showDictionary`	It enables the display of the report dictionary. The property has the `true` value by default.
`newReportDictionary`	Allows you to use aliases in the data dictionary. It can take one of the specified below enumeration values: • `StiNewReportDictionary.Auto` - defines the mode of using aliases from a saved value in cookies (default value); • `StiNewReportDictionary.True` - sets the mode of using aliases in the data dictionary; • `StiNewReportDictionary.False` - disables the mode of using aliases in the data dictionary.

© 2003-2024 Stimulsoft

`useAliases`	It allows you to create a new data dictionary or join the existing one when creating a new report in the designer. It can take one of the specified below enumeration values: • `StiUseAliases.Auto` - defines the mode to create or join the data dictionary from a saved value in cookies (default value); • `StiUseAliases.DictionaryNew` - sets the mode to create a new data dictionary when creating a new report; • `StiUseAliases.DictionaryMerge` - sets the mode to join the existing data dictionary with a new one when creating a new report in the designer.
`showDictionaryContextMenuProperties`	Sets a visibility of the **Properties** item in the dictionary context menu. By default, the property is set to `true`.
`showDictionaryActions`	Sets a visibility of the **Actions** menu on the dictionary toolbar. By default, the property is set to **true**.
`dataSourcesPermissions`	It sets available actions on report data sources. It can take one or several values from the `StiDesignerPermissions` enumeration.
`dataConnectionsPermissions`	It sets available actions on connections to report data. It can take one or several values from the `StiDesignerPermissions` enumeration.

© 2003-2024 Stimulsoft

`dataColumnsPermissions`	It sets available actions on report data columns. It can take one or several values from the `StiDesignerPermissions` enumeration.
`dataRelationsPermissions`	It sets available actions on report data connections. It can take one or several values from the `StiDesignerPermissions` enumeration.
`businessObjectsPermissions`	It sets available actions on report business objects. It can take one or several values from the `StiDesignerPermissions` enumeration.
`variablesPermissions`	It sets available actions on report variables. It can take one or several values from the `StiDesignerPermissions` enumeration.
`resourcesPermissions`	It sets available actions on sources in report dictionary. It can take one or several values from the `StiDesignerPermissions` enumeration.
`dataTransformationsPermissions`	Sets the available actions on data transformation. It can take one or more values from the **StiDesignerPermissions** enumeration.

In the table below you can see all available values for the `StiDesignerPermissions` enumeration. They can be set for report dictionary elements.

Name	Description

© 2003-2024 Stimulsoft

`StiDesignerPermissions::None`	Disables any action on the item of the data dictionary.
`StiDesignerPermissions::Create`	It allows you to create definite element of the dictionary.
`StiDesignerPermissions::Delete`	It allows you to delete a definite element of the dictionary.
`StiDesignerPermissions::Modify`	It allows you to edit a definite element of the dictionary.
`StiDesignerPermissions::View`	It allows you to view a definite element of the dictionary.
`StiDesignerPermissions::ModifyView`	It allows you to edit and view a definite element of the dictionary.
`StiDesignerPermissions::All`	It allows you to make any actions on the dictionary element.

You can configure the built-in `StiViewer` component used to preview the report. To get access to all of its settings, you should use the `viewerOptions` property, which is an object of the viewer options. All its properties are described in the Viewer settings section. For example, you want to change the report display mode and disable unnecessary export formats:

designer.php

```
<?php
   $options = new \Stimulsoft\Designer\StiDesignerOptions();
   $options->viewerOptions->appearance->reportDisplayMode = \Stimulsoft
   \Viewer\StiHtmlExportMode::FromReport;
   $options->viewerOptions->exports->ShowExportToWord2007 = false;
   $options->viewerOptions->exports->showExportToCsv = false;

   $designer = new \Stimulsoft\Designer\StiDesigner($options);
?>
```

11 Reports and Dashboards for WinForms

YouTube

© 2003-2024 Stimulsoft

Watch the video tutorials for working with the components of the Stimulsoft Reports.NET product. Subscribe to the Stimulsoft channel and be the first to watch new video lessons. Questions and suggestions is recommended be left in the comments to the video.

Samples

See on our website examples for working with the Reports.NET components. All examples are separate projects, grouped into one solution for Visual Studio. Also, you can view and download specific examples on GitHub.

- WinForms Viewer
- Activation

11.1 Activation

YouTube

Watch videos which show how to activate the WinForms components. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this done in various ways. Below is an example of activating the **WinForms** component.

Form1.cs

```
...
public partial class Form1 : Form
{
   public Form1()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
```

© 2003-2024 Stimulsoft

```
      Stimulsoft.Base.StiLicense.LoadFromFile("license.key");

      //Activation from byte array
      Stimulsoft.Base.StiLicense.LoadFromBytes(bytes);

      //Activation from stream
      Stimulsoft.Base.StiLicense.LoadFromStream(stream);

      //Activation from assembly
      Stimulsoft.Base.StiLicense.LoadFromEntryAssembly(assembly,
      "stimulsoft-license.key");

      InitializeComponent();
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

11.2 WinForms Viewer

The **StiViewerControl** component is used to view reports in the WinForms. The component can show a report, zoom, save rendered reports to various formats, print reports, send them to a recipient via Email.

- How to Show Report
- Dot-Matrix Viewer for WinForms

11.2.1 How to Show Report

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

Just call only one method to show a report:

© 2003-2024 Stimulsoft

C#

```
...
StiReport report = new StiReport();
report.Load("report.mrt");
report.Show();
...
```

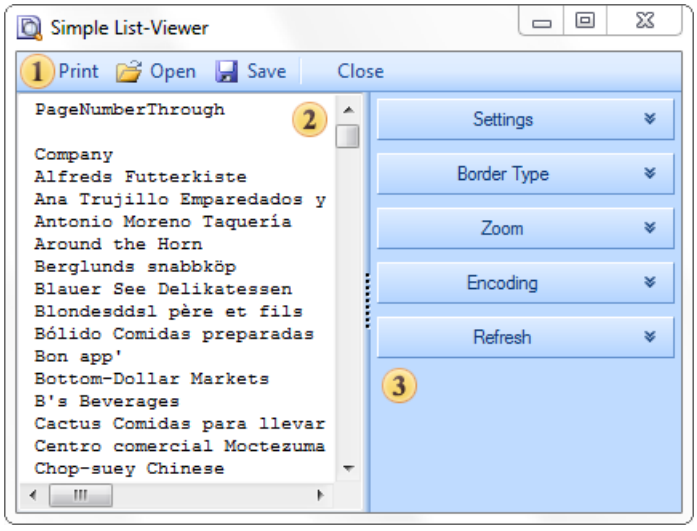
VB.NET

```
...
Dim Report As StiReport = New StiReport()
Report.Load("report.mrt")
Report.Show()
...
```

If the report was not rendered before showing, the **Show** method will render a report using the **Render** method.

11.2.2 Dot-Matrix Mode of WinForms Viewer

The **Dot-Matrix** viewer is designed to preview the report before printing it on dot matrix printer. The Dot matrix printer is used to print only the text and characters of pseudographics. Accordingly the viewer displays only the text and borders of objects as pseudographics characters. The picture below shows the Dot-matrix viewer dialog box:



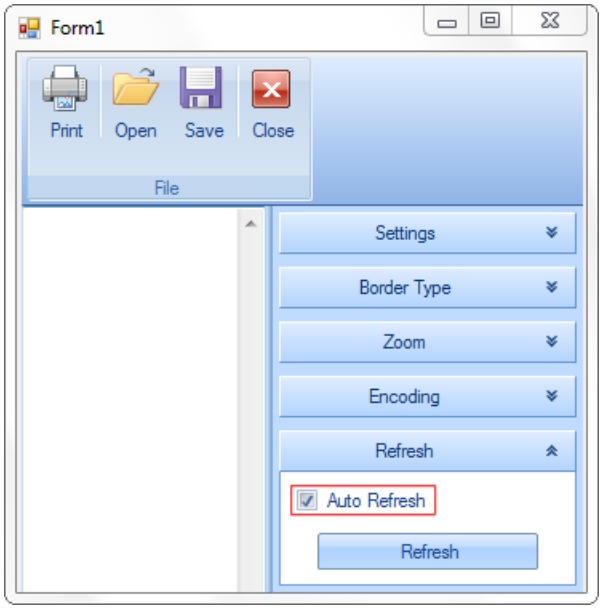
1. The Dot-matrix viewer toolbar.
2. The panel displays the text of a report

© 2003-2024 Stimulsoft

3. The options bar of a report.

Setting Dot-Matrix Viewer in WinForms

The **Dot-Matrix** viewer can be configured from code using static properties. Depending on the value of the static properties in the Dot-matrix viewer, these or that parameters will be specified. For example, the **AutoRefresh** property. The picture below shows the **Dot-Matrix** viewer dialog box:



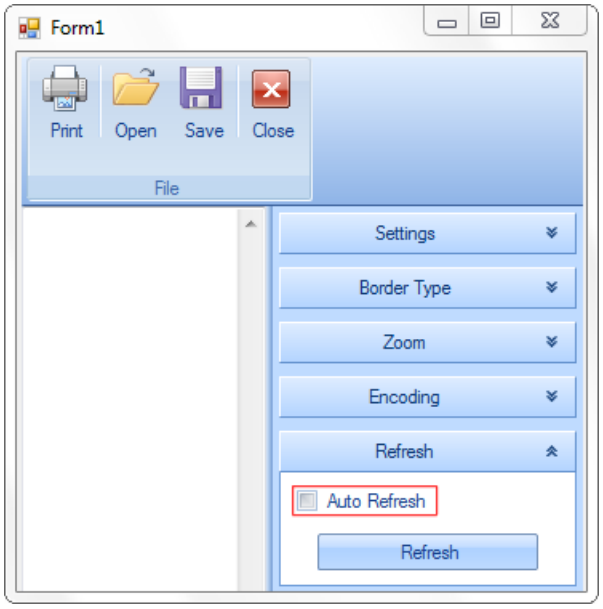
As can be seen on the picture above, the **Auto Refresh** property is enabled. This means that the **AutoRefresh** static property of the **Dot-Matrix** viewer is set to true. If the AutoRefresh static property is set to false, then the AutoRefresh property in the Dot-Matrix viewer is disabled. Add the following code into the project code:

C#

```
...
StiOptions.Viewer.DotMatrix.AutoRefresh = false;
...
```

Thus, the AutoRefresh property will be disabled. The picture below shows the Dot-Matrix viewer dialog box with disabled auto refresh function:

© 2003-2024 Stimulsoft



Most parameters can be set using the static properties.

Dot-Matrix and Escape Codes

For inserting the escape sequence to text the commands that may look like <#command> should be used as seen in the code sample below:

```
Normal text <#b> Bold text <#/b><#i> Italic text <#/i> Again normal text
```

Also commands of selecting bold, italic or underlined text are automatically inserted depending on the style of the text box font. When printing to matrix printer and exporting to text format these commands are changed on appropriate escape sequences.
The **StiEscapeCodesCollection** is used for this process. It is inherited from the Hashtable class. This is a collection of "key-value" pairs where the key is the command and value is the escape-sequence. For different types of printers different collections with different set of command can be defined. Collections are stored in the **StiOptions.Export.Txt.EscapeCodesCollectionList** static variable. By default, the following collections will be created: "None", "EpsonFX", "Oki ML92/93". The "None" collection is empty and used to output the text without escape codes.

© 2003-2024 Stimulsoft

Command/Collection	EpsonFX	Oki ML92/93
b	ESC E	ESC T
/b	ESC F	ESC I
i	ESC 4	
/i	ESC 5	
u	ESC -1	ESC H
/u	ESC -0	ESC D
sup	ESC S0	ESC J
/sup	ESC T	ESC K
sub	ESC S1	ESC L
/sub	ESC T	ESC M
condensed	0x0F	0x1d
/condensed	0x12	0x1e
elite	ESC M	0x1c
pica	ESC P	0x1e
doublewidth	ESC W1	0x1f
/doublewidth	ESC W0	0x1e

It is possible to add new collections of change the existing ones. The selection of the required collection is done by the name. If the collection with the name is not found then the "None" collection is used. The collection name can be selected from the DotMatrixViewer settings and passed as an option to the exporting and printing methods.

11.3 Add custom functions

You can add a custom function to the Dictionary in the report designer when you integrate it into your application. After adding the custom function, you can use this in creating reports and dashboards. Below is the example of adding a function for calculating the sum total.

Designer.cs

```
. . .
```

© 2003-2024 Stimulsoft

```
StiFunctions.AddFunction("MyCategory", "MySum",
"description", typeof(MyClass),
typeof(decimal), "Calculates a sum of the specified set of values.",
new[] { typeof(object) },
new[] { "values" },
new[] { "A set of values" }).UseFullPath = false;
...
public static decimal MySum(object value)
{
   if (!ListExt.IsList(value))
   return Stimulsoft.Base.Helpers.StiValueHelper.TryToDecimal(value);
   return
   Stimulsoft.Data.Functions.Funcs.SkipNulls(ListExt.ToList(value)).TryCast
   ToDecimal().Sum();
}
...
```

12 Reports and Dashboards for Python

Stimulsoft Reports.PYTHON and Stimulsoft Dashboards.PYTHON comprise a combination of a JavaScript client and a Python server, along with all the necessary functionality for their interaction. The report is built and exported on the client side using JavaScript, while the Python server side contains everything needed to work with report files and communicate with various SQL data sources. Communication between the client and the server is carried out through AJAX requests that transmit and receive JSON data in a specific format. To enhance the usability of the product, special events and functions have been introduced both on the JavaScript client side and on the Python server side.

12.1 Report Engine

The reporting tool Stimulsoft Reports.PYTHON allows you to download, build and export a report in various formats without deploying a viewer and designer. This allows you to avoid loading a large number of scripts on the client side and increases the speed and ease of operation.

The report is built and exported on the client side using JavaScript; the Python server side contains everything needed to work with report files and communicate with various SQL data sources. Communication between the client and the server is carried out through AJAX requests that transmit and receive JSON data in a specific format. For ease of use of the product, special events and functions have been developed both on the JavaScript client side and on the Python server side.

Dashboards

The dashboard tool Stimulsoft Dashboards.PYTHON allows loading and analyze data,

© 2003-2024 Stimulsoft

exporting and printing analytical panels to various formats without deployment of the viewer and designer.

All data analysis, except for certain SQL operations, is performed on the client side using JavaScript; the Python server-side contains everything needed to work with dashboard files and communicate with various SQL data sources. The client communicates with the server in exactly the same way as in the reporting tool; the same events and functions are used.

- Deployment
- Report Rendering
- Optimizing Scripts Loading
- Printing a Report From Code
- License Activation
- Exporting a Report From Code

12.1.1 Deployment

Samples

The complete code sample can be found on GitHub.

To use the reporting tool, just install `stimulsoft-reports` or `stimulsoft-dashboards` package using the package installer by running the following command:

console

```
python -m pip install stimulsoft-reports
```

console

```
python -m pip install stimulsoft-dashboards
```

The latest available version of the reporting tool will be installed in the current

© 2003-2024 Stimulsoft

workspace, after which you can use classes and functions for working with reports.

Information

For code examples, the **Flask** framework is used as one of the most popular and easy-to-understand frameworks. You can use any web framework because all classes and functions for working with the reporting tool are completely independent.

The `StiReport` class is designed to work with a reporting tool in a web project. Using this class, you can create a report, load a report from a file or a line, build a report, or export a report. For example, if you need to load a report from a file, build it, and export it to HTML:

app.py

```
from flask import Flask, render_template, url_for, request
from stimulsoft_reports.report import StiReport
from stimulsoft_reports.report.enums import StiExportFormat

app = Flask(__name__)

@app.route('/report', methods = ['GET', 'POST'])
def report():
    report = StiReport()
    if report.processRequest(request):
        return report.getFrameworkResponse()

    report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
    report.render()
    report.exportDocument(StiExportFormat.HTML)

    js = report.javascript.getHtml()
    html = report.getHtml()
    return render_template('report.html', reportJavaScript = js,
reportHtml = html)
```

index.html

```
<!DOCTYPE html>
<html>

<head>
```

© 2003-2024 Stimulsoft

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Render and Export a Report</title>

{{ reportJavaScript|safe }}
</head>

<body>
{{ reportHtml|safe }}
</body>

</html>
```

In this example, an instance of the `StiReport` object is created, a report template is loaded from the SimpleList.mrt file located in the static files directory, and the command to build and export the report is called. Next, the JavaScript and HTML code of the components necessary for the operation of the reporting tool are generated. The `report.javascript.getHtml()` function generates HTML code for connecting the necessary resources, while the `report.getHtml()` function generates HTML code for working with the reporting tool. The generated HTML code is passed as parameters to the HTML template `report.html`, where it is displayed in the required places.

The special function `report.processRequest(request)` processes the current request. If it returns `True`, the request is intended for the reporting tool, and you need to return the result of its execution instead of the page template. This is described in more detail in a separate section of the documentation.

Information

Our products Stimulsoft Reports.PYTHON and Stimulsoft Dashboards.PYTHON do not have a native Python report engine. Report generation and export are performed on the client side using JavaScript code. Therefore, when using Python code to work with components, you should call the getHtml() function, which will return all the necessary JavaScript code to add to the web page.

Framework support

Currently, there is built-in support for the three main Python frameworks - **Flask**, **Django**, and **Tornado**. Universal functions for working in any other frameworks are

© 2003-2024 Stimulsoft

also available. For code examples, the **Flask** framework was used, as it is one of the most popular and easy-to-understand codes; product deployment for it was discussed above. Below are examples of the same code for other frameworks. They are all very similar and differ only in the features used for a specific framework.

Django

app.py

```
from django.shortcuts import render
from django.templatetags.static import static
from stimulsoft_reports.report import StiReport
from stimulsoft_reports.report.enums import StiExportFormat

def report(request):
    report = StiReport()
    if report.processRequest(request):
        return report.getFrameworkResponse()

    report.loadFile(static('reports/SimpleList.mrt'))
    report.render()
    report.exportDocument(StiExportFormat.HTML)

    js = report.javascript.getHtml()
    html = report.getHtml()
    return render(request, 'report.html', {'reportJavaScript': js,
'reportHtml': html})
```

report.html

```
<!DOCTYPE html>
<html>

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Render and Export a Report</title>

    {{ reportJavaScript|safe }}
</head>

<body>
    {{ reportHtml|safe }}
</body>

</html>
```

© 2003-2024 Stimulsoft

Tornado

app.py

```
import asyncio, os
from tornado.web import Application, RequestHandler, url
from stimulsoft_reports.report import StiReport
from stimulsoft_reports.report.enums import StiExportFormat

class ReportHandler(RequestHandler):
    def get(self):
        report = StiReport()
        if report.processRequest(request):
            return report.getFrameworkResponse(self)

        report.loadFile(self.static_url('reports/SimpleList.mrt'))
        report.render()
        report.exportDocument(StiExportFormat.HTML)

        js = report.javascript.getHtml()
        html = report.getHtml()
        self.render('report.html', reportJavaScript = js, reportHtml =
html)

    def post(self):
        handler = StiHandler()
        if handler.processRequest(self.request):
            return handler.getFrameworkResponse(self)
```

report.html

```
<!DOCTYPE html>
<html>

<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Render and Export a Report</title>

    {% raw reportJavaScript %}
</head>

<body>
    {% raw reportHtml %}
</body>

</html>
```

Universal function

© 2003-2024 Stimulsoft

app.py

```
from stimulsoft_reports.report import StiReport
from stimulsoft_reports.report.enums import StiExportFormat

def report():
    report = StiReport()
    query = 'query string'
    body = 'post data'
    if report.processRequest(None, query, body):
        response = report.getResponse()
        data = response.data
        contentType = response.contentType
        mimetype = response.mimetype

    report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
    report.render()
    report.exportDocument(StiExportFormat.HTML)

    js = report.javascript.getHtml()
    html = report.getHtml()
```

Information

In most cases, to work with the product, you should use only Python code, which provides interaction with all the main features. To configure the product more precisely and utilize all the capabilities of the JS reporting software, you must use JavaScript code. The option to deploy a product using only JavaScript code is described in the Reports and Dashboards for JS section. In this case, the use of Python code is required only to connect data adapters.

Various deployment and optimization options are discussed in the section Optimizing scripts loading.

12.1.2 Optimizing Scripts Loading

Due to the impressive functionality of the product, the scripts are quite large. When loading a web application for the first time, or when browser caching is disabled, loading may take some time, especially on a low-speed internet connection. We offer two options for solving this problem: using packaged scripts or employing partial functionality and loading only what is required.

© 2003-2024 Stimulsoft

Packed scripts

Packed scripts have the same structure as regular scripts but end with `*.pack.js` in the file name. Such scripts contain a block of packed data in the form of a JavaScript variable and a compact unpacker. When loading all scripts, the unpacker automatically unpacks all downloaded data and launches the prepared script for execution. Unpacking takes some time, but under certain circumstances - such as with a slow internet connection - this time is much less than the loading speed of regular scripts.

To use packaged scripts, all you should do is set the `javascript.usePacked` property of the report object to `True`, for example:

app.py

```
report = StiReport()
report.javascript.usePacked = True
```

Partial loading of scripts

When deploying the reporting tool, by default only one file with scripts, `stimulsoft.reports.js`, is loaded. It contains all the functionality for building and exporting reports. If you only need some of the capabilities to generate reports, you can download only the required parts of the generator that contain a specific set of capabilities. For example, if your reports do not use maps, you don't have to download them. This will speed up the loading of the web project and reduce memory consumption by the browser.

Information

This feature is implemented only for the report engine; the viewer and designer cannot be divided into parts; their scripts will be loaded entirely in one block.

To use partial loading of scripts, just set the necessary options for the `javascript` property of the report object:

app.py

© 2003-2024 Stimulsoft

```
from stimulsoft_reports.report import StiReport

report = StiReport()

report.javascript.reportsSet = False
report.javascript.reportsChart = True
report.javascript.reportsExport = True
report.javascript.reportsImportXlsx = True
report.javascript.reportsMaps = True
```

Each `javascript` property option controls the loading of a script containing specific functionality. This table presents the entire set of scripts that can be loaded separately:

Name	Description
`javascript.reportsSet`	Contains a complete set of scripts for working with the reporting tool. It should be set to False when using partial script loading.
`javascript.reportsExport`	Contains algorithms for exporting the generated report to various formats - PDF, HTML, Excel, RichText, and others.
`javascript.reportsChart`	Contains components for working with all types of charts in a report.
`javascript.reportsMaps`	Contains components for working with regional and online maps.
`javascript.blocklyEditor`	Contains the Blockly visual editor for creating event scripts in the report. The event handler itself is built into the reporting engine.
`javascript.reportsImportXlsx`	Contains algorithms for working with Excel data sources.

Information

© 2003-2024 Stimulsoft

The report viewer and report designer components also have a `javascript` property, with which you can control the configuration of scripts the way described above.

12.1.3 License Activation

After purchasing the product, you need to activate the license for the components you use. There are several ways to activate the license.

Activation using code

To activate using a string, just copy the encrypted license text from your account, and register it using the special `setKey()` function of the `license` property of the report object:

app.py

```
from stimulsoft_reports.report import StiReport

report = StiReport()
report.license.setKey('Your activation code...')
```

Activation using a license.key file

To activate using a license file, just download the `license.key` file from your account, and copy it to the Web project folder, for example, to the `static` folder with static files. Then, just register it using the special `setFile()` function of the `license` property of the report object:

app.py

```
from stimulsoft_reports.report import StiReport

report = StiReport()
report.license.setFile(url_for('static', filename='license.key'))
```

Information

© 2003-2024 Stimulsoft

The report viewer and report designer components also have the `license` property using which you can manage the license key in the way described above.

Protecting the license key

If you activate a license using a string, you can add its code under a certain condition. For example, the `sessionID` variable stores information about the current client session; you need to add a license key only for authorized users:

app.py

```
from stimulsoft_reports.report import StiReport

report = StiReport()

if sessionID != None:
    report.license.setKey('Your activation code...')
```

It would also work if you change the location and name of the license key file, for example:

app.py

```
from stimulsoft_reports.report import StiReport

report = StiReport()
report.license.setFile(url_for('private', filename='a15fc0ef64e6.key'))
```

12.1.4 Report Rendering

To build a loaded report, you need to call the `render()` function on the `StiReport` report object. For example, you need to build a report before exporting it:

app.py

```
from stimulsoft_reports.report import StiReport
from stimulsoft_reports.report.enums import StiExportFormat

report = StiReport()
```

© 2003-2024 Stimulsoft

```
report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
report.render()
report.exportDocument(StiExportFormat.PDF)
```

If you need to perform any actions after building a report using JavaScript code, simply utilize the `onAfterRender` event on the report object. More details about this are provided in the Reporting tool engine. For instance, after building a report, you may need to display the following message:

app.py

```
from stimulsoft_reports.report import StiReport

report = StiReport()
report.onAfterRender += 'afterRender'
report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
report.render()
```

render.html

```
<script>
    function afterRender(args) {
        alert("The report rendering is completed.");
    }
</script>
```

If you need to perform any actions with the report before building it using JavaScript code, just use the `onBeforeRender` event on the report object. For example, before rendering a report, you need to register JSON data:

app.py

```
from stimulsoft_reports.report import StiReport

report = StiReport()
report.onBeforeRender += 'beforeRender'
report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
report.render()
```

© 2003-2024 Stimulsoft

render.html

```
<script>
    function onBeforeRender(args) {
        var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
        dataSet.readJsonFile("Demo.json");

        var report = args.report;
        report.regData(dataSet.dataSetName, "", dataSet);
    }
</script>
```

12.1.5 Printing a Report From Code

The reporting tool allows you to print a report from code. To achieve this, you can utilize the special `print()` method on the report object.

app.py

```
report = StiReport()
report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
report.print()
```

By default, all pages of the generated report will be printed. It is possible to specify a page or range of pages to print. To achieve this, simply pass the required value as a parameter to the `print()` function. For example:

app.py

```
report = StiReport()
report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))

report.print(5)
report.print('1,3-8')
```

12.1.6 Exporting a Report From Code

The reporting tool allows you to export the generated report or dashboard to various formats. Below is a list of all available export formats for reports and dashboards:

Name	Reports	Dashboards

© 2003-2024 Stimulsoft

Document (Snapshot)	+	+
Adobe PDF	+	+
Microsoft XPS	+	-
Microsoft PowerPoint (.pptx)	+	-
HTML	+	+
HTML5	+	-
Text	+	-
Microsoft Word (.docx)	+	-
Microsoft Excel (.xlsx)	+	+
OpenDocument Writer (.odt)	+	-
OpenDocument Calc (.ods)	+	-
Comma Separated Value (.csv)	+	+
Scalable Vector Graphics (.svg)	+	+

To export a report, you should utilize the special `exportDocument()` function on the report object.

app.py

```
from stimulsoft_reports.report import StiReport
from stimulsoft_reports.report.enums import StiExportFormat

report = StiReport()
report.loadFile(url_for('static', filename='reports/SimpleList.mrt'))
report.render()
report.exportDocument(StiExportFormat.PDF)
```

Information

Exporting a report does not automatically trigger its construction. Therefore, the

© 2003-2024 Stimulsoft

loaded report template must first call the `render()` function, which initiates the pre-building of the report. For generated reports, calling the specified function is not necessary.

As arguments to the `exportDocument()` function, you should specify the required export format from the `StiExportFormat` enumeration. The available format options are as follows:

Name	Description
`StiExportFormat.DOCUMENT`	Saves the rendered document.
`StiExportFormat.PDF`	Saves to Adobe PDF.
`StiExportFormat.XPS`	Saves to XPS (XML Paper Specification).
`StiExportFormat.POWERPOINT`	Saves to Microsoft PowerPoint.
`StiExportFormat.HTML`	Saves to HTML.
`StiExportFormat.HTML5`	Saves to HTML5, using SVG markup elements.
`StiExportFormat.TEXT`	Saves to text.
`StiExportFormat.WORD`	Saves to Microsoft Word.
`StiExportFormat.EXCEL`	Saves to Microsoft Excel.
`StiExportFormat.ODT`	Saves to OpenDocument Text.
`StiExportFormat.ODS`	Saves to OpenDocument Spreadsheet.
`StiExportFormat.CSV`	Saves to CSV (Comma-Separated Values).
`StiExportFormat.IMAGE_SVG`	Saves to SVG.

After the report is exported, the resulting data stream will be transferred to the web browser for downloading as a file. The file name and MIME data type will be

© 2003-2024 Stimulsoft

detected automatically.

13 Reports.Java

YouTube

Watch the video tutorials for working with the components of the Stimulsoft Reports.JAVA product. Subscribe to the Stimulsoft channel and be the first to watch new video lessons. Questions and suggestions is recommended be left in the comments to the video.

Samples

See on our website examples for working with the Reports.JAVA components. All examples are separate projects, grouped into one solution for Visual Studio. Also, you can view and download specific examples on GitHub.

Tools for creating and editing reports:	Tools for viewing and converting reports:
› HTML5 designer	› HTML5 Viewer

13.1 Activation

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this by specifying a license key or by downloading a file with the license key. Below is an example of activating the **StiWebViewer** component.

index.jsp

```
...
<body>
   <%
      //Activation with using license code
      com.stimulsoft.base.licenses.StiLicense.setKey(
      "Your activation code...";

      //Activation with using license file
```

© 2003-2024 Stimulsoft

```
    com.stimulsoft.base.licenses.StiLicense.loadFromFile(request.getSessi
    on().getServletContext().getRealPath("/license/license.key"));
  %>
  <stiwebviewer:webviewer report="${report}" options="${options}" />
</body>
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

13.2 Java Viewer

The **StiViewerFx** component is delivered as a part of Stimulsoft Reports.JAVA. This component is used to display reports in Java applications. To run the viewer you will need to place the **StiViewerFx** component on the **Frame** and set necessary properties to it. For running the report viewer it requires the Java 1.5 platform or higher.

- Showing Reports,
- Custom Functions.

13.2.1 Showing Reports

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer. Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

Add the **StiViewerFx** component on the required component (for example on JFx):

Jfx

```
...
StiViewerFx viewerfx = new StiViewerFx(parentFrame);
fx.add(viewerfx);
```

© 2003-2024 Stimulsoft

```
...
```

where the **JFrameparentFrame** is a main Frame of the application. For better showing **StiViewerFx**, the parent component must have **BoxLayoutManager**. For loading and showing the report use the following method:

Jfx

```
...
viewerfx.getStiViewModel().loadDocumentFile(documentFile, showProgress);
...
```

where the argument **documentFile** is a file of **mdc** documents, and the boolean value **showProgress** provides the ability to define whether to show the loading progress of the document (if it is set to true then the process is displayed, if false - it is not displayed.) It is also possible to display a report in the form of a dialog box, you can use the method:

Jfx

```
...
StiViewerFx viewerfx = new StiViewerFx(parentFrame);
JDialogviewerPopup = viewerfx.createPopup(parentFrame, modal);
viewerPopup.setVisible(true);
...
```

Where the argument **parentFrame** is a frame from which the dialog is displayed, and the Boolean value **modal** - dialog modality. The method returns **JDialog**, which subsequently can make the necessary manipulations, such as resizing, changing dialog parameters, visibility, etc.

13.2.2 Custom Functions

In addition to the standard (built-in) functions, there is an ability to define your own (custom) functions. To do this, we have the list customFunctions implemented in the class **StiReport**. Before rendering a report all required functions must be added in it. Classes of user-defined functions must implement the interface **com.stimulsoft.report.StiCustomFunction**. The description of the interface **StiCustomFunction**:

- **public String getFunctionName()** – the function class should return the name of

© 2003-2024 Stimulsoft

the custom function. Register is taken into account. Do not use the names of existing built-in functions, methods, variables, reserved words as true/false/null, etc.
› **public List<Class> getParametersList()** – the function class should return a list of classes of variables used in the custom function.
› **public Object invoke(List<Object> args)** – there must be a realization of a custom function.

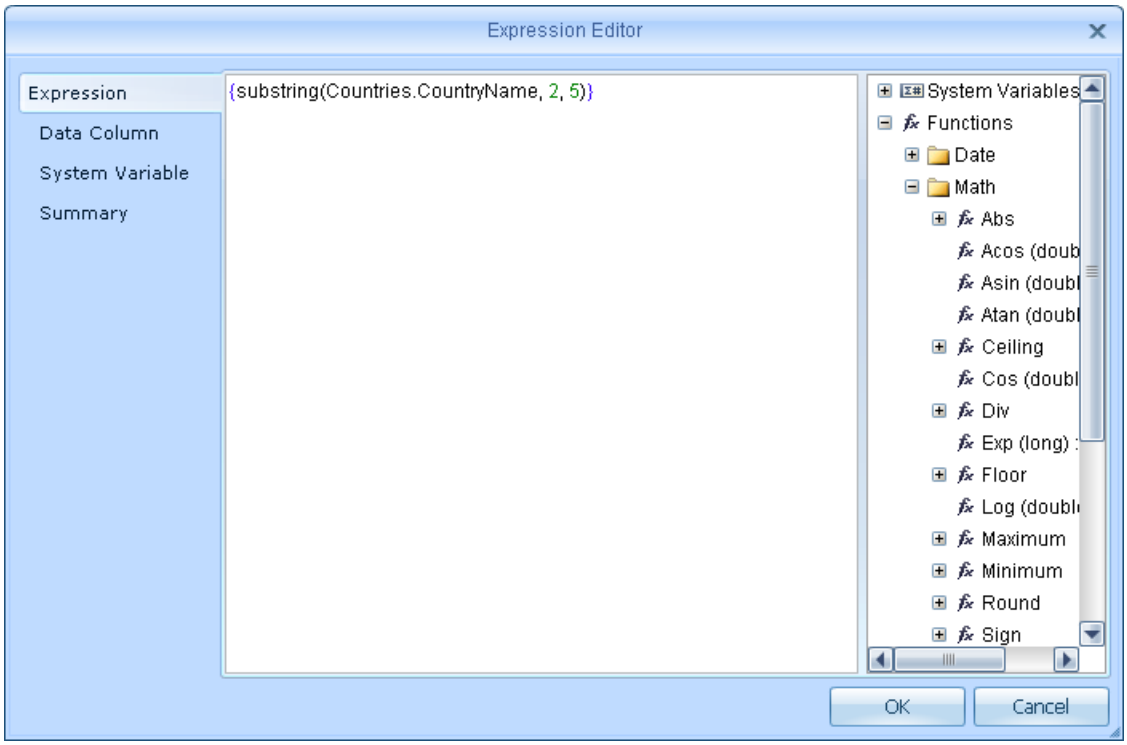
An example of using on the base of **Samples\webfx**. Suppose you need to implement a custom **substring** function. In the class **my.actions.MyRenderReportAction** write the following:

webfx

```
...
public StiReport render(StiReport report) throws IOException, StiException
{
   report.getCustomFunctions().add(new StiCustomFunction() {
      public Object invoke(List<Object> args) {
         return ((String) args.get(0)).substring((Integer)args.get(1),
         (Integer) args.get(2));
      }

      public List<Class> getParametersList() {
         return new ArrayList<Class>(Arrays.asList(String.class,
         Integer.class, Integer.class));
      }

      public String getFunctionName() {
         return "substring";
      }
   });
   return super.render(report);
}
...
```

Now you can use a custom substring function in a report:

© 2003-2024 Stimulsoft



13.3 Java HTML5 Viewer

The Java Angular Viewer will be described in this section.

- Installation
- Create a Sample Page With Report HTML5 Viewer
- Creating Project
- Description of Webviewer Tag
- Creating a Sample Page
- Template JDBC Connections

13.3.1 Installation

Installation:

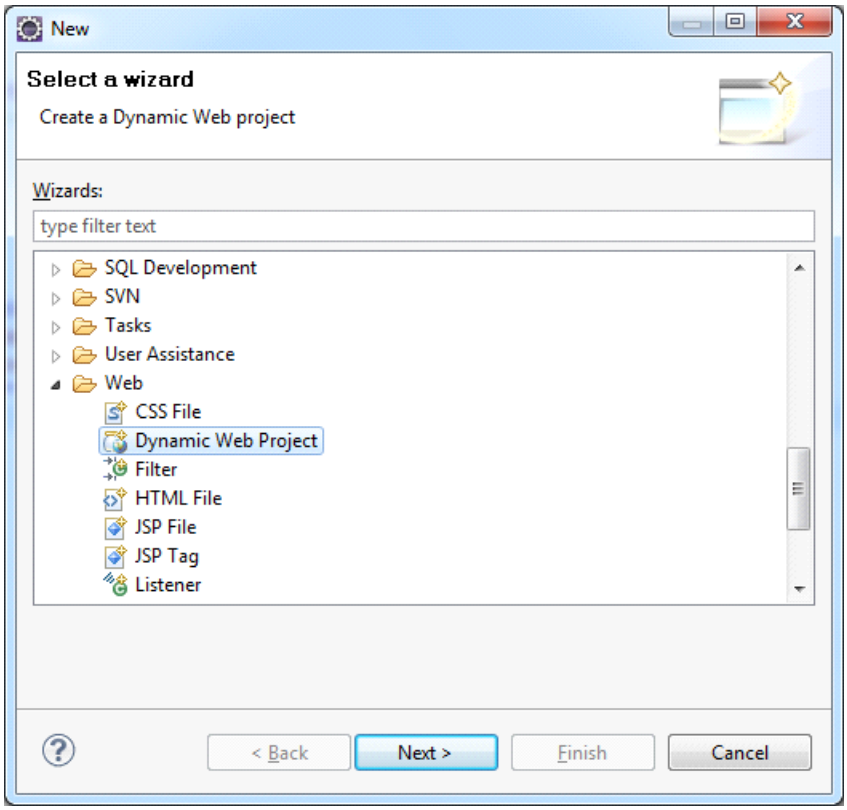
- Download and install **Java™ SE** version 1.5 or higher (for the version 1.5 jaxb-impl and jaxb-api libraries are required).
- Download and install **EclipsePlatform**.
- Download an archive with **jar** files on stimulsoft

13.3.2 Creating Project

Launch the **Eclipse IDE**, choose **File> New> Project**. In the project wizards open the Web type and in the drop-down list select **Dynamic Web Project** wizard and click

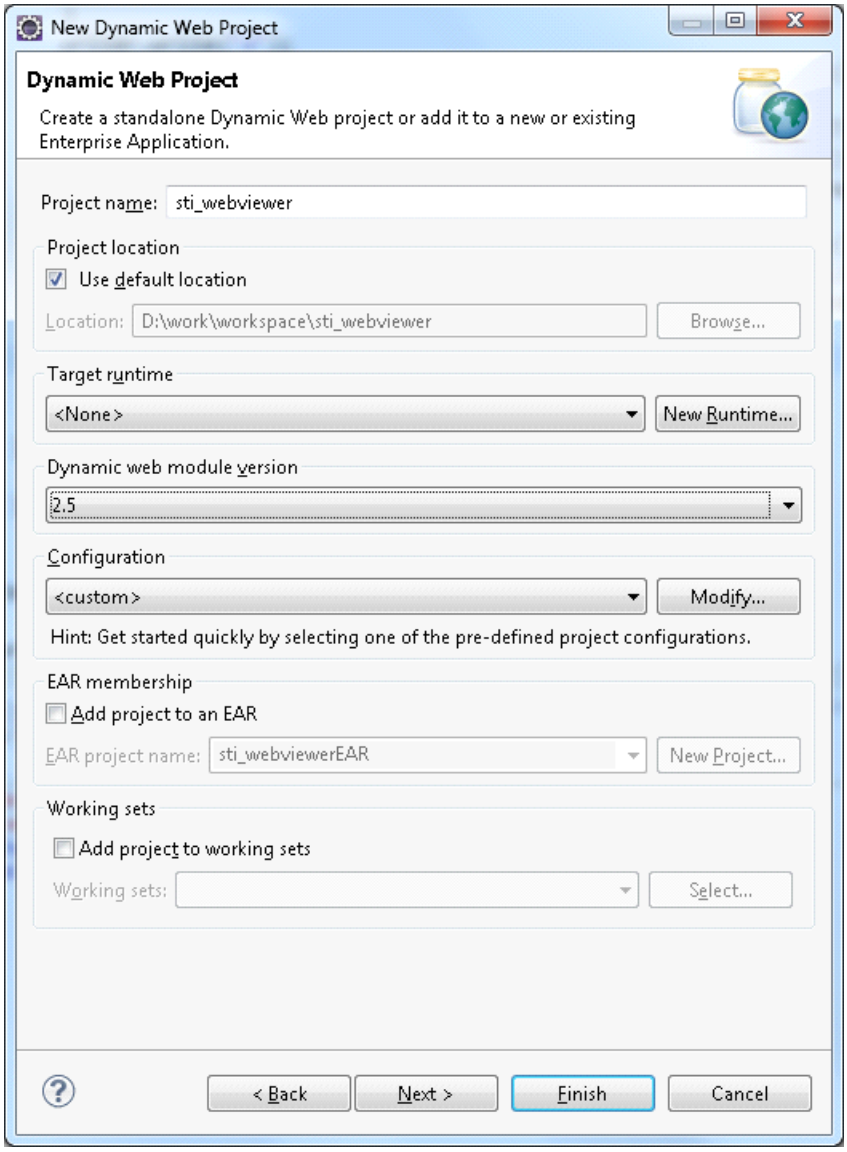
© 2003-2024 Stimulsoft

Next (Figure 1). Select dynamic Web project:



In the window opened fill in the Project name (e.g. sti_webviewer, as shown on Figure2). Then configure the web server, on which the application will run. Create a new dynamic Web project:

© 2003-2024 Stimulsoft



Target a runtime

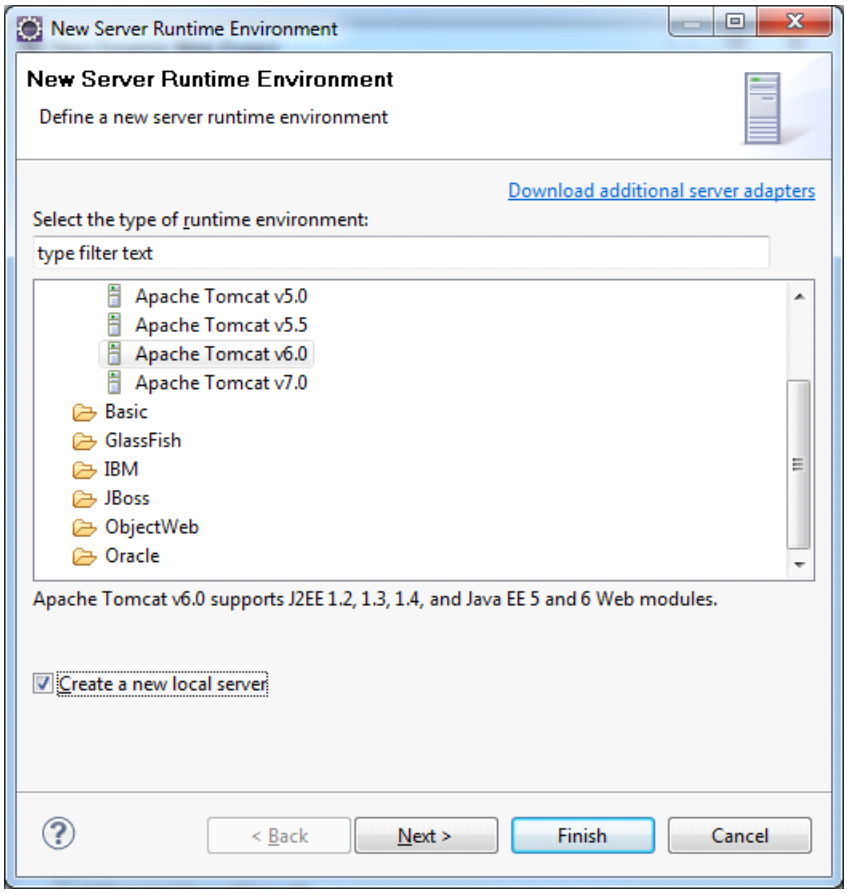
Under Target Runtime, you see <None>, as shown in Figure 1, because you haven't created a runtime yet for Apache Tomcat. Click New Runtime to open the New Target Runtime Wizard. Select Apache Tomcat of the required version from the list of available, check the Create a new local server as shown in Figure 3, then click Next.

Target a runtime

Under Target Runtime, you see <None>, as shown in Figure 1, because you haven't created a runtime yet for Apache Tomcat. Click New Runtime to open the New Target Runtime Wizard. Select Apache Tomcat of the correct version from a

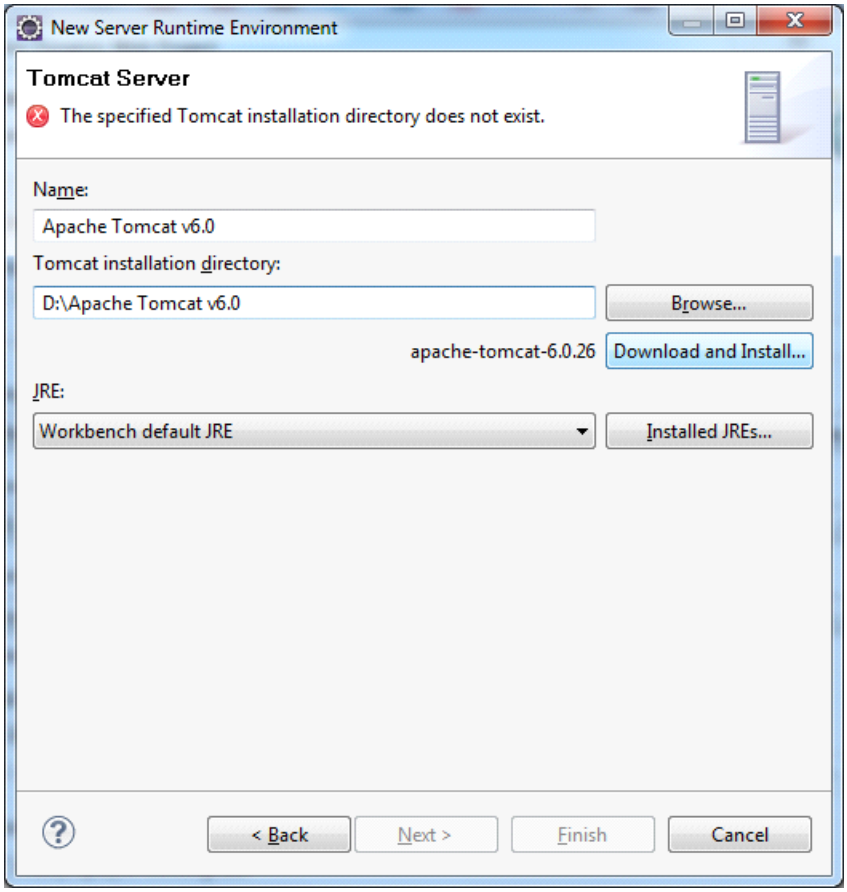
© 2003-2024 Stimulsoft

list. Check Create a new local server as shown on Figure 3, then click Next.Create a new server runtime:



Then define the Tomcat installation directory, in which Apache Tomcat is installed, or in which one needs to install it, as shown on Figure 4. If it is not installed, then click Download and Install. After all fields are specified, click Finish. Define the server location:

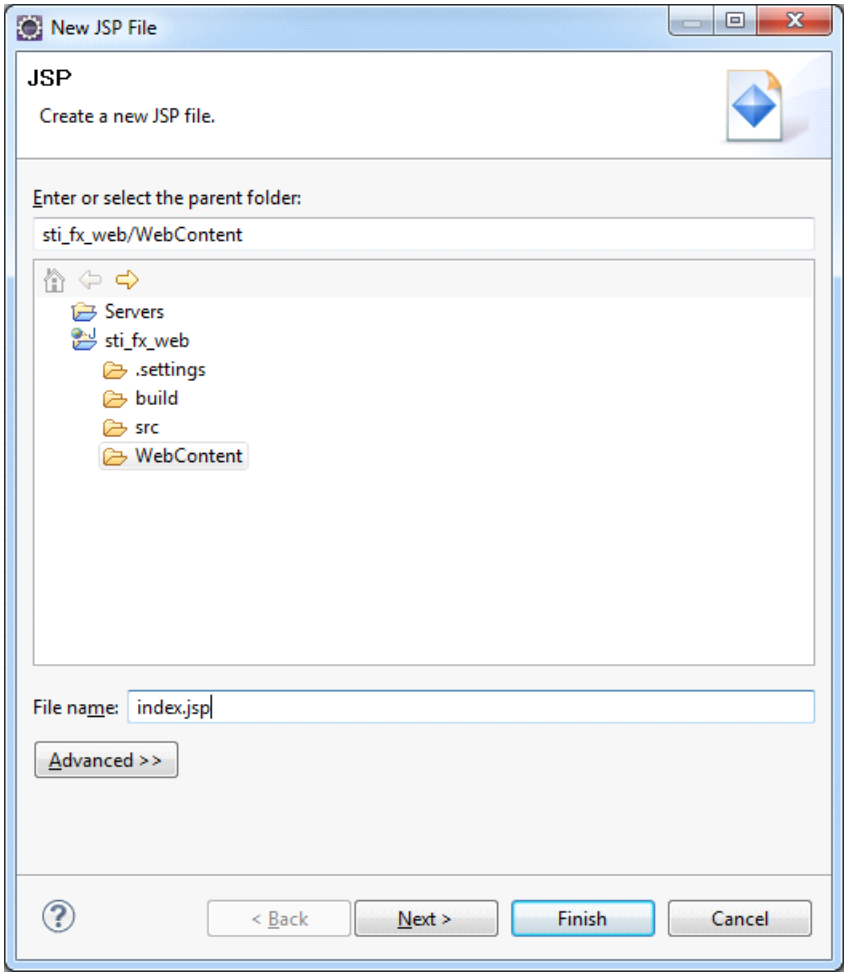
© 2003-2024 Stimulsoft



13.3.3 Creating a Sample Page

In order to verify the project and the Tomcat server, create a simple JSP and deploy it on Tomcat. To do this, one can create a new JSP, by choosing **File> New> Other**, or one can use the context menu, right-click the project name in the **Project Explorer** and select **New> JSP** file. In the next window (see Figure 5) define the directory **WebContent**, and in the **File** name write **index.jsp**. Click **Finish** to create pages using the default template:

© 2003-2024 Stimulsoft



Now open the **index.jsp** and edit it so that it displays the current date. The code page is specified in the Listing 1.

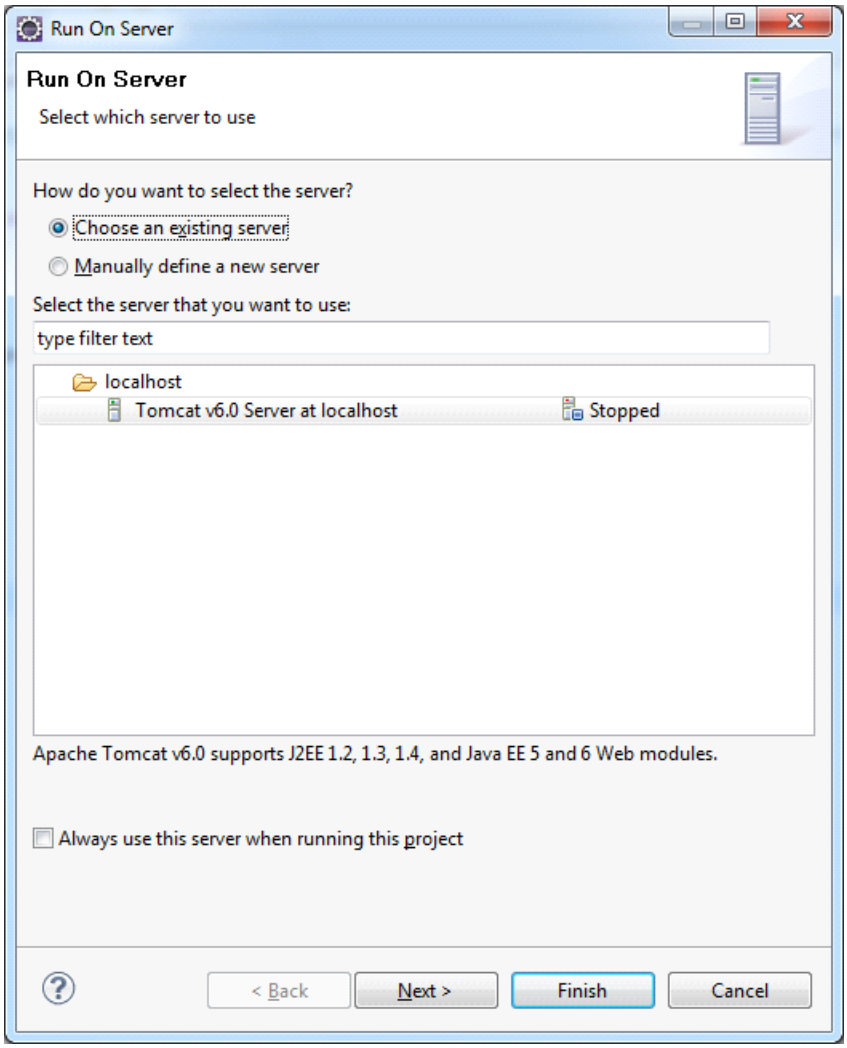
index.jsp

```
...
<!DOCTYPEhtmlPUBLIC"-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>stimulsoft reports</title>
</head>
<body>
	<%java.util.Date date = new java.util.Date();%>
	<h1>
		This index.jsp.<br>
		<%=date.toString()%>
```

© 2003-2024 Stimulsoft

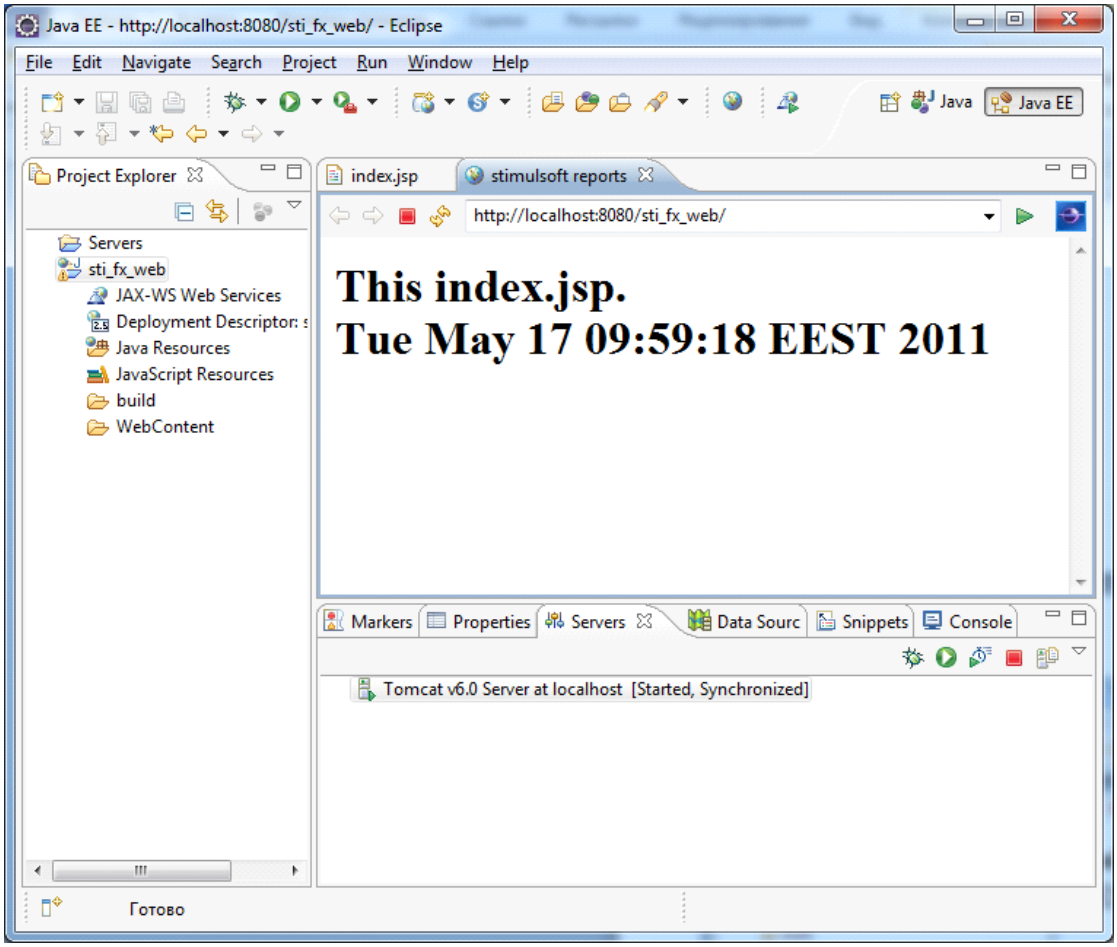
```
      </h1>
</body>
</html>
...
```

Now deploy it on the server. For this one need to use the context menu, right-click the project name, select **Run> Run as> Run** on server. Define a previously created server and click **Finish**:



As a result, you receive the following (see Figure 7). This page will be available from any browser at **http://localhost:8080/{ProjectName}** (where the **{ProjectName}** name of the created project, in our case **sti_webviewer**):

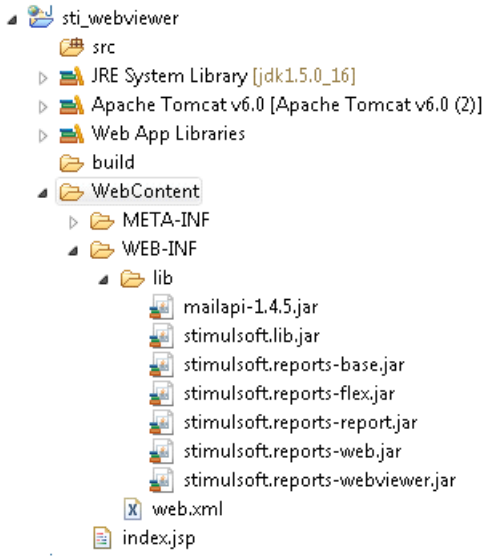
© 2003-2024 Stimulsoft



13.3.4 Create a Sample Page With Report HTML5 Viewer

Create a simple page with a report webviewer. To do this, put the following libraries into the **WebContent\WEB-INF\lib** directory: stimulsoft.lib.jar, stimulsoft.reports-base.jar, stimulsoft.reports-report.jar, stimulsoft.reports-flex.jar, stimulsoft.reports-web.jar, stimulsoft.reports-webviewer.jar . As a result, one can see the following (Figure 8):

© 2003-2024 Stimulsoft



Next, open the web.xml for editing, it should look like in Listing 2:

web.xml

```
...
<?xml version="1.0" encoding="UTF-8" ?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/
xml/ns/javaee/webapp_2_5.xsd"
	xsi:schemaLocation="http://java.sun.com/xml/ns/javaee"
	id="WebApp_ID" version="2.5">
	<display-name>sti_webviewer</display-name>
	<welcome-file-list>
		<welcome-file>index.jsp</welcome-file>
	</welcome-file-list>
	<!-- configuration, this parameter indicates the main application
directory -->
	<servlet>
		<servlet-name>StimulsoftResource</servlet-name>
		<servlet-class>com.stimulsoft.web.servlet.StiWebResourceServlet</
servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>StimulsoftResource</servlet-name>
		<url-pattern>/stimulsoft_web_resource</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>StimulsoftAction</servlet-name>
		<servlet-
class>com.stimulsoft.webviewer.servlet.StiWebViewerActionServlet</servlet-class>
	</servlet>
	<servlet-mapping>
```

© 2003-2024 Stimulsoft

```
            <servlet-name>StimulsoftAction</servlet-name>
            <url-pattern>/stimulsoft_webviewer_action</url-pattern>
      </servlet-mapping>
</web-app>
...
```

Leave unchanged the remaining **web.xml** blocks, which defines the servlets required for working. Then, edit the **index.jsp** (Listing 4).

index.jsp

```
...
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<%@page import="com.stimulsoft.report.StiReport"%>
<%@ page language="java" contentType="text/html; charset=utf-8"
      pageEncoding="UTF-8"%>
<%@ taglib uri="http://stimulsoft.com/webviewer" prefix="stiwebviewer"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Stimulsoft Reports for Java</title>
<stiwebviewer:resources />
<style type="text/css">
.t1 td {
      padding-right: 30px
}
</style>
</head>
<body>
      <%
          pageContext.setAttribute("report", new StiReport());
      %>
      <h1 align="center">My first report!</h1>
      <stiwebviewer:webviewer  report="${report}" />
</body>
</html>
...
```

It will display empty webviewer (because of empty StiReport object). Add taglib directives in the JSP. They will work with custom tags on the page.

Custom Stimulsoft tag

```
...
<%@ taglib uri="http://stimulsoft.com/webviewer" prefix="stiwebviewer"%>
...
```

© 2003-2024 Stimulsoft

Add a tag `<stiwebviewer:resources />`, tag used to load necessary resources (css & js) for webviewer, it haven't any attributes, it must be placed inside HTML <head> tag.

13.3.5 Description of Webviewer Tag

index.jsp

```
...
<stiwebviewer:webviewer  report="${report}" />
...
```

This tag contains next attributes:

- **report** [required] – StiReport object to display in webviewer;
- **options** [optional] – StiWebViewerOptions object to customize webviewer. If not present – default options are used;
- **viewerID** [optional] – String value identifier of webviewer HTML element. If more than one webviewer present in HTML page each webviewer must have different identifier.

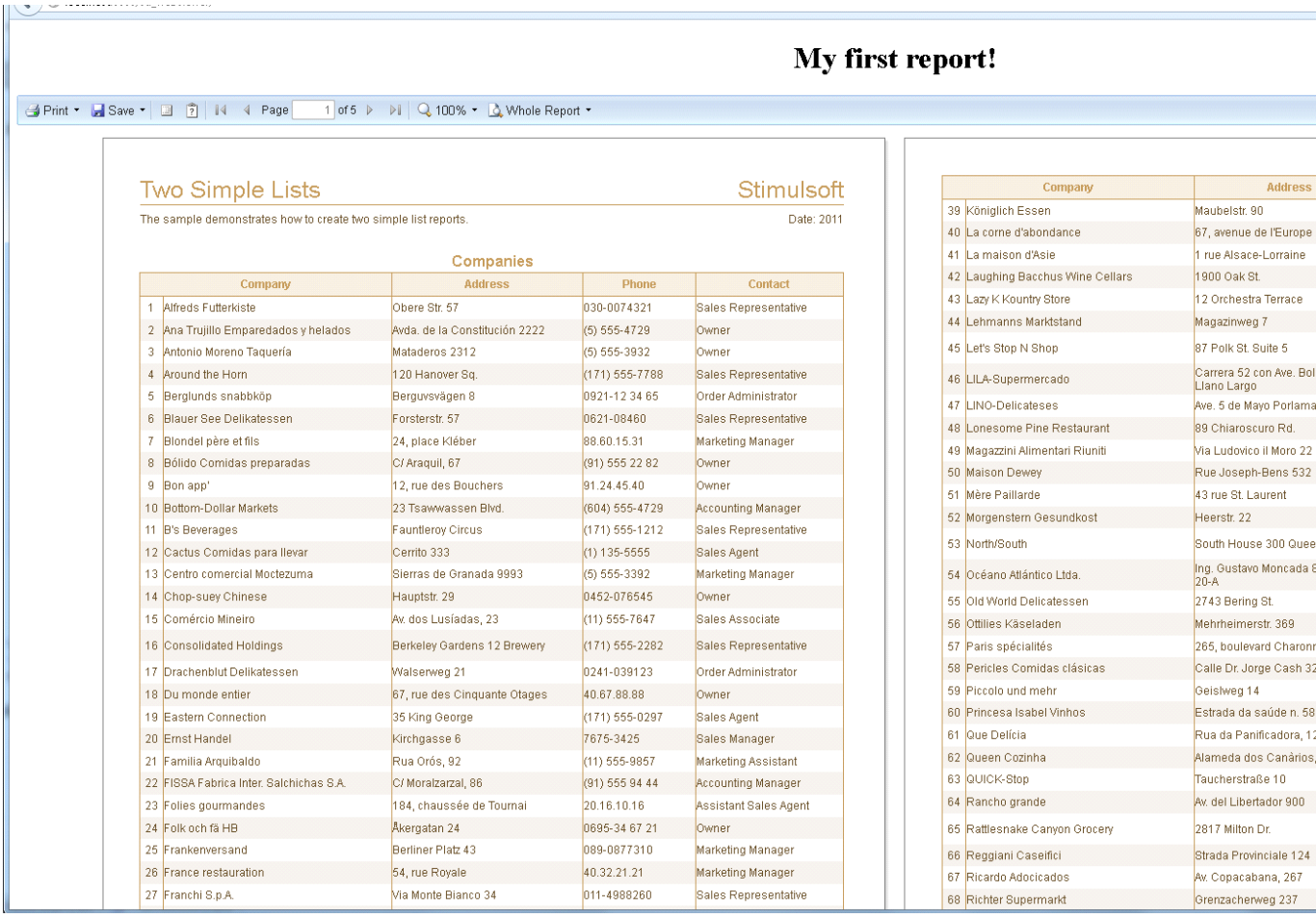
Example of usage webviewer tag (display generated (mdc) report from d:\repots \TwoSimpleLists.mdc with custom parameters)

index.jsp

```
...
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<%@page import="com.stimulsoft.webviewer.enums.StiWebViewerTheme"%>
<%@page import="com.stimulsoft.webviewer.enums.StiPagesViewMode"%>
<%@page import="com.stimulsoft.webviewer.StiWebViewerOptions"%>
<%@page import="java.io.File"%>
<%@page import="com.stimulsoft.report.StiSerializeManager"%>
<%@page import="com.stimulsoft.report.StiReport"%>
<%@ page language="java" contentType="text/html; charset=utf-8"
      pageEncoding="UTF-8"%>
<%@ taglib uri="http://stimulsoft.com/webviewer" prefix="stiwebviewer"%>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Stimulsoft Reports for Java</title>
<stiwebviewer:resources />
</head>
<body>
      <%
          StiReport report = StiSerializeManager.deserializeDocument(new
```

© 2003-2024 Stimulsoft

```
File("d:/reports/TwoSimpleLists.mdc")).getReport();
            StiWebViewerOptions options = new StiWebViewerOptions();
            options.setTheme(StiWebViewerTheme.Office2007Blue);
            options.setPagesViewMode(StiPagesViewMode.WholeReport);

            pageContext.setAttribute("report", report);
            pageContext.setAttribute("options", options);
        %>
        <h1 align="center">My first report!</h1>
        <stiwebviewer:webviewer  report="${report}" options="${options}"/>
</body>
</html>
...
```



13.3.6 Options

Webviewer have described below options:

- String viewerID - the viewerID
- StiWebViewerTheme theme - The current visual theme which is used for drawing visual elements of the webviewer.

© 2003-2024 Stimulsoft

- String width - The width of webviewer, must ends width % or px, default is "100%".
- String height - The height of webviewer, must ends width % or px, default is "100%".
- StiColor backColor - The background color, default is White.
- int countColumnsParameters - A count columns in parameters Panel, default is 2;

- String localization - A path to the localization file for the web viewer.
- boolean rightToLeft - A value which controls of output objects in the right to left mode, default is **false**.
- boolean scrollbarsMode - A value which indicates that the web viewer will show the report with, default is **false**.
- boolean menuAnimation - A value which indicates that menu animation is enabled, default is **true**.
- StiShowMenuMode menuShowMode - The mode that shows menu of the viewer, default id StiShowMenuMode.Click.
- StiPrintDestination menuPrintDestination - The default mode of the report print destination, default is StiPrintDestination.Default.

- StiPagesViewMode pagesViewMode - The mode of showing a report in the web viewer - one page or the whole report, default is StiPagesViewMode.OnePage.
- int menuZoom - The report showing zoom. The default value is **100**.
- StiContentAlignment pageAlignment - The alignment of the web viewer page, default is StiContentAlignment.Center.
- boolean pageShowShadow - A value which indicates that the shadow of the page will be displayed in the webviewer, default is **true**.
- StiColor pageBorderColor - A color of the page border, default is **Gray**.
- boolean bookmarksVisible - A visibility of the Toolbar of the web viewer, default is **true**;

- boolean bookmarksPrint - A value which allows printing report bookmarks, default is **false**;
- int bookmarksTreeWidth - A width of the bookmarks tree in the web viewer, default is 180.
- boolean toolbarVisible - A value which indicates that report bookmarks will be shown in the web viewer, default is **true**;
- StiColor toolbarBackgroundColor - A color of the toolbar background.
- StiColor toolbarFontColor - A color of the toolbar texts.
- String toolbarFontFamily - A value which indicates which font family will be

© 2003-2024 Stimulsoft

used for drawing texts in the webviewer, default is Arial;

- StiContentAlignment toolbarAlignment - The alignment of the web viewer toolbar, default is StiContentAlignment.Default;
- boolean toolbarButtonCaptions - A value which allows displaying or hiding toolbar buttons captions, default is **false**;
- boolean toolbarMenuCaptions - A value which allows displaying or hiding toolbar menu captions, default is **true**;
- boolean showCurrentPageControl - A visibility of the current page control in the toolbar of the web viewer, default is **true**;
- boolean showButtonPrint - A visibility of the Print button in the toolbar of the web viewer, default is **true**;
- boolean showButtonSave - A visibility of the Save button in the toolbar of the web viewer, default is **true**;

- boolean showButtonBookmarks - A visibility of the Parameters button in the toolbar of the web viewer, default is **true**;
- boolean showButtonParameters - A visibility of the Parameters button in the toolbar of the web viewer, default is **true**;
- boolean showButtonFirstPage - A visibility of the First Page button in the toolbar of the web viewer, default is **true**;
- boolean showButtonPreviousPage - A visibility of the Prev Page button in the toolbar of the web viewer, default is **true**;
- boolean showButtonNextPage - A visibility of the Next Page button in the toolbar of the web viewer, default is **true**;
- boolean showButtonLastPage - A visibility of the Last Page button in the toolbar of the web viewer, default is **true**;

- boolean showButtonZoom - A visibility of the Zoom control in the toolbar of the webviewer, default is **true**;
- boolean showButtonViewMode - visibility of the View Mode button in the toolbar of the web viewer, default is **true**;
- boolean showExportDialog - A value which allows to display the export dialog, or to export with the default settings, default is **true**;
- boolean showExportToDocument - A value which indicates that the user can save the report from the web viewer to the report document file, default is **true**;
- boolean showExportToPdf - A value which indicates that the user can save the report from the web viewer to the PDF format, default is **true**;
- boolean showExportToXps - A value which indicates that the user can save the report from the web viewer to the XPS format, default is **true**;

© 2003-2024 Stimulsoft

- boolean showExportToHtml - A value which indicates that the user can save the report from the web viewer to the HTML format, default is **true**;
- boolean showExportToText - A value which indicates that the user can save the report from the web viewer to the TEXT format, default is **true**;
- boolean showExportToRtf - A value which indicates that the user can save the report from the web viewer to the RTF format, default is **true**;
- boolean showExportToWord2007 - A value which indicates that the user can save the report from the web viewer to the Word 2007-2010 format, default is **true**;
- boolean showExportToExcel - A value which indicates that the user can save the report from the web viewer to the Excel BIFF format, default is **true**;
- boolean showExportToExcelXml - A value which indicates that the user can save the report from the web viewer to the ExcelXML format, default is **true**;

- boolean showExportToExcel2007 - A value which indicates that the user can save the report from the web viewer to the Excel 2007-2010 format, default is **true**;
- boolean showExportToCsv - A value which indicates that the user can save the report from the web viewer to the CSV format, default is **true**;
- boolean showExportToXml - A value which indicates that the user can save the report from the web viewer to the XML format, default is **true**;
- boolean showExportToSylk - A value which indicates that the user can save the report from the web viewer to the Sylk format, default is **true**;
- boolean showExportToImageBmp - A value which indicates that the user can save the report from the web viewer to the BMP image format, default is **true**;
- boolean showExportToImageJpeg - A value which indicates that the user can save the report from the web viewer to the JPEG image format, default is **true**;

- boolean showExportToImagePcx - A value which indicates that the user can save the report from the web viewer to the PCX image format, default is **true**;
- boolean showExportToImagePng - A value which indicates that the user can save the report from the web viewer to the PNG image format, default is **true**;
- boolean showExportToImageSvg - A value which indicates that the user can save the report from the web viewer to the SVG image format, default is **true**;
- boolean showExportToImageSvgz - A value which indicates that the user can save the report from the web viewer to the SVGZ image format, default is **true**;
- int refreshTimeout - A value wich indicates timeout in minutes for execute dummy request to avoid end session, default is 0 (disabled);

© 2003-2024 Stimulsoft

13.3.7 Template JDBC Connections

Template

```
...
jdbc.driver={myDriver};
jdbc.url={myConnectionUrl};
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

An example for a SQLServer

```
...
jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver;
jdbc.url= jdbc:sqlserver://[serverName[\instanceName][:portNumber]]
[;property=value[;property=value]];
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

http://msdn.microsoft.com/en-us/library/ms378428

An example for a Oracle

```
...
jdbc.driver=oracle.jdbc.driver.OracleDriver
jdbc.url=jdbc:oracle:thin:@[HOST][:PORT]:SID;
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

http://www.orafaq.com/wiki/JDBC

An example for a postgresql

```
...
jdbc.driver= org.postgresql.Driver
jdbc.url= jdbc:postgresql://[host]:[port]/[database]
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

http://jdbc.postgresql.org/documentation/80/connect.html

© 2003-2024 Stimulsoft

13.4 HTML5 Designer

The Java designer will be described in this section.

- Installation and Description HTML5 Designer,
- Template JDBC Coonections

13.4.1 Installation and Description HTML5 Designer

Create a sample page with report webdesigner

Create a simple page with a report webdesigner. To do this, put the following libraries into the **WebContent\WEB-INF\lib** directory: stimulsoft.lib.jar, stimulsoft.reports-base.jar, stimulsoft.reports-report.jar, stimulsoft.reports-flex.jar, stimulsoft.reports-web.jar, stimulsoft.reports-webdesigner.jar .

Next, edit **web.xml** it should look like in Listing 1:

web.xml

```
...
<?xml version="1.0" encoding="UTF-8" ?>
  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://
    java.sun.com/xml/ns/javaee/webapp_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee"
    id="WebApp_ID" version="2.5">
  <display-name>sti_webdesigner</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
    <!-- configuration, this parameter indicates the main application
    directory -->
  <servlet>
    <servlet-name>StimulsoftResource</servlet-name>
    <servlet-class>com.stimulsoft.web.servlet.StiWebResourceServlet</
    servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>StimulsoftResource</servlet-name>
    <url-pattern>/stimulsoft_web_resource/*</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>StimulsoftDesignerAction</servlet-name>
    <servlet-
    class
    >com.stimulsoft.webdesigner.servlet.StiWebDesignerActionServlet</
    servlet-class>
  </servlet>
  <servlet-mapping>
```

© 2003-2024 Stimulsoft

```
    <servlet-name>StimulsoftDesignerAction</servlet-name>
    <url-pattern>/stimulsoft_webdesigner_action</url-pattern>
  </servlet-mapping>
</web-app>
...
```

Leave unchanged the remaining web.xml blocks, which defines the servlets required for working. Then, edit the index.jsp (Listing 2).

index.jsp

```
...
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<%@page import="java.io.FileOutputStream"%>
<%@page  import="java.io.FileInputStream"%>
<%@page  import="com.stimulsoft.report.utils.data.StiDataColumnsUtil"%>
<%@page
import="com.stimulsoft.report.dictionary.StiDataColumnsCollection"%>
<%@page  import="com.stimulsoft.report.dictionary.StiDataColumn"%>
<%@page  import="com.stimulsoft.report.utils.data.StiSqlField"%>
<%@page
import="com.stimulsoft.report.dictionary.dataSources.StiDataTableSource"%>
<%@page  import="com.stimulsoft.report.utils.data.StiXmlTable"%>
<%@page
import="com.stimulsoft.report.utils.data.StiXmlTableFildsRequest"%>
<%@page  import="com.stimulsoft.webdesigner.StiWebDesigerHandler"%>
<%@page  import="com.stimulsoft.webdesigner.StiWebDesignerOptions"%>
<%@page
  import="com.stimulsoft.report.dictionary.databases.StiXmlDatabase"%>
<%@page  import="java.io.File"%>
<%@page  import="com.stimulsoft.report.StiSerializeManager"%>
<%@page  import="com.stimulsoft.report.StiReport"%>
<%@page  language="java" contentType="text/html; charset=utf-8"
  pageEncoding="UTF-8"%>
<%@taglib uri="http://stimulsoft.com/webdesigner" prefix="stiwebdesigner"%
>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Stimulsoft Webdesigner for Java</title>
  <style type="text/css">
  </style>
</head>
<body>
  <%
    final String reportPath =
    request.getSession().getServletContext().getRealPath("/reports/
    Master-Detail.mrt");
    final String xmlPath =
    request.getSession().getServletContext().getRealPath("/data/
    Demo.xml");
    final String xsdPath =
    request.getSession().getServletContext().getRealPath("/data/
    Demo.xsd");
```

```
final String savePath =
request.getSession().getServletContext().getRealPath("/save/");

StiWebDesignerOptions options = new StiWebDesignerOptions();

StiWebDesigerHandler handler = new StiWebDesigerHandler(){
  public StiReport getEditedReport(HttpServletRequest request){
    try{
      StiReport report = StiSerializeManager.deserializeReport(new
      File(reportPath));
      report.getDictionary().getDatabases().add(new
      StiXmlDatabase("Demo", xsdPath, xmlPath));
      return report;
    } catch (Exception e){
        e.printStackTrace();
    }
    return null;
  }

  public void onOpenReportTemplate(StiReport report,
  HttpServletRequest request){
    report.getDictionary().getDatabases().add(new
    StiXmlDatabase("Demo", xsdPath, xmlPath));
  }

  public void onNewReportTemplate(StiReport report,
  HttpServletRequest request){
    report.getDictionary().getDatabases().add(new
    StiXmlDatabase("Demo", xsdPath, xmlPath));
    try{
      // In new report if you want to use wizard, you must populate
      report with datasources
      StiXmlTableFildsRequest tables =
      StiDataColumnsUtil.parceXSDSchema(new
      FileInputStream(xsdPath));
      for (StiXmlTable table : tables.getTables()){
        StiDataTableSource tableSource = new
        StiDataTableSource("Demo." + table.getName(),
        table.getName(), table.getName());
        tableSource.setColumns(new StiDataColumnsCollection());
        for (StiSqlField field : table.getColumns()){
          StiDataColumn column = new StiDataColumn(field.getName(),
          field.getName(), field.getSystemType());
          tableSource.getColumns().add(column);
        }
        tableSource.setDictionary(report.getDictionary());
        report.getDictionary().getDataSources().add(tableSource);
      }
    } catch (Exception e){
      e.printStackTrace();
    }
  }

  public void onSaveReportTemplate(StiReport report, StiRequestParams
  requestParams, HttpServletRequest request) {
    try {
      FileOutputStream fos = new FileOutputStream(savePath +
```

© 2003-2024 Stimulsoft

```
            requestParams.designer.fileName);
            if (requestParams.designer.password != null) {
               StiSerializeManager.serializeReport(report, fos,
               requestParams.designer.password);
            } else {
               StiSerializeManager.serializeReport(report, fos, true);
            }
            fos.close();
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
   };
   pageContext.setAttribute("handler", handler);
   pageContext.setAttribute("options", options);
%>

<stiwebdesigner:webdesigner
   handler="${handler}" options="${options}" />
</body>
</html>
...
```

Add taglib directives in the JSP. They will work with custom tags on the page.

Listing 3. Custom Stimulsoft tag

index.jsp

```
...
<%@ taglib uri="http://stimulsoft.com/webdesigner"
prefix="stiwebdesigner"%>
...
```

Add a tag `<stiwebdesigner:resources />`, tag used to load necessary resources (css & js) for webdesigner, it haven't any attributes, it must be placed inside HTML <head> tag.

Description of webdesigner tag

web.xml

```
...
<stiwebdesigner:webdesigner handler="${handler}" report="${report}" />
...
```

© 2003-2024 Stimulsoft

This tag contains next attributes:

- **handler** [required] – com.stimulsoft.webdesigner.StiWebDesigerHandler object to handle webdesigner;
- **options** [optional] – StiWebdesignerOptions object to customize webdesigner. If not present – default options are used;
- **designerID** [optional] – String value identifier of webdesigner HTML element. If more than one webdesigner present in HTML page each webdesigner must have different identifier.

Description of StiWebDesigerHandler

To handle designer events, class that implement **StiWebDesigerHandler** must be created and setup in stiwebdesigner tag. Occured on opening {@link StiReport}.

- **public StiReport getEditedReport(HttpServletRequest request)**;
  Occurred on loading webdesinger. Here must present implementation of loading report and population it with Database\Data sources (if required).

- **public void onOpenReportTemplate(StiReport report, HttpServletRequest request)**;
  Occurred on opening StiReport. Method intended for populate report with Database\Data sources (if required).

- **public void onNewReportTemplate(StiReport report, HttpServletRequest request);**
  Occurred on new StiReport. Method intended for populate report with Database \Data sources (if required).
  In new report if you want to use wizard, you must populate report with datasources.

- **public void onSaveReportTemplate(StiReport report, String reportName, HttpServletRequest request);**
  Occurred on save StiReport. Method must implement saving StiReport.

13.4.2 Template JDBC Coonections

***.mrt**

© 2003-2024 Stimulsoft

```
...
jdbc.driver={myDriver};
jdbc.url={myConnectionUrl};
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

An example for a **SQLServer**:

***.mrt**

```
...
jdbc.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver;
jdbc.url= jdbc:sqlserver://[serverName[\instanceName][:portNumber]]
[;property=value[;property=value]];
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

http://msdn.microsoft.com/en-us/library/ms378428

An example for a **Oracle**:

***.mrt**

```
...
jdbc.driver=oracle.jdbc.driver.OracleDriver;
jdbc.url=jdbc:oracle:thin:@[HOST][:PORT]:SID;
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

http://www.orafaq.com/wiki/JDBC

An example for a **POSTGreSQL**:

***.mrt**

```
...
jdbc.driver= org.postgresql.Driver
jdbc.url= jdbc:postgresql://[host]:[port]/[database]
jdbc.username={myUserName };
jdbc.password={ myUserPassword };
...
```

© 2003-2024 Stimulsoft

http://jdbc.postgresql.org/documentation/80/connect.html

14 Reports.WPF

YouTube

Watch the video tutorials for working with the components of the Stimulsoft Reports.WPF product. Subscribe to the Stimulsoft channel and be the first to watch new video lessons. Questions and suggestions is recommended be left in the comments to the video.

Samples

See on our website examples for working with the Reports.WPF components. All examples are separate projects, grouped into one solution for Visual Studio. Also, you can view and download specific examples on GitHub.

- Wpf Viewer
- Activation

14.1 Activation

YouTube

Watch videos which show how to activate the WPF components. Subscribe to the Stimulsoft channel to find out about the new video lessons uploaded. Leave your questions and suggestions in the comments to the video.

After purchasing a Stimulsoft product, you need to activate the license for the components you are using. You can do this done in various ways. Below is an example of activating the **WPF** component.

MainWindows.xaml..cs

© 2003-2024 Stimulsoft

```
...
public partial class MainWindow : Window
{
   public MainWindow()
   {
      //Activation with using license code
      Stimulsoft.Base.StiLicense.Key = "Your activation code...";

      //Activation with using license file
      Stimulsoft.Base.StiLicense.LoadFromFile("license.key");

      //Activation from byte array
      Stimulsoft.Base.StiLicense.LoadFromBytes(bytes);

      //Activation from stream
      Stimulsoft.Base.StiLicense.LoadFromStream(stream);

      //Activation from assembly
      Stimulsoft.Base.StiLicense.LoadFromEntryAssembly(assembly,
      "stimulsoft-license.key");

      InitializeComponent();
   }
}
...
```

You can get a license key or download a file with a license key in the user's account. To log in to your account, please use the username and password specified when purchasing the product.

14.2 Wpf Viewer

The **StiWpfViewerControl** component is used to view reports in **Reports.WPF**. The component can show a report, zoom, save rendered reports to various formats, print reports, send them to a recipient via Email.

- How to Show Report,
- Dot-Matrix Viewer for Wpf

14.2.1 How to Show Report

Notice

When a report is assigned to a viewer component, it is automatically generated. You only need to call the `Report.Render()` method if you want to perform specific actions with the rendered report before it is displayed in the viewer.

© 2003-2024 Stimulsoft

Likewise, when using the compilation mode, you need to call the `Report.Compile()` method only if you have actions to perform with the compiled report before it is built and shown in the viewer.

Just call one method to show a report:

C#

```
...
StiReport report = new StiReport();
report.Load("report.mrt");
report.ShowWithWpf();
...
```

VB.NET

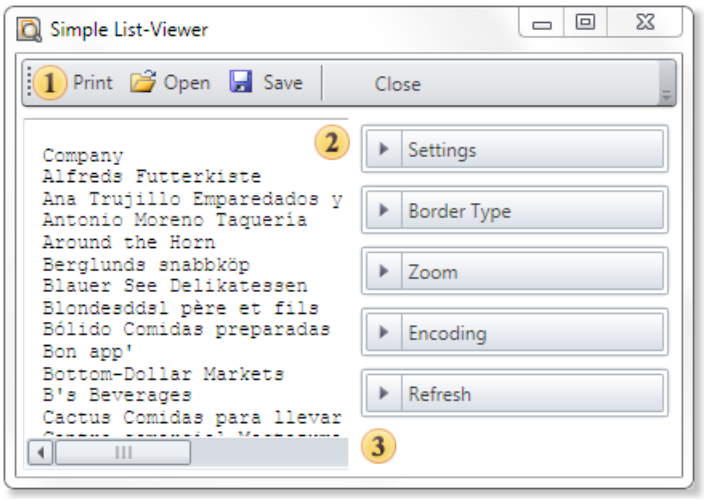
```
...
Dim Report As StiReport = New StiReport()
Report.Load("report.mrt")
Report.ShowWithWpf()
...
```

If the report was not rendered before showing, the **ShowWithWpf** method will render a report using the **RenderWithWpf** method.

14.2.2 Dot-Matrix Mode of Wpf Viewer

The **Dot-matrix** viewer is designed to preview the report before printing it on dot matrix printer. The Dot matrix printer is used to print only the text and characters of pseudographics. Accordingly the viewer displays only the text and borders of objects as pseudographics characters. The picture below shows the Dot-matrix viewer dialog box:

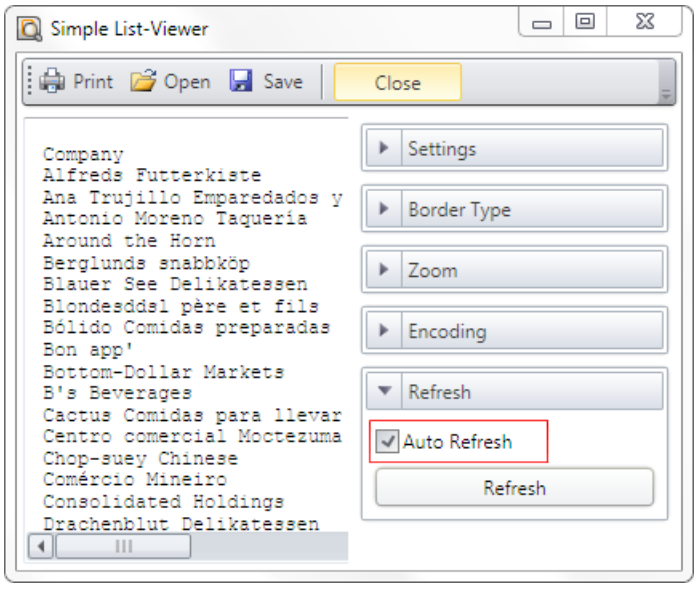
© 2003-2024 Stimulsoft



1. The **Dot-Matrix** viewer toolbar.
2. The panel displays the text of a report
3. The options bar of a report.

Dot-Matrix Viewer Settings for WPF

The Dot-Matrix viewer can be configured from code using static properties. Depending on the value of the static properties in the Dot-matrix viewer, these or that parameters will be specified. For example, the AutoRefresh property. The picture below shows the Dot-matrix viewer dialog box:



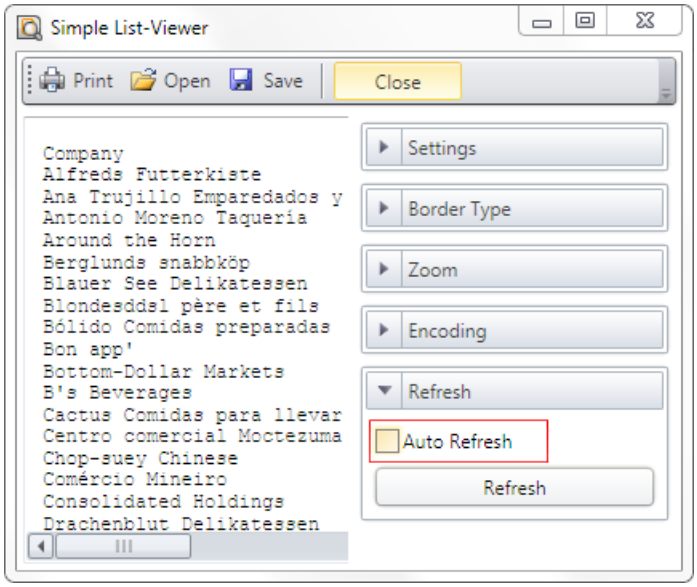
© 2003-2024 Stimulsoft

As can be seen on the picture above, the AutoRefresh property is enabled. This means that the AutoRefresh static property of the Dot-matrix viewer is set to true. If the AutoRefresh static property is set to false, then the AutoRefresh property in the Dot-matrix viewer is disabled. Add the following code into the project code:

C#

```
...
StiOptions.Viewer.DotMatrix.AutoRefresh = false;
...
```

Thus, the **AutoRefresh** property will be disabled. The picture below shows the **Dot-matrix** viewer dialog box with disabled auto refresh function:



Most parameters can be set using the static properties.

DotMatrix and Escape Codes

For inserting the escape sequence to text the commands that may look like <#command> should be used as seen in the code sample below:

```
Normal text <#b> Bold text <#/b><#i> Italic text <#/i> Again normal text
```

© 2003-2024 Stimulsoft

Also commands of selecting bold, italic or underlined text are automatically inserted depending on the style of the text box font. When printing to matrix printer and exporting to text format these commands are changed on appropriate escape sequences.
The **StiEscapeCodesCollection** is used for this process. It is inherited from the Hashtable class. This is a collection of "key-value" pairs where the key is the command and value is the escape-sequence. For different types of printers different collections with different set of command can be defined. Collections are stored in the **StiOptions.Export.Txt.EscapeCodesCollectionList** static variable. By default, the following collections will be created: "None", "EpsonFX", "Oki ML92/93". The "None" collection is empty and used to output the text without escape codes.

Command/Collection	EpsonFX	Oki ML92/93
b	ESC E	ESC T
/b	ESC F	ESC I
i	ESC 4	
/i	ESC 5	
u	ESC -1	ESC H
/u	ESC -0	ESC D
sup	ESC S0	ESC J
/sup	ESC T	ESC K
sub	ESC S1	ESC L
/sub	ESC T	ESC M
condensed	0x0F	0x1d
/condensed	0x12	0x1e
elite	ESC M	0x1c
pica	ESC P	0x1e
doublewidth	ESC W1	0x1f
/doublewidth	ESC W0	0x1e

It is possible to add new collections of change the existing ones. The selection of the required collection is done by the name. If the collection with the name is not found

© 2003-2024 Stimulsoft

then the "None" collection is used. The collection name can be selected from the DotMatrixViewer settings and passed as an option to the exporting and printing methods.

15 Engine

In this chapter you can read questions related to the capabilities of the report engine.

- Data
- Exports
- Report Inheritance
- Scripts
- Right to Left

15.1 Data

YouTube

Watch the video tutorials for working with the components of the Stimulsoft product. Subscribe to the Stimulsoft channel and be the first to watch new video lessons. Questions and suggestions is recommended be left in the comments to the video.

- Business Objects in Net, Web
- Working with OData Using Business Objects

15.1.1 Business Objects in Net, Web

Business Object is a data type, which is a set of objects related to each other, using what it is possible to present data in various structures: tables, lists, arrays, etc. These data can be passed to a reporting tool based on them the report can be rendered. Business Objects are created, registered and passed to the report generator from code.

Filling the Business Objects manually in .NET

© 2003-2024 Stimulsoft

This example creates a report with a business object. First we need to create the structure of the business object. Below is a sample code to create a business object class:

C#

```
...
public class MyObject
{
   public class Category
   {
      public int number;
      public int Number
      {
         get
         {
            return number;
         }
      }

      public string name;
      public string Name
      {
         get
         {
            return name;
         }
      }

      public string description;
      public string Description
      {
         get
         {
            return description;
         }
      }
   }

   public Category[] list = null;
   public Category[] List
   {
      get
      {
         return list;
      }
   }
}
...
```

Now, you should populate the business object. Below is an example of code to populate a Business Object is with data:

© 2003-2024 Stimulsoft

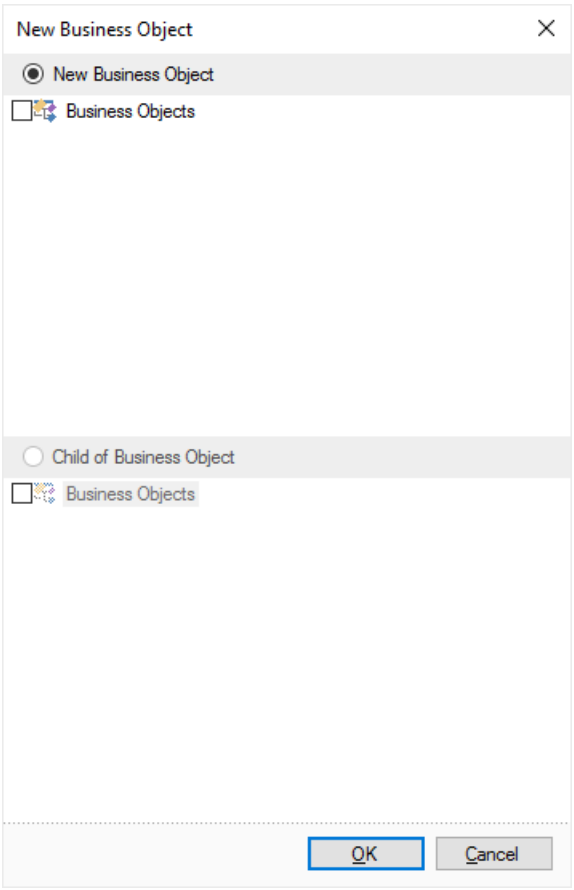
C#

```
...
MyObject obj = new MyObject();
obj.list = new MyObject.Category[2];

MyObject.Category c1 = new MyObject.Category();
c1.number = 1;
c1.name = "Cat1";
c1.description = "desc for n1";

MyObject.Category c2 = new MyObject.Category();
c2.number = 2;
c2.name = "Cat2";
c2.description = "desc for n2";

obj.list[0] = c1;
obj.list[1] = c2;

StiReport mainreport = new StiReport();
mainreport.RegBusinessObject("MyObject", obj);
mainreport.Design();
...
```

Using Business Object in Report

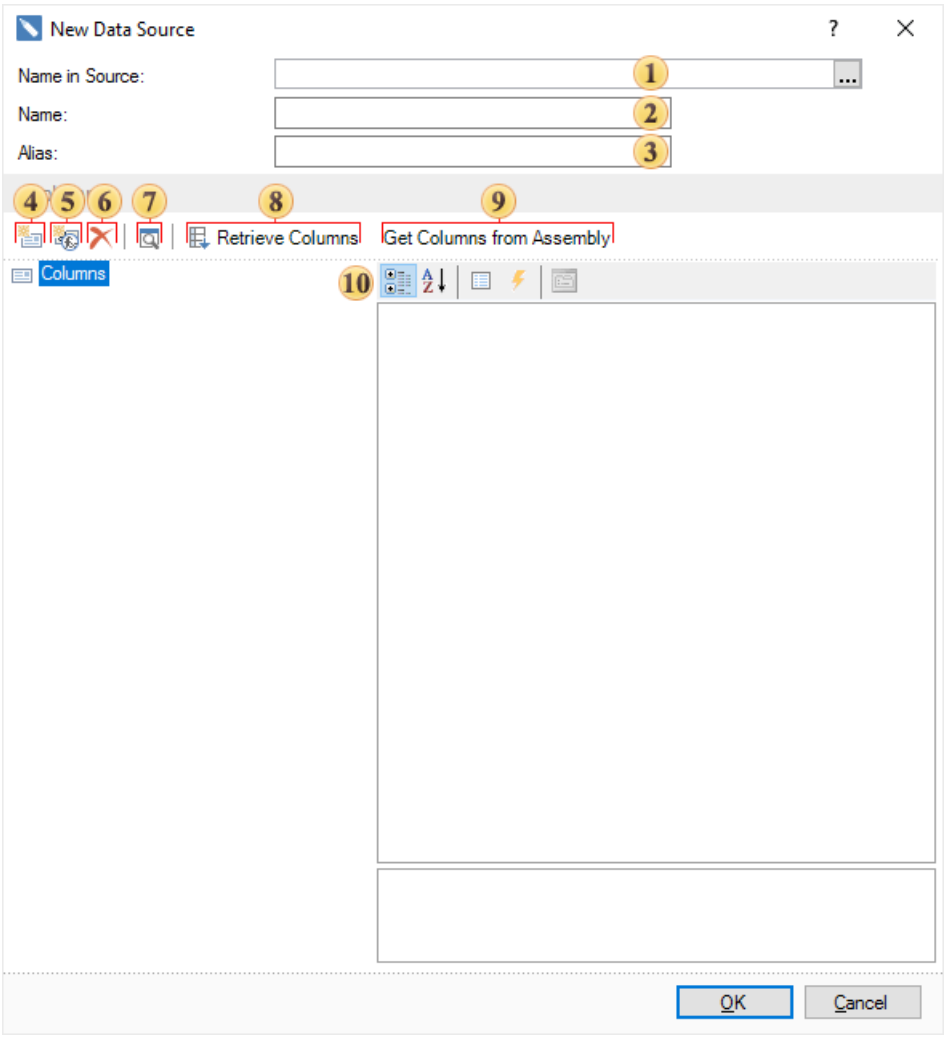
After that, the business object is created, filled with data, registered and passed to the reporting tool. In order to create a report in the designer using business objects, you should create a data description in the report dictionary. To do this, select MyObject (created Business Object) in the report dictionary in and choose New Business Object... from the context menu or the menu New Item. After selecting this command, the window will open a New Business Object, in which you should specify the Child Business Object and select lists of data. The picture below shows the dialog New Business Object.

© 2003-2024 Stimulsoft



After you click Ok, you will be shown the second dialog box form of the New Business Object, where you can change the detail business object. The picture below shows the second dialog box form of the New Business Object.

© 2003-2024 Stimulsoft



1 The field Category displays the category name. When you create a business object the field is not editable and is purely informative. Also, it may be empty, as in this case.

2 The field Name is used to specify the name of the business object. This field is always available for editing, and, in this case, the name List is used.

3 The field Alias specifies an alias of the business object. This field is always available for editing, and, in this case, the name List is used.

4 The button New Column. Pressing it a new data column will be created in the business object. It should be noted that the data column created this way is a virtual data column and it does not contain actual data.

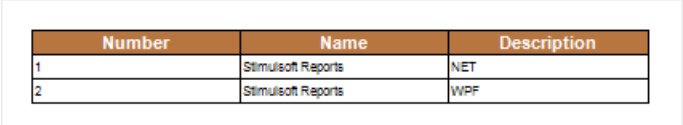
5 The button New Calculated Column is used to insert a new calculated column into the business object.

6 The button Delete is used to delete selected data columns. If you select a bookmark Columns, then all the columns which are in the tab will be deleted.

© 2003-2024 Stimulsoft

7 The button Retrieve Columns is used to get the data column from the business object.
8 The button Get Columns from Assembly will open the dialog Open Assembly, in which you may choose an assembly file. After selecting the file, press the button Open and, from this file, data columns will be extracted, if they are present there.
9 The panel Columns consists of three fields. In these fields show a list of columns, their properties, and a description of these properties.

Press the Ok button once the fields are filled and parameters are specified. After that, in the data dictionary of the report a description of a new business object will be created, which can be used to create reports. The picture below shows a report built using a business object:



Number	Name	Description
1	Stimulsoft Reports	NET
2	Stimulsoft Reports	WPF

Provide the data to business objects from the data source in .NET

Created business objects that are registered and passed to the report generator, but do not contain the actual data are called a description of business objects. Using the description of the business object, you can create a report template (define the structure and design the report), and then, before building, connect the real data and render a report. This is useful if you want to create reports with the same structure and design, but with different data. Create a structural description of the business object first. Below is a sample code to create a business object class:

C#

```
...
public class MyObject
{
   public class Category
   {
      public int categoryID;
      public int CategoryID
      {
         get
         {
            return categoryID;
         }
      }

      public string categoryName;
```

© 2003-2024 Stimulsoft

```
    public string CategoryName
    {
      get
      {
        return categoryName;
      }
    }

    public string description;
    public string Description
    {
      get
      {
        return description;
      }
    }

  }

  public Category[] list = null;
  public Category[] Categories
  {
    get
    {
      return list;
    }
  }
}
...
```

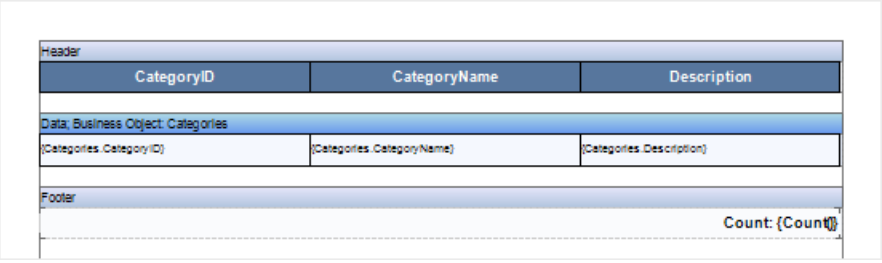
You then need to create a new business object class, register and pass it to the report generator. Below is a sample code to create and register a new business object:

C#

```
...
MyObject.Category obj = new MyObject.Category();
int busobjLevel = 1;

StiReport report = new StiReport();
report.RegBusinessObject("Categories", obj);
report.Dictionary.SynchronizeBusinessObjects(busobjLevel);
report.Design();
...
```

Now with help of the created description of the business object, create a report template in the designer. The picture below shows a report template created with the description of the business object:

© 2003-2024 Stimulsoft



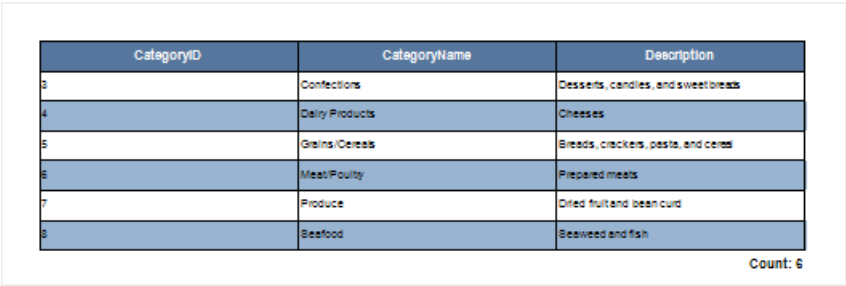
Once a report template is created, you can save it, for example, to the following path D:\\Report.mrt. Because the description of the business object does not contain the actual data, in order to render a report, you will get the real data to business objects, in our example we take the data from the database Northwind. For a start, create a connection to the database in Visual Studio. After that, put the code to obtain data for the business object. Getting real data for the business object occurs immediately before the report. Here is the code to obtain data for the business object:

C#

```
...
int busobjLevel = 1;

StiReport report = new StiReport();
report.Load("D:\\Report.mrt");

using (NorthwindDataContext context = new NorthwindDataContext())
{
   var categories =
      from c in context.Categories
      select new { c.CategoryID, c.CategoryName, c.Description };

   report.RegBusinessObject("Categories", categories);
   report.Show();
}
...
```

After that, the report generator will receive the data for the business object from the specified source, in this case from the database Northwind. Then, the report will be rendered by the existing template. The picture below shows the rendered report:

© 2003-2024 Stimulsoft



CategoryID	CategoryName	Description
3	Confections	Desserts, candies, and sweet breads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereal
6	Meat/Poultry	Prepared meats
7	Produce	Dried fruit and bean curd
8	Seafood	Seaweed and fish

Count: 6

Business objects in Web

Creating, filling, signing and sending business objects to the Web is almost the same as in .NET. First, create a class of the business object that is identical as in .NET. Next, create an object of the business object class, register it manually fill data and pass them. Here are the differences that, instead of the mainreport.Design() method, you should use the StiWebDesigner1.Design(mainreport) method. Also perform synchronization using the mainreport.Dictionary.SynchronizeBusinessObjects(), because in the Web designer it is not possible to create a description of the business object from the data dictionary (the description can only be created from code). Below is a sample code to create, fill, register and pass the business objects:

C#

```
...
MyObject obj = new MyObject();
obj.list = new MyObject.Category[2];

MyObject.Category c1 = new MyObject.Category();
c1.number = 1;
c1.name = "Cat1";
c1.description = "desc for n1";

MyObject.Category c2 = new MyObject.Category();
c2.number = 2;
c2.name = "Cat2";
c2.description = "desc for n2";

obj.list[0] = c1;
obj.list[1] = c2;

int busobjLevel = 1;

StiReport mainreport = new StiReport();
mainreport.RegBusinessObject("MyObject", obj);
mainreport.Dictionary.SynchronizeBusinessObjects(busobjLevel);
StiWebDesigner1.Design(mainreport);
...
```

© 2003-2024 Stimulsoft

Just as in .NET, in Web you can create a description of the business objects first, then the report template, and then connect the data source with the real data and render a report. Create a description of the business object. But previously you have to make the class of the business object that is identical to the class of business object in .NET. Here is an example of writing a business object:

C#

```
...
MyObject.Category obj = new MyObject.Category();
int busobjLevel = 1;

StiReport report = new StiReport();
report.RegBusinessObject("Categories", obj);
report.Dictionary.SynchronizeBusinessObjects(busobjLevel);
StiWebDesigner1.Design(report);
...
```

Now with the description created, design a report template identical to .NET. Once a report template is created, you can save it, for example to the following path D:\\Report.mrt. Since the description of the business object does not contain the actual data, in order to build a report, you should get the real data to business objects, in this example, we take the data from the database Northwind. First, create a connection to the database in Visual Studio. After that, write the code to obtain data for the business object. Getting real data for the business object occurs immediately before the report. Here is the code to obtain data for the business object:

C#

```
...
int busobjLevel = 1;

StiReport report = new StiReport();
report.Load("D:\\Report.mrt");

using (NorthwindDataContext context = new NorthwindDataContext())
{
   var categories =
      from c in context.Categories
      select new { c.CategoryID, c.CategoryName, c.Description };

   report.RegBusinessObject("Categories", categories);
   StiWebViewer1.Report = report;
}
...
```

© 2003-2024 Stimulsoft

15.1.2 Working with OData Using Business Objects

The protocol **Open Data (OData)** is used to access from different sources, including relational databases, file systems, content management systems and ordinary web sites. **OData** realizes the **CRUD** conception (Create, Read, Update, Delete) in relation to data. In **Visual Studio 2010** and **.NET Framework 4.0** was simplified with support of **OData** using the access technology **Entity Framework**. On the basis of received (using OData protocol) data, it is possible for a user to create reports. Passing data to the report goes through business objects. Let's have an example of retrieving data from the report and passing data to the report:

- Connect the **Stimulsoft** assemblies;
- Add **Service Reference** specifying the address of the entry point to OData-Service. In this case the address is http://services.odata.org/V3/OData/OData.svc;
- Use following code.

C#

```
...
//Connecting to Data Storage
Uri uri = new Uri("http://services.odata.org/V3/OData/OData.svc");
var container = new ServiceReference1.DemoService(uri);

//Creating Query with Selection Parameters
var product = container.Products.Where(p => p.ID < 50).ToList();

//Transffering Data to Report via Business Objects
var report = var StiReport();
report.RegBusinessObject("Products", product);
report.Dictionary.SynchronizeBusinessObjects(2);
report.Design();
...
```

15.2 Report Inheritance

There are two ways of report inheritance:

- Creation of the basic class of a report;
- Creation of the master-report.

In both ways you should create a basic report in the designer that includes all necessary elements. You may add the following components to the basic reports:

- Pages;
- Components;
- Data sources;
- Variables;
- Connections.

© 2003-2024 Stimulsoft

To learn more of Basic Approaches.

15.2.1 Basic Approaches

After the report has been created you may either save the report as a special basic class (for this you should use the Save as command) or save the report as a regular report and then use it as a master report. In the first case, you will get the C# or VB.NET class, and will be able to create new reports. For example:

C#

```
...
Reports.Report master = new Reports.Report();
master.RegData(dataSet);
master.Design();
...
```

In order to use the basic report when creating a new report in the designer, you need to add the following string of a code:

C#

```
...
StiReport.ReportType = typeof(Reports.Report);
...
```

Then all new reports will be automatically inherited from the basic class. In the second way you need to use the following code:

C#

```
...
StiReport masterReport = new StiReport();
masterReport.Load("d:\\master-detail.mrt");

StiReport report = new StiReport();
report.RegData(dataSet);

report.MasterReport = masterReport.SaveToString();
report.Design();
...
```

15.3 Right To Left

By default, components are output from left to right. The **Right to Left** property allows changing the mode of showing report items.

© 2003-2024 Stimulsoft

- WinForms Viewer,
- WPF Viewer,
- Icons.

15.3.1 WinForms Viewer

There is a capability to change the mode of showing viewer items and order of showing report pages in WinForms. By default, showing of all elements of the viewer and the display order of pages of the report is left to right. How the viewer will look like can depend on the static **RightToLeft** property of the viewer. If the **RightToLeft** property is set to No, then the viewer items and pages of the report in the viewer window are shown from left to right. Code below show how to set the left to right mode of showing viewer items and report pages:

C#

```
...
StiOptions.Viewer.RightToLeft = StiRightToLeftType.No;
...
```

If the **RightToLeft** property is set to **Yes**, then viewer items and report pages in a viewer window are displayed from right to left. The code for setting the right to left mode is shown below:

C#

```
...
StiOptions.Viewer.RightToLeft = StiRightToLeftType.No;
...
```

15.3.2 WPF Designer and Viewer

In the designer and WPF report viewer it is possible to set the mode of showing controls in the right to left direction. By default, all controls are displayed in the left to right order. It is possible to change this. The **FlowDirection** property is used for this. If the property is set to **LeftToRight**, then controls are shown from left to right (see the code below).

XAML

```
...
FlowDirection="LeftToRight"
...
```

© 2003-2024 Stimulsoft

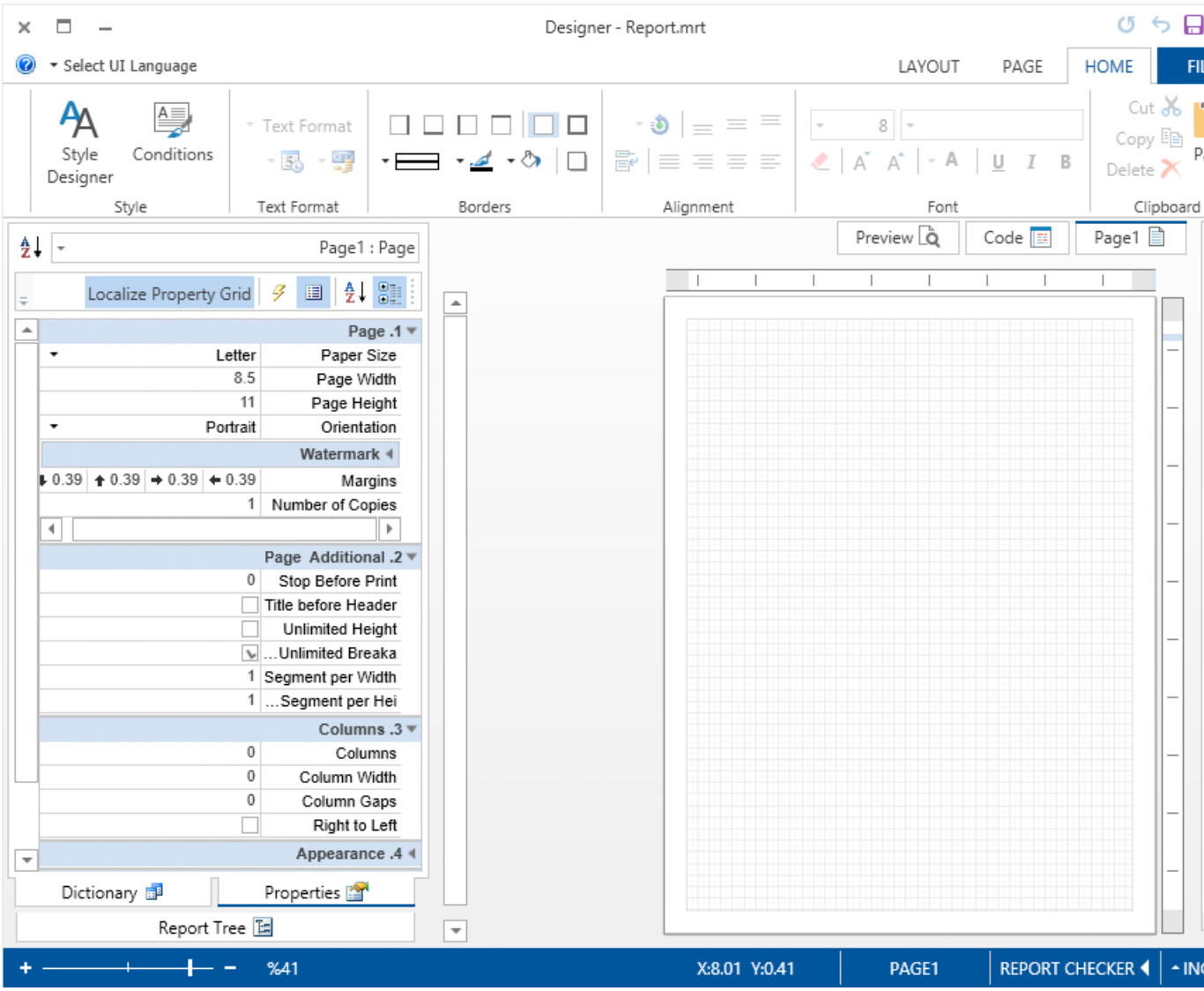
The picture below shows the left to right order of showing controls in the designer and viewer. If the **FlowDirection** property is set to **RightToLeft**, then controls are shown in the right to left order. See the code below how to achieve this result.

XAML

```
...
FlowDirection="RightToLeft"
...
```

The picture below shows the right to left order of showing controls in the designer and viewer.

© 2003-2024 Stimulsoft



As can be seen from the picture above, the order of showing report pages in the viewer also depends on the value of the **FlowDirection** property.

15.3.3 Icons

With the help of icons information about the components, settings and tools applied to these components in the reports is displayed. These icons are filtering, conditions and events of the inherited report, dynamic collapsing, dynamic sorting, order, quick information. By default, these icons are displayed in the left to right order in a component, but if necessary, they can be displayed in the right to left order. It is possible using the properties:

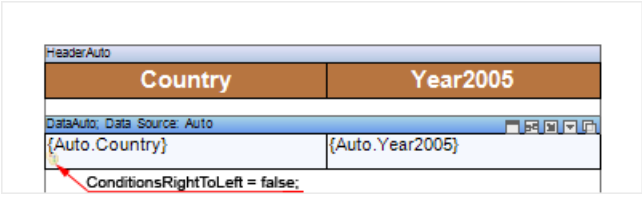
- ConditionsRightToLeft,
- EventsRightToLeft,

© 2003-2024 Stimulsoft

- InheritedRightToLeft,
- InteractionCollapsingRightToLeft,
- InteractionSortRightToLeft,
- OrderAndQuickInfoRightToLeft,
- FiltersRightToLeft,
- QuickButtonsRightToLeft.

They belong to the **StiOptions.Viewer.Pins** class. Consider these features in more detail:

› The mode of displaying the Conditions icon depends on the value of the **ConditionsRightToLeft** property. For example, if to place a Condition in the DataBand, then the Conditions icon will be displayed by default in the lower left corner of this DataBand, because the **ConditionsRightToLeft** property is set to **false**. The picture below shows an example of a report template with the Conditions icon in the left to right mode:

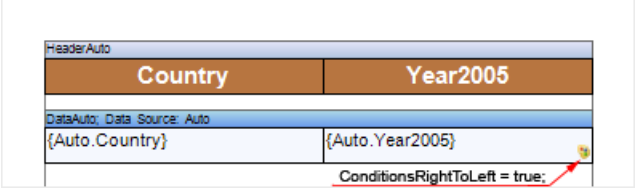


Set the **ConditionsRightToLeft** property to true to change the position of the icon:

C#

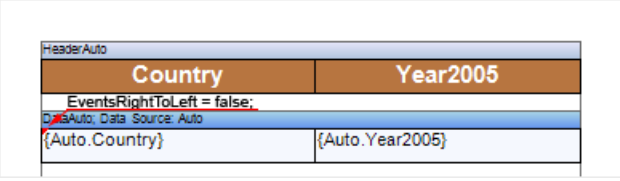
```
...
StiOptions.Viewer.Pins.ConditionsRightToLeft = true;
...
```

And then the icon will be displayed in the "**right to left**" mode, i.e. in the lower right corner of the DataBand. The picture below shows an example of a report template with the Conditions icon in the right to left mode:



© 2003-2024 Stimulsoft

› Change the value of the **EventsRightToLeft** property to change the location of the Events icon. For example, if to place an Event in the text component, then the Events bookmark will be displayed by default in the upper left corner of the text component, because the **EventsRightToLeft** property is set to **false**. The picture below shows an example of a report template with the Events icon in the left to right mode:

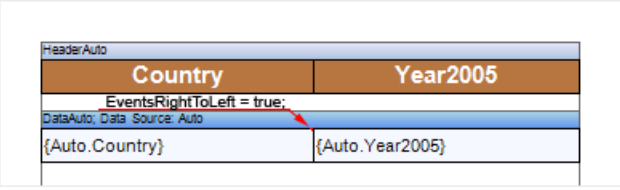


Set the **EventsRightToLeft** property to true to change the location of the **Events** icon.

C#

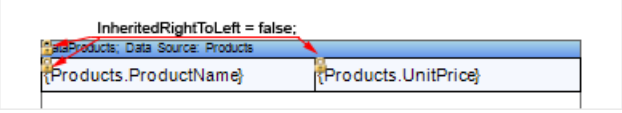
```
...
StiOptions.Viewer.Pins.EventsRightToLeft = true;
...
```

The picture below shows an example of a report template with the **Events** icon in the right to left mode:



› The **InheritedRightToLeft** property is used to change the location of the Inherited icon. By default, this property is set to false, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the Inherited icon in the left to right mode:

© 2003-2024 Stimulsoft

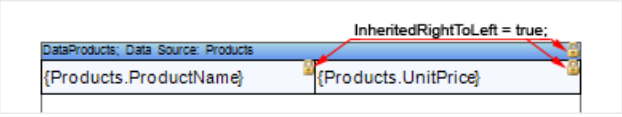


If the **InheritedRightToLeft** property is set to **true**

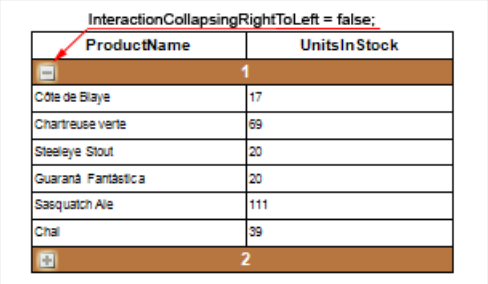
C#

```
...
StiOptions.Viewer.Pins.InheritedRightToLeft = true;
...
```

the Inherited icon will appear in the right to left mode (see the picture below):



› The **InteractionCollapsingRightToLeft** property is used to change the location of the Collapsing icon. By default, this property is set to false, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the Collapsing icon in the left to right mode:



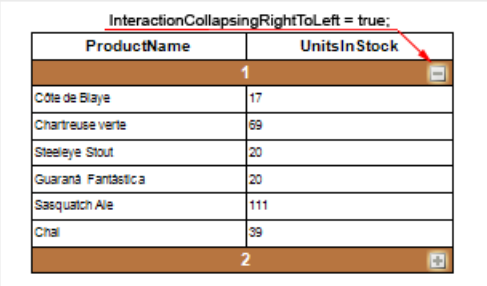
If the **InteractionCollapsingRightToLeft** property is set to **true**

C#

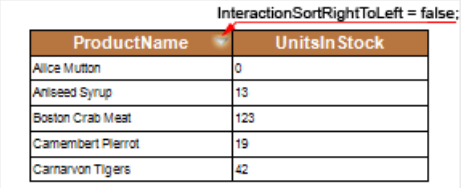
```
...
StiOptions.Viewer.Pins.InteractionCollapsingRightToLeft = true;
...
```

© 2003-2024 Stimulsoft

Collapsing icons will appear in the right to left mode (see the picture below):



› The **InteractionSortRightToLeft** property is used to change the location of the **InteractionSort** icon. By default, this property is set to false, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the **InteractionSort** icon in the left to right mode:

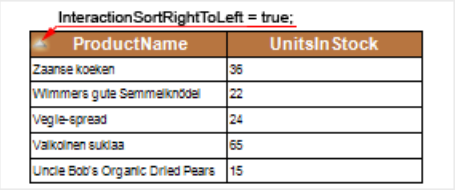


If the **InteractionSortRightToLeft** property is set to **true**

C#

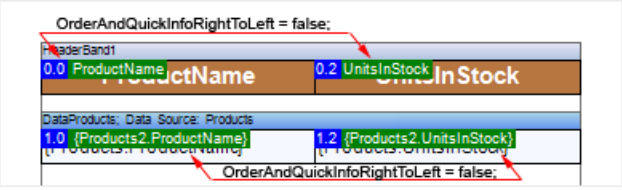
```
...
StiOptions.Viewer.Pins.InteractionSortRightToLeft = true;
...
```

the **InteractionSort** icon will appear in the right to left mode (see the picture below):



© 2003-2024 Stimulsoft

› The **OrderAndQuickInfoRightToLeft** property is used to change the location of the **Show Order** icon. By default, this property is set to false, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the Show Order icon in the left to right mode:

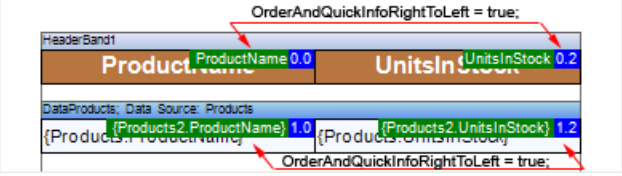


If the **OrderAndQuickInfoRightToLeft** property is set to **true**:

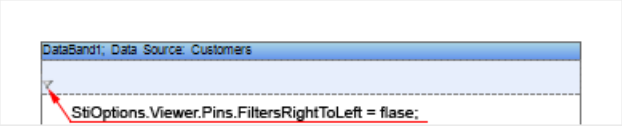
C#

```
...
StiOptions.Viewer.Pins.InteractionSortRightToLeft = true;
...
```

the **Show Order** icon will appear in the right to left mode (see the picture below):



› The **FiltersRightToLeft** property is used to change the location of the Filters icon. By default, this property is set to false, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the Filters icon in the left to right mode:



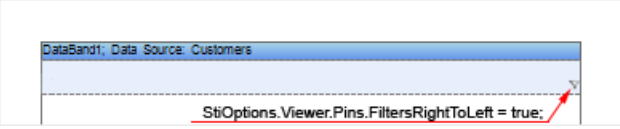
If the **FiltersRightToLeft** property is set to **true**:

© 2003-2024 Stimulsoft

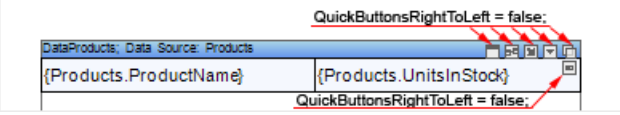
C#

```
...
StiOptions.Viewer.Pins.FiltersRightToLeft = true;
...
```

the **Filters** icon will appear in the right to left mode (see the picture below):



› The **QuickButtonsRightToLeft** property is used to change the location of the **QuickButtons** icon. By default, this property is set to false, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the Filters icon in the left to right mode:

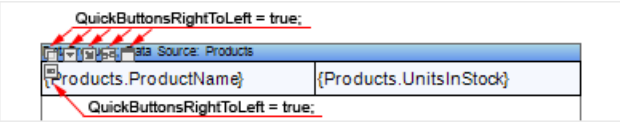


If the **QuickButtonsRightToLeft** property is set to **true**:

C#

```
...
StiOptions.Viewer.Pins.QuickButtonsRightToLeft = true;
...
```

the **QuickButtons** icon will appear in the right to left mode (see the picture below):



15.4 Exports

YouTube

© 2003-2024 Stimulsoft

Watch the video tutorials how to export report. Subscribe to the Stimulsoft channel and be the first to watch new video lessons. Questions and suggestions is recommended be left in the comments to the video.

This section describes principles of saving rendered reports to different formats, basic characteristics of methods for export, export optimization guidelines data structure which are used in export methods. Stimulsoft Reports supports great many export formats to save rendered reports. Many clients think that there are too many formats. But when you need to get file of definite format type, write only one string of code and the format is not PDF, HTML or RTF, only Stimulsoft Reports may help. We do not think that too many export formats in our report generator is disadvantage and continually work on adding new formats. The more exports the better, as they say.

- Exports Reports From Code
- Formats with Fixed Page layout
- Web Documents
- Text Formats
- Spreadsheets
- Data
- Images

Available File Formats

The **StiExportFormat** enumeration describes export formats. Brief information of exports is represented below.

Formats	Description
Formats which are used for representing documents and allows for easy viewing and printing	
PDF	export to Adobe PDF.
Microsoft Power Point 2007/2010	export to Microsoft Power Point 2007/2010

© 2003-2024 Stimulsoft

XPS	export to Microsoft XPS.
	Web formats
HTML	export to HTML by default. This element duplicates the HTMLTable mode.
HTMLTable	export to HTML using the HTML Table element, to create a report structure.
HTMLSpan	export to HTML using the HTML Span element, to create a report structure.
HTMLDiv	export to HTML using the HTML Div element, to create a report structure.
MHT	export to WebArchive. This format is supported only in Microsoft IE.
	Text formats
Text	export to Text.
RTF	export to Rich Text Format by default. This element duplicates the RTFTable mode.
RTFTable	export to Rich Text Format using the RTF Table element, to create a report structure.
RTFFrame	export to Rich Text Format using the RTF Frame element, to create a report structure.
RTFWinWord	export to Rich Text Format using the Microsoft Word graphic element, to create a report structure.
RTFTabbedText	export to Rich Text Format using the symbols of tabulation, to create a report structure.
Word 2007/2010	export to Microsoft Word 2007/2010. This format is supported starting with Microsoft Office 2007/2010.
ODT	export to the OpenDocument Writer file.
	Spreadsheets
Excel	export to Microsoft Excel. The file is created using the BIFF (Binary Interchange File Format).

© 2003-2024 Stimulsoft

ExcelXML	export to Microsoft Excel XML. The file is created using the XML. This format is supported starting with Microsoft Office 2003.
Excel 2007/2010	export to Microsoft Excel 2007/2010. This format is supported starting with Microsoft Office 2007/2010.
ODS	export to OpenDocument Calc file.
Export as Data	
CSV	export to CSV (Comma Separated Value).
DBF	export to dBase/FoxPro.
XML	export to XML as data. This format is a saved DataSet.
DIF	export to DIF (Data Interchange Format).
SYLK	export to SYLK (Symbolic Link).
Export as Image	
ImageGif	export to GIF.
ImageBmp	export to BMP.
ImagePcx	export to PCX.
ImagePng	export to PNG.
ImageTiff	export to TIFF.
ImageJpeg	export to JPEG.
ImageEmf	export to Windows Metafile.

15.4.1 Export Reports From Code

Stimulsoft Reports offers many ways of exporting rendered reports to other formats. Each method of export to other format has several settings. For exporting rendered reports Stimulsoft Reports uses a system of services. This means that all objects which are used in export are represented in the collection of services and when it is necessary to export a report, the report generator searches the appropriate service in the collection of services. There are two ways of exporting rendered formats to other formats from code: using the ExportDocument method of the StiReport class, and using direct creating or getting from a collection of services the required export service.

© 2003-2024 Stimulsoft

- ExportDocument Method,
- Export Service.

15.4.1.1 ExportDocument Method

The **ExportDocument** method is a simplified wrapping for report exports. There is no need to get the required export service. All you need is to define the export type, pass parameters of export and define the folder where the file should be saved. For example:

C#

```
...
StiPdfExportSettings pdfSettings = new StiPdfExportSettings();
report.ExportDocument(StiExportFormat.Pdf, "MyReport.Pdf", pdfSettings);
...
```

The following code id used to export reports to PDF. The PDF file will be placed in the MyReport.Pdf. The export parameters can be passed using the **StiPdfExportSettings** object type. This class is described in the description of the PDF format. If there is no need to change export parameters then it is possible to use the short code line:

C#

```
...
report.ExportDocument(StiExportFormat.Pdf, "MyReport.Pdf");
...
```

In this case the export parameters are not passed and the report generator will use parameters which are set by default for each export. Besides, the result of export can be placed in the stream. For example:

C#

```
...
MemoryStream stream = new MemoryStream();
report.ExportDocument(StiExportFormat.Pdf, stream);
...
```

Information

© 2003-2024 Stimulsoft

The **ExportDocument** method does not call the Render method automatically. Before calling the **ExportDocument** method it is necessary to render a report or load a previously rendered report.

As you can see, no services in examples were not created and samples contain simple code. All work by creating services and checking parameters can be done using the **ExportDocument** method.

The code above requires connection the following namespaces from assemblies **Stimulsoft.Reports.dll**:

C#

```
...
Stimulsoft.Report
...
```

15.4.1.2 Export Service

The way to create the export service is shown below. See the code:

C#

```
...
StiPdfExportService service = new StiPdfExportService();
StiPdfExportSettings settings = new StiPdfExportSettings();
MemoryStream stream = new MemoryStream();
service.ExportPdf(report, stream, settings);
...
```

If you exported from the WinForms Viewer, then you should notice, than for each export the special form for setting parameters of export is shown. This form can be called from the code. The code below how to do it for the export to the **PDF**:

C#

```
...
service.Export(report, "MyReport.pdf");
...
```

© 2003-2024 Stimulsoft

This code will call the dialog form for setting parameters of export. If a user clicks "OK", then the file will be created. If to click the "Cancel" button, then the file creation will be interrupted.

Information

The name of the method for the report export with dialog forms differs from the name of the export method without parameters.

The export service of a report contains yet another ability. The report can be sent via Email. For example:

C#

```
...
bool sendEMail = true;
service.Export(report, "MyReport.pdf", sendEMail);
...
```

This code will call the dialog form for setting parameters of reports, and if a user clicks "OK", then the reporting tool will call the Email client and will create a new Email Email, the exported report will be attached to the Email Email. The code above requires connection of the following names from the **Stimulsoft.Report.dll** assemblies:

C#

```
...
Stimulsoft.Report
Stimulsoft.Report.Export
...
```

All Export Services

The **StiExportFormat** enumeration describes export formats. Brief information of exports is represented below.

© 2003-2024 Stimulsoft

Export services to Adobe PDF and Microsoft XPS:

- StiPdfExportService,
- StiXpsExportService

Export services to HTML and MHT:

- StiHtmlExportService
- StiMhtExportService

Export services to text formats:

- StiTxtExportService
- StiRtfExportService
- StiWord2007ExportService
- StiOdtExportService

Export services to Microsoft Excel and Open Document Calc:

- StiExcelXmlExportService
- StiExcelExportService
- StiExcel2007ExportService
- StiOdsExportService

Export services to graphic formats:

- StiBmpExportService
- StiGifExportService
- StiJpegExportService
- StiPcxExportService
- StiPngExportService
- StiTiffExportService
- StiEmfExportService

Export services to data:

- StiCsvExportService
- StiDbfExportService
- StiXmlExportService
- StiDifExportService
- StiSylkExportService

15.4.2 Formats with Fixed Page Layout

Stimulsoft Reports supports two exports with fixed page layout. What is the fixed page layout? This means that all elements of a page can be placed at any part of a page. In this case, if to change a position of one element then other components position will not be changed.

- PDF (Portable Document Format),
- Microsoft Power Point 2007/2010,
- XPS (XML Paper Specification).

15.4.2.1 PDF

PDF (Portable Document Format) – is a file format created by Adobe Systems for document exchange used to create electronic editions using the Adobe Acrobat package. The PDF format is a file text format that is used to publish documents on any platform and OS. The PDF document contains one or more pages. Each page

© 2003-2024 Stimulsoft

may contain any components: text, graphic and illustrations, information, that provides navigation across the document.

Export to PDF is based on the "Adobe Portable Document Format, Version 1.3, second edition", using some elements of latest format specifications.

Information

Now, when exporting to PDF, the fields for which **Printable=false** are exported to a separate layer that is not printed in Adobe Acrobat. There is one limitation - layers are not supported in the PDF-A compatibility mode, so the **Printable** property is ignored.

If you print a report to PDF (**Print to PDF**) from a web viewer, then these fields are not exported.

Digital Signature

Digital signature is a requisite of an electronic document used to protect this document from falsification. This document is a result of cryptographic conversion of information using the closed key of the electronic signature and allows identifying the owner of the certificate of the key of the signature. Digital signatures are often used to implement electronic signatures

The **StiPdfExportSettings** class is used to control digital signature. It has the following properties:

C#

```
...
public bool UseDigitalSignature
public bool UseLocalMachineCertificates
public bool GetCertificateFromCryptoUI
public string SubjectNameString
...
```

By default:

© 2003-2024 Stimulsoft

C#

```
...
UseDigitalSignature = false;
UseLocalMachineCertificates = true;
GetCertificateFromCryptoUI = true;
SubjectNameString = string.Empty;
...
```

A sample how to use these properties is shown below:

C#

```
...
StiReport report = new StiReport();
report.Load("c:\\test.mrt");
report.Render(false);

StiPdfExportSettings settings = new StiPdfExportSettings();
settings.UseDigitalSignature = true;
settings.GetCertificateFromCryptoUI = false;
settings.UseLocalMachineCertificates = true;
settings.SubjectNameString = "John Smith <johns@google.com>";

report.ExportDocument(StiExportFormat.Pdf, "c:\\test.pdf", settings);
...
```

Encryption

A PDF document can be encoded to protect the content from unauthorized access. A user may set the following parameters of encryption:

- User password;
- Owner password;
- Access permission;
- Key length.

Using the StiPdfExportSettings class it is possible to set the encryption parameters from code. The following properties of this class are used:

C#

```
...
public string PasswordInputUser
public string PasswordInputOwner
public StiUserAccessPrivileges UserAccessPrivileges
public StiPdfEncryptionKeyLength KeyLength
```

© 2003-2024 Stimulsoft

```
...
```

The **StiUserAccessPrivileges** enumeration contains the following elements (flags):

- None,
- PrintDocument,
- ModifyContents,
- CopyTextAndGraphics,
- AddOrModifyTextAnnotations,
- All.

The StiPdfEncryptionKeyLength enumeration contains the following elements:

- Bit40,
- Bit128,
- Bit256.

By default the values set as follow:

C#

```
...
PasswordInputUser = string.Empty;
PasswordInputOwner = string.Empty;
UserAccessPrivileges = StiUserAccessPrivileges.All;
KeyLength = StiPdfEncryptionKeyLength.Bit40;
...
```

An example of using:

C#

```
...
StiReport report = new StiReport();
report.Load("c:\\test.mrt");
report.Render(false);

StiPdfExportSettings settings = new StiPdfExportSettings();
settings.PasswordInputUser = "user";
settings.PasswordInputOwner = "owner";
settings.UserAccessPrivileges = StiUserAccessPrivileges.PrintDocument;
settings.KeyLength = StiPdfEncryptionKeyLength.Bit128;

report.ExportDocument(StiExportFormat.Pdf, "c:\\test.pdf", settings);
...
```

© 2003-2024 Stimulsoft

Embedded Fonts

By default all embedded fonts are optimized. Characters which are not used in a report are excluded. It allows decreasing the size of a file. But, for correct work of the editable field, the font should be complete. Therefore, for fonts, which are used in editable fields, optimization is not done. This increases the output file size. If Asian languages are used, the file size can be 15-20mb.

If by some reasons the font optimization is not working correct it can be forcibly disabled using the static property:

C#

```
...
StiOptions.Export.Pdf.ReduceFontFileSize = false;
...
```

Editable Fields

To enable the export of editable fields it is necessary to set the static property

C#

```
...
StiOptions.Export.Pdf.AllowEditablePdf = true;
...
```

Editable fields in the PDF-file has two conditions:

- **First** – a condition before editing, it is shown when opening the file. This condition corresponds to the type of a text box in the preview.
- **Second** - the type in the mode of field editing, and after editing. In this condition it is impossible to set the vertical alignment of the text (always Top) and some parameters of a font. Therefore, after editing a field, even if the contents is not changed, the type of this field can be change.

If it is necessary to have the **MultiLine** editable field, then it is necessary to set the **WordWrap** property of the text box to **true**.

Export Settings

© 2003-2024 Stimulsoft

The export parameters of the PDF export are described in the **StiPdfExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution dpi; can take any value, by default 100
EmbeddedFonts	bool	embed font files into the PDF file; is true then all fonts are embedded into the file and this file will have the same look on any computer (there is no need to embed additional fonts); if false then fonts are not embedded; by default true
StandardPdfFonts	bool	use standard fonts which are embedded in Adobe Acrobat Reader and there is no need to embed them into the file; all fonts are changed on standard fonts (Courier, Helvetica, Times-Roman); by default false
Compressed	bool	compress the PDF file; decreases the file size by compressing the text information (images are always compressed); by default true
UseUnicode	bool	writes a text in the

© 2003-2024 Stimulsoft

		Unicode; if false then 190 symbols can be written, and a lot of problems with native language symbols may occur; if true then any symbols can be used; by default true
ExportRtfTextAsImage	bool	export RichText objects as images; if false then export tries to convert RichText objects into PDF primitives; if true the RichText is written as an image; by default false
PasswordInputUser	string	user password (see Encryption); by default empty string
PasswordInputOwner	string	owner password (see Encryption); by default empty string
UserAccessPrivileges	enum	user access privileges (see Encryption); by default StiUserAccessPrivileges.All
KeyLength	enum	key length (see Encryption); by default StiPdfEncryptionKeyLength .Bit40
UseDigitalSignature	bool	use digital signature of a document; by default false
GetCertificateFromCryptoUI	bool	get the certificate from the Crypto interface; if false then a certificate is searched by certificate identifier without using the interface; by default true

© 2003-2024 Stimulsoft

SubjectNameString	string	certificate identifier; this is a certificate name (empty string) or a part of a name (substring); by default empty string
UseLocalMachineCertificates	bool	search certificates on the local machine; if false then certificate is searched in the store of the current user; by default false
CreatorString	string	the "Creator" field in the document description (application name that created the original file); if it is not set (empty string) then the StiOptions.Export.Pdf static property is used. CreatorString; by default empty string
KeywordsString	string	the "Keywords" field in the document description (application name that created the original file); if it is not set (empty string) then the StiOptions.Export.Pdf.KeywordsString static property is used; by default empty string
ImageCompressionMethod	enum	image compression method - JPEG (with quality loss) or Flate (without quality loss); by default StiPdfImageCompressionMethod.Jpeg

© 2003-2024 Stimulsoft

DitheringType	enum	using this property as a format of monochrome image (with dithering and without) is defined

If the **UseUnicode** is used then, for Acrobat Reader 5.0, it is necessary to use the **Embedded fonts = true**. If the **UseUnicode** + encoding is used, then it is necessary to use the **Embedded fonts = true**.

The following should be done to compress file:

- enable **Compressed**;
- disable **Embedded fonts**;
- if Embedded fonts is required then enable the **ReduceFontFileSize**.

Static Options

Except the **StiPdfExportSettings** class, parameters of export to PDF are also set using the static properties. All properties are described in the table below. To access to export properties it is necessary to add the **StiOptions.Export.Pdf...** prefix. For example, **StiOptions.Export.Pdf.DivideSegmentPages**.

Name	Type	Description
DivideSegmentPages	bool	divide segmented pages into separate pages; if false then are exported "as is" without dividing; by default true
ConvertDigitsToArabic	bool	convert ASCII digits Arabic; by default false
ArabicDigitsType	enum	Select Arabic digits type; by default Standard
ReduceFontFileSize	bool	optimize embedded fonts - eliminate symbols which are not met in a report; if false then fonts are not

© 2003-2024 Stimulsoft

Name	Type	Description
		changed; by default true
AllowEditablePdf	bool	export editable fonts as editable PDF objects (in this case fonts which are used in editable fields are not optimized); if false then editable fields are exported as simple text; by default false
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true
AllowImageTransparency	bool	use transparency in export images; by default true
AllowInheritedPageResources	bool	store resources of pages in the parent dictionary and inherit from it; if false then resources of pages are specified in each page; this property is critical for some programs of PDF files processing; by default true
AllowExtGState	bool	use command to control transparency when creating a document; if false then commands are not used; this property is critical for some programs of PDF files processing; by default true
CreatorString	string	the "Creator" field in document description

© 2003-2024 Stimulsoft

Name	Type	Description
		(application name, which created the original document); by default the "Stimulsoft Reports.NET" string
KeywordsString	string	the "Keywords" field in document description (keywords); by default empty string

15.4.2.2 ZUGFeRD

Stimulsoft Company has added support for the format of electronic invoices - ZUGFeRD.

Invoices in the **ZUGFeRD** format pass both human-readable invoices and its structured machine-readable XML based representation. Human-readable representation is encoded in the form of one or more PDF pages of the PDF/A format. XML based representation is embedded in the PDF document as an object in accordance with the specifications of the PDF/A-3 format. In other words, the invoice of the **ZUGFeRD** format contains two separate representations - human-readable in the PDF/A-3 format that is used as a container for the XML representation.

At this moment, you can use the **ZUGFeRD** format only from code. To do this, select the desired format option (V1 or V2) using the **ZUGFeRDComplianceMode** option in the export settings to PDF, using the **ZUGFeRDConformanceLevel** option, select the desired **Conformance Level**, and also load the pre-prepared XML file into the **ZUGFeRDInvoiceData** property. This will automatically add the file to the **EmbeddedFiles** collection with the standard **FileName** and **Description**. If you need to use another **Description**, you yourself can add the file to the **EmbeddedFiles** collection with the desired **FileName** and **Description**.

Pay attention:
- The name of the XML file in different versions of the standard is case-sensitive.
- ConformanceLevel COMFORT in ZUGFeRD 2.0 replaced by EN 16931.

The following is a sample code for exporting a report using the **ZUGFeRD** format:

© 2003-2024 Stimulsoft

C#

```
...
FileStream fileStream = new FileStream(@"d:\test.pdf", FileMode.Create);
byte[] buf = File.ReadAllBytes(@"d:\ZUGFeRD-invoice.xml");

//for ZUGFeRD 1.0
var pdfExportSettings = new StiPdfExportSettings()
{
   ZUGFeRDComplianceMode = StiPdfZUGFeRDComplianceMode.V1,
   ZUGFeRDInvoiceData = buf,
   ZUGFeRDConformanceLevel = "COMFORT"     //BASIC, COMFORT, EXTENDED
};

//for ZUGFeRD 2.0
var pdfExportSettings = new StiPdfExportSettings()
{
   ZUGFeRDComplianceMode = StiPdfZUGFeRDComplianceMode.V2,
   ZUGFeRDInvoiceData = buf,
   ZUGFeRDConformanceLevel = "EN 16931"    //BASIC, EN 16931, EXTENDED
};

report.ExportDocument(StiExportFormat.Pdf, fileStream, pdfExportSettings);
fileStream.Close();
...
```

Below is an example of code for exporting a report using the **ZUGFeRD** format, if you need to use an alternative **Description** for the XML file you should to the following:

C#

```
...
FileStream fileStream = new FileStream(@"d:\test.pdf", FileMode.Create);
byte[] buf = File.ReadAllBytes(@"d:\ZUGFeRD-invoice.xml");

//for ZUGFeRD 1.0, Custom settings
var pdfExportSettings = new StiPdfExportSettings();
pdfExportSettings.ZUGFeRDComplianceMode = StiPdfZUGFeRDComplianceMode.V1;
pdfExportSettings.EmbeddedFiles.Add(new StiPdfEmbeddedFileData("ZUGFeRD-
invoice.xml", "ZUGFeRD Invoice", buf));
pdfExportSettings.ZUGFeRDConformanceLevel = "COMFORT";

//for ZUGFeRD 2.0, Custom settings
var pdfExportSettings = new StiPdfExportSettings();
pdfExportSettings.ZUGFeRDComplianceMode = StiPdfZUGFeRDComplianceMode.V2;
pdfExportSettings.EmbeddedFiles.Add(new StiPdfEmbeddedFileData("zugferd-
invoice.xml", "ZUGFeRD Invoice", buf));
pdfExportSettings.ZUGFeRDConformanceLevel = "EN 16931";

report.ExportDocument(StiExportFormat.Pdf, fileStream, pdfExportSettings);
fileStream.Close();
...
```

© 2003-2024 Stimulsoft

15.4.2.3 Special Features of PDF/A

PDF/A is an ISO-standardized version of the PDF specialized for use in the archiving and long-term preservation of electronic documents. The PDF/A standard does not define an archiving strategy or the goals of an archiving system. It identifies a "profile" for electronic documents that ensures the documents can be reproduced exactly the same way using various software in years to come. A key element to this reproducibility is the requirement for PDF/A documents to be 100% self-contained. All of the information necessary for displaying the document in the same manner is embedded in the file. This includes, but is not limited to, all content (text, raster images and vector graphics), fonts, and color information. A PDF/A document is not permitted to be reliant on information from external sources.

When the "PDF/A Compliance" mode is enabled, the exported report is subjected to the following restrictions:

- no transparency is allowed (PDF/A-1 only);
- no hyperlinks can be used;
- tooltips are not permitted;
- fonts must always be included;
- document **encryption** is not utilized.

When exporting a report to a PDF document using the PDF/A standard, certain errors may occur. In this chapter, we will discuss some of them.

The "Font not embedded" error

The PDF-A standard mandates the inclusion of fonts within the PDF file. Consequently, when the "PDF/A Compliance" mode is enabled, the fonts will be automatically included in the PDF export, regardless of the "Embedded fonts" parameter in the export settings. However, in applications utilizing .NET Core and JS components, the report engine can only access fonts that have been loaded into the `StiFontCollection`. Therefore, if a particular font has not been loaded, it will not be incorporated into the resulting PDF file. Consequently, the PDF/A compliance check will display an error indicating that the font has not been embedded.

HomeController.cs

```
StiFontCollection.AddFontFile(StiNetCoreHelper.MapPath(this, "Reports/
Font.ttf"));
```

© 2003-2024 Stimulsoft

The "Width information for rendered glyphs is inconsistent" error
("Glyph widths in the font dictionary are not consistent with embedded font program widths")
The error message "Width information for rendered glyphs is inconsistent" or "Glyph widths in the font dictionary are not consistent with embedded font program widths" typically occurs when dealing with TrueType or OpenType fonts. These fonts store the contours of symbols as points connected by straight lines or arcs, making them vector fonts. The font file itself contains information about the font, such as:

- Name;
- Coding;
- And a "Widths" table that specifies the width of all the symbols used. This table serves as a cache for viewers, enabling them to quickly process text without calculating the width of characters from the font data each time.

The PDF/A compliance verification process involves multiple stages, one of which is checking the consistency between the data in the Widths table and the actual font data. During this step, the verification utility calculates the width of characters and compares it with the values recorded in the Widths table. According to the standards, a deviation of up to 0.1% is permissible. However, a problem arises due to different graphic libraries calculating character widths differently. As a result, the calculated values may differ, sometimes by 1-2%, which exceeds the allowed deviation of 0.1%. Additionally, different verification utilities may use varying methods to calculate character widths, leading to discrepancies and errors flagged in different files.

Notice

In one instance from our experience, we encountered a scenario where a PDF file underwent verification using Adobe Acrobat Pro successfully, but failed when tested using an online utility. Our graphics library determined the width of a specific character to be 554, a value that was acceptable to Adobe Acrobat. However, upon manually adjusting the number to 553, the file successfully passed the online utility's verification but began to encounter issues with Adobe Acrobat Pro.

While there is no definitive solution to this problem, it is crucial to consider this

© 2003-2024 Stimulsoft

particular peculiarity when encountering such an error.

15.4.2.4 Microsoft Power Point 2007/2010

Microsoft PowerPoint is a presentation program developed by Microsoft. It is a part of the Microsoft Office suite. PowerPoint presentations consist of a number of individual pages or "slides". Slides may contain text, graphics, movies, and other objects, which may be arranged on the slide. The presentation can be printed, displayed on a PC, or navigated through at the command of the presenter. In Stimulsoft Reports each report page corresponds to one slide.

Export Settings

The export parameters of the PPT export are described in the **StiPpt2007ExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution dpi; can take any value, by default 100

Static Options

Besides the **StiPpt2007ExportSettings** class, the parameters of the export to PPT are also set using the static properties. All properties are described in the table below. To access to export properties it is necessary to add the **StiOptions.Export.Ppt2007...** prefix. For example, **StiOptions.Export.Ppt2007.ReduceFontFileSize**.

Name	Type	Description
AllowImageComparer	bool	use the image comparer, e.g. replace image

© 2003-2024 Stimulsoft

Name	Type	Description
		duplicates (see Common export settings); if false then an image is exported "as is"; by default true

15.4.2.5 XPS

XPS (XML Paper Specification) is the open graphic format of fixed page layout on the base XML (more precisely XAML-based) used to store printed output as electronic documents. This format was developed by Microsoft as alternative to the PDF format.

The XPS document format consists of structured XML markup that defines the layout of a document and the visual appearance of each page, along with rendering rules for distributing, archiving, rendering, processing and printing the documents. The markup language for XPS is a subset of XAML that allows including vector graphic elements, using XAML to mark up the WPF-primitives.

The XPS is a ZIP-archive that contains the files which make up the document. The archive includes page mark up (one file per each page of a document), text, embedded fonts, raster images, 2D vector graphics and other information.

Export Settings

The export parameters of the XPS export are described in the StiXpsExportSettings class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution dpi; can take any value, by default 100

© 2003-2024 Stimulsoft

Static Options

Besides the **StiXpsExportSettings** class, the parameters of export to XPS are also set using the static properties. All properties are described in the table below. To access to export properties it is necessary to add the **StiOptions.Export.Xps**... prefix. For example, **StiOptions.Export.Xps.ReduceFontFileSize**.

Name	Type	Description
ReduceFontFileSize	bool	optimize embedded fonts - exclude symbols which are not met in a report; if false then fonts are not changed; by default true
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true
AllowImageTransparency	bool	use the transparency in exporting images; by default true

15.4.3 Web Documents

There are two formats **HTML** (HyperText Markup Language), **HTML5** and **MHTML** (MIME HTML) are described in this chapter. The first and second formats are used for web page layout. The second format is a web page archive format used to bind resources together with the HTML code into a single file.

- HTML (HyperText Markup Language),
- MHTML (MIME HTML).

15.4.3.1 HTML

HTML (HyperText Markup Language) is the predominant markup language for Web pages. The majority of web pages are created using the HTML language. The HTML language is interpreted by browser and shown as a document. HTML is a tag language of the document layout. It provides a means to describe the structure of text-based information in a document by denoting certain text as links, headings,

© 2003-2024 Stimulsoft

paragraphs, lists, etc. Elements are the basic structure for HTML markup. Elements have two basic properties: attributes and content. Each attribute and each element's content has certain restrictions that must be followed for a HTML document to be considered valid. An element usually has a start tag (e.g. <element-name>) and an end tag (e.g. </element-name>).

Export Settings

The export parameters of the HTML export are described in the **StiHtmlExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
Zoom	double	zoom factor. By default a value is 1.0 what is equal 100% in export settings window
ImageFormat	ImageFormat	sets an image export format; by default ImageFormat.Png
ExportMode	StiHtmlExportMode	sets the mode of the document export using the div, span or table elements; by default StiHtmlExportMode.Table
ExportQuality	StiHtmlExportQuality	export quality of components size; by default StiHtmlExportQuality.High
Encoding	Encoding	file encoding; by default Encoding.UTF8
AddPageBreaks	bool	add page breaks; by default false
BookmarksTreeWidth	int	bookmark column width, in pixels; by default 150

© 2003-2024 Stimulsoft

Name	Type	Description
ExportBookmarksMode	StiHtmlExportBookmarksMode	a mode the export a document with bookmarks; by default StiHtmlExportBookmarksMode.All
UseStylesTable	bool	use the Styles table; if false then the style table is empty and all properties of each component will described directly in the style of this component; by default true

Static Options

Except the **StiHtmlExportSettings** class parameters of export to HTML are set using the static properties. All properties are described in the table below. To access to export properties it is necessary to add the **StiOptions.Export.Html...** prefix. For example, **StiOptions.Export.Html.ConvertDigitsToArabic**.

Name	Type	Description
ConvertDigitsToArabic	bool	convert ASCII digits to Arabic digits; by default false
ArabicDigitsType	enum	select Arabic digits type; by default Standard
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true
ForceWysiwygWordwrap	bool	Forcibly break text in rows as well as in the WYSIWYG mode; by default - false

© 2003-2024 Stimulsoft

Name	Type	Description
ReplaceSpecialCharacters	bool	change symbols '<', '>', '&', ' " ' on < > & " by default true

15.4.3.2 MHT

MHTML (MIME HTML) is a web page archive format used to bind resources which are typically represented by external links (such as images, Flash animations, Java applets, audio files) together with HTML code into a single file. This file is a web archive and has the «.mht» extension. The content of a file is written as an Email message using the MIME standard: in the beginning of a file the HTML file is written. Then all resources in the base64 encoding with headers are written. Internet Explorer, Opera, Microsoft Word can work with the MHTML format.

Export Settings

The export parameters of the MHT export are described in the **StiMhtExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
Zoom	double	zoom factor. By default a value is 1.0 what is equal 100% in export settings window
ImageFormat	ImageFormat	sets an image export format; by default ImageFormat.Png
ExportMode	StiHtmlExportMode	sets the mode of the document export using the div, span or table elements; by default StiHtmlExportMode.Table
ExportQuality	StiHtmlExportQuality	export quality of components size; by default

© 2003-2024 Stimulsoft

Name	Type	Description
		StiHtmlExportQuality.High
Encoding	Encoding	file encoding; by default Encoding.UTF8
AddPageBreaks	bool	add page breaks; by default false
BookmarksTreeWidth	int	bookmark column width, in pixels; by default 150
ExportBookmarksMode	StiHtmlExportBookmarksMode	a mode the export a document with bookmarks; by default StiHtmlExportBookmarksMode.All

15.4.4 Text Formats

This chapter describes exports formats of text files. In other words the files which are used to create text documents.

- TXT,
- RTF,
- Word 2007/2010,
- ODT.

15.4.4.1 TXT

Text file (TXT) is a kind of computer file that is structured as a sequence of lines. A text file exists within a computer file system. The end of a text file is often denoted by placing one or more special characters, known as an end-of-file marker, after the last line in a text file.

Text files are commonly used for storage of information.

Export Settings

The export parameters of the TXT export are described in the **StiTxtExportSettings** class. The description of all class properties are in the table below.

© 2003-2024 Stimulsoft

Name	Type	Description
Encoding	Encoding	text file coding; by default Encoding.UTF8
DrawBorder	bool	draw border lines; if false, then borders are not drawn; by default true
BorderType	StiTxtBorderType	a type of a border line; by default StiTxtBorderType.UnicodeSingle
KillSpaceLines	bool	remove all empty rows of a text; by default true
KillSpaceGraphLines	bool	remove all rows of a text which contains only blank spaces and symbols of the vertical border; by default true
PutFeedPageCode	bool	put feed page code after each page; by default true
CutLongLines	bool	cut too long lines of a text which cannot be placed in text boxes; by default true
ZoomX	float	horizontal zoom factor by X axis. By default a value is 1.0 what is equal 100% in export settings window
ZoomY	float	vertical zoom factor by Y axis. By default a value is 1.0 what is equal 100% in export settings window

Static Options

Static properties of export to TXT are shown on the table below. To access to export properties it is necessary to add the **StiOptions.Export.Txt...** prefix. For example,

© 2003-2024 Stimulsoft

StiOptions.Export.Txt.ColumnWidths.

Name	Type	Description
ColumnWidths	string	forcibly set the text column width (the list through the semicolon); if a row is empty then the column width is not changed; by default empty string
UseFullTextBoxWidth	bool	use all text box width for a text; in this case if the text is laid on a border, then the border is erased in this place; if false, then when drawing a text, one blank space on the right is always left for correct drawing borders; by default false
UseOldMode	bool	use the old mode of the text export; this property is left for keeping compatibility with old versions; by default false
UseFullVerticalBorder	bool	draw vertical border outside a cell. So a border will never be closed with a text; by default true
UseFullHorizontalBorder	bool	draw horizontal border outside a cell. So a border will never be closed with a text; by default true
CheckBoxTextForTrue	string	a text that shows the checkbox true status ; by default "+"
CheckBoxTextForFalse	string	a text that shows the

© 2003-2024 Stimulsoft

Name	Type	Description
		check false status; by default "-"

15.4.4.2 RTF

Rich Text Format (RTF) is a free document file format developed by Microsoft for cross-platform document interchange. The first version of the RTF standard appeared in 1987. Since that time format specification was changed and added. RTF-documents are supported by many text editors.

Export Settings

The export parameters of the RTF export are described in the **StiRtfExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution dpi; can take any value, by default 100
UsePageHeadersAndFooters	bool	process headers and footers of a page (see Table mode); by default false
ExportMode	enum	select export mode (see Common knowledge); by default StiRtfExportMode.Table
CodePage	int	this property is obsolete and is not used any longer, remained for compatibility with earlier versions

© 2003-2024 Stimulsoft

Static Options

Except the **StiRtfExportSettings** class parameters of export to RTF can be set using the static properties. All properties are described in the table below. To access to export properties it is necessary to add the **StiOptions.Export.Rtf...** prefix. For example, **StiOptions.Export.Rtf.UsePageRefField**.

Name	Type	Description
UsePageRefField	bool	when exporting a header with page numbers (for example, the "Anchors" report) the MS-Word "PAGEREF" command should be used for page numbers. Page numbers in the table of contents will be dynamically changed; if false, then numbers of pages will be static; by default true
ConvertDigitsToArabic	bool	convert ASCII digits into Arabic digits; by default false
ArabicDigitsType	enum	select type of Arabic digits; by default Standard
DivideSegmentPages	bool	divide segmented pages into separate pages; if false then are exported "as is" without dividing; by default true
LineHeightExactly	bool	export rows heights of a table "exactly"; if false then the height is exported as "at least"; by default true
RemoveEmptySpaceAtBott	bool	remove empty space on

© 2003-2024 Stimulsoft

Name	Type	Description
om		the bottom of a page; by default true
LineSpacing	double	coefficient of correction of a row height in multilined text fields; by default 0.965
RightMarginCorrection	int	correction of the right margin of a cell; by default 0
SpaceBetweenCharacters	int	sets space between characters of a font in twips; negative value corresponds to condensation; by default -2
UseCanBreakProperty	bool	use the CanBreak property when exporting rows of a table; by default true
DivideBigCells	bool	divide big cells into smaller ones for easier editing and scrolling; by default true

15.4.4.3 Word 2007/2010

Microsoft Word is a text processing software produces by Microsoft. It is a component of the Microsoft Office system. The first version was released for IBM PC's running DOS in 1983. Later there was a release for Apple Macintosh (1984), SCO UNIX, and Microsoft Windows (1989). Microsoft Word is the most popular text processors. Starting with first versions MS Word could write files in binary code with the «.doc» extension. The Word specification was secret and only in 2008 was published. The latest version of Word 2007/2010 "uses by default" the XML based format: Microsoft Office Open XML. For a new format the «.docx» file extension is used. This format is a zip-archive that contains a text as XML, graphics, and other data. When exporting, a report is converted into one table. Such a document is easy to edit.

© 2003-2024 Stimulsoft

Export Settings

The export parameters of the Word 2007 export are described in the **StiWord2007ExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution dpi; can take any value, by default 100
UsePageHeadersAndFooters	bool	process headers and footers of a page; by default false

Static Options

Static properties of export to Word 2007. To access to export properties it is necessary to add the **StiOptions.Export.Word2007...** prefix. For example, **StiOptions.Export.Word2007.DivideSegmentPages**.

Name	Type	Description
DivideSegmentPages	bool	divide segmented pages into separate pages; if false then are exported "as is" without dividing; by default true
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true

© 2003-2024 Stimulsoft

Name	Type	Description
LineHeightExactly	bool	export the rows height of a table "exactly"; if false then the height is exported as "at least"; by default true
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true
RightMarginCorrection	int	correction of the right margin of a cell; by default 0
SpaceBetweenCharacters	int	sets the space between characters of a font (in twips); negative value corresponds to condensed; by default -2

15.4.4.4 ODT

Open Document Text (**ODT**) is the open document for storing documents of the OpenOffice Writer, which is included into the OpenOffice.org package.

OpenOffice.org is the open package of office applications created as alternative to Microsoft Office. OpenOffice.org was one of the first what supported the new open OpenDocument. Works on Microsoft Windows and UNIX systems: GNU/Linux, Mac OS X, FreeBSD, Solaris, Irix.

OpenDocument Format (ODF) is the open file format for storing office documents, including text documents, spreadsheets, images, data bases, presentations. This format is based on the XML format.

OpenOffice Writer is the text processor and visual HTML editor, included into the OpenOffice. It is open software (LGPL license). Writer is similar to Microsoft Word and has approximately the same functionality. Writer allows saving documents in different formats including Microsoft Word, RTF, XHTML, and OASIS Open Document Format. Starting with the OpenOffice version 2.0, the OpenDocument Format is the default format for saving documents. File have the «.odt» extension.

© 2003-2024 Stimulsoft

When exporting the report is converted into a single table. The document is easily editable but some objects can be changed.

Export Settings

The export parameters of the ODT export are described in the **StiOdtExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution, dot per inch; may have any value, by default 100

Static Options

Static properties of export to ODT. To access to export properties it is necessary to add the **StiOptions.Export.Odt...** prefix. For example, **StiOptions.Export.Odt.DivideSegmentPages**.

Name	Type	Description
DivideSegmentPages	bool	divide segmented pages into separate pages; if false then are exported "as is" without dividing; by default true
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true

© 2003-2024 Stimulsoft

Name	Type	Description
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true

15.4.5 Spreadsheets

This group of exports create spreadsheets. They are exports to both different formats of Microsoft Excel and to OpenOffice Calc.

- Excel,
- Excel 2007/2010,
- ODS.

15.4.5.1 Excel

Microsoft Excel is a spreadsheet application written and distributed by Microsoft for Microsoft Windows. It allows using calculation, graphing tools, pivot tables and a macro programming language called VBA. So, it is the most popular table processor available for these platforms since version 5 in 1993.

Microsoft Excel up until Excel 2007 version used a proprietary binary file format called Binary Interchange File Format (BIFF) and **.xls** file extension. Specification was closed but since 2008 it was published. Besides, most of Microsoft Excel can read CSV, DBF, SYLK, DIF, and other formats.

Export Settings

The export parameters of the XLS export are described in the **StiExcelExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution, dot per inch; may have any value,

© 2003-2024 Stimulsoft

Name	Type	Description
		by default 100
UseOnePageHeaderAndFooter	bool	remove from a report all page headers (except the first one) and all page footers (except the last one); by default false
ExportDataOnly	bool	export data only, e.g. all components which are placed on data bands; by default false
ExportPageBreaks	bool	export page breaks; by default false
ExportObjectFormatting	bool	export object formatting; by default true
ExportEachPageToSheet	bool	export each page of a report as a sheet; by default false

The **ExportObjectFormatting** property works only if the ExportDataOnly is set to true.

Static Options

Static properties of export to Excel. To access to export properties it is necessary to add the **StiOptions.Export.Excel...** prefix. For example, **StiOptions.Export.Excel.AllowExportDateTime**.

Name	Type	Description
AllowExportDateTime	bool	export date and time; if false then date and time are exported as text strings; by default false
ColumnsRightToLeft	bool	set the order of columns

© 2003-2024 Stimulsoft

Name	Type	Description
		from right to left; by default false
MaximumSheetHeight	int	maximal number of rows on a sheet; remaining rows are transferred on the next sheet; by default 65534
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true
ShowGridLines	bool	show grid lines; by default true
DivideBigCells	bool	divide big cells into smaller ones for easier editing and scrolling; by default true

15.4.5.2 Excel 2007/2010

For storing documents as the basic Microsoft Excel format, right up to the Excel 2007 version, used its own binary format of files (BIFF) and the file extension was «.xls». In **Excel 2007/2010**, the basic format is the Microsoft Office Open XML format and stores document in files with the «.**xlsx**» extension. The Excel 2007 is compatible with binary formats such as CSV, DBF, SYLK, DIF, and others.

Export Settings

The export parameters of the Excel 2007 export are described in the **StiExcel2007ExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to1.0 (the highest quality); by default 0.75

© 2003-2024 Stimulsoft

Name	Type	Description
ImageResolution	float	image resolution, dot per inch; may have any value, by default 100
UseOnePageHeaderAndFooter	bool	remove from a report all page headers (except the first one) and all page footers (except the last one); by default false
ExportDataOnly	bool	export data only, e.g. all components which are placed on data bands; by default false
ExportPageBreaks	bool	export page breaks; by default false
ExportObjectFormatting	bool	export object formatting; by default true
ExportEachPageToSheet	bool	export each page of a report as a sheet; by default false

The **ExportObjectFormatting** property works only if the **ExportDataOnly** is **true**.

Static Options

Static properties of export to Excel 2007.

Name	Type	Description
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true

© 2003-2024 Stimulsoft

Name	Type	Description
ColumnsRightToLeft	bool	set the order of columns from right to left; by default false
MaximumSheetHeight	int	maximal number of rows on a sheet; odd rows are moved to the next sheet; by default 1048574
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true

15.4.5.3 ODS

Open Document Spreadsheet (**ODS**) is the opened format to store OpenOffice Calc spreadsheet documents, that is included into the OpenOffice.org package.

OpenOffice.org is a free package of office applications developed as alternative to Microsoft Office. The OpenDocument is one of the first what started to support the opened format. it works on Microsoft Windows and UNIX-like systems: GNU/Linux, Mac OS X, FreeBSD, Solaris, Irix.

OpenDocument Format (ODF) — an open document file format for storing and exchanging editable documents including text documents (such as notes, reports, and books), spreadsheets, drawings, databases, presentations. The format is based on the XML-format. The standard was jointly developed by public and various organizations and is available to all and can be used without restrictions.

OpenOffice Calc is the table processor that is included into the OpenOffice and is a free software (LGPL license). Calc is similar to the Microsoft Excel spreadsheet and functionality of these processors is approximately equal. Calc allows you to saving documents to various formats, including Microsoft Excel, CSV, HTML, SXC, DBF, DIF, UOF, SLK, SDC. Starting with version OpenOffice 2.0, for document storage format by default OpenDocument Format, files are saved with the extension «. Ods». Starting with the OpenOffice version 2.0 for storing documents, by default, the OpenDocument Format is used. Files are stored with the «.ods» extension.

Export Settings

© 2003-2024 Stimulsoft

The export parameters of the ODS export are described in the **StiOdsExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ImageQuality	float	image quality; may have values from 0.0 (the lowest quality) to1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution, dot per inch; may have any value, by default 100

Static Options

Static properties of export to ODS. To access to export properties it is necessary to add the **StiOptions.Export.Ods...** prefix. For example, **StiOptions.Export.Ods.AllowImageComparer**.

Name	Type	Description
AllowImageComparer	bool	use the image comparer, e.g. replace image duplicates (see Common export settings); if false then an image is exported "as is"; by default true
DivideSegmentPages	bool	divide segmented pages into separate pages; if false then are exported "as is" without dividing; by default true
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true

© 2003-2024 Stimulsoft

15.4.6 Data

This is a group of file formats which are used to store table data.

- CSV
- DBF
- XML
- DIF
- SYLK

15.4.6.1 CSV

CSV (Comma Separated Values) is a text format that is used to represent table data. Each string of the file is one row of the table. The values of each column are separated by the delimiter that depends on regional settings. The values that contain reserved characters (such as a comma or a new string) are framed with the double quotes (") symbol; if double quotes are found in the value they are represented as two double quotes in the file.

Information

Only those data (components) can be exported to the **CSV** format which are placed on data bands. If the **SkipColumnHeaders** property is set to false then, additionally, column headers are exported as the first row.

Export Settings

The export parameters of the CSV export are described in the StiCsvExportSettings class. The description of all class properties are in the table below.

Name	Type	Description
Separator	string	sets the symbol-separator of a list that is used when exporting; by default CurrentCulture.TextInfo.ListSeparator
Encoding	Encoding	text file coding; by default Encoding.UTF8

© 2003-2024 Stimulsoft

SkipColumnHeaders	bool	skip headers of columns; by default false

Static Options

Static properties of export to CSV. To access to export properties it is necessary to add the **StiOptions.Export.Csv...** prefix. For example, **StiOptions.Export.Csv.ForcedSeparator**.

Name	Type	Description
ForcedSeparator	string	sets the separator forcibly which are used in export; if the empty string is set then the symbol from export settings in used; by default - empty string

15.4.6.2 DBF

The **DBF** (DataBase File) is the format to store data and it is used as the standard way to store and pass information. The DBF file consist of a header section for describing the structure of the data in the file. There are several variations on the .dbf file structure.

Information

Only data can be exported to the DBF format, in other words only the components, which are placed on data bands.

Controlling Exports

The following elements can be specified in the Tag field to control export:

- DataType [: FieldLength [: DecimalPartLength]],
- ExportType : "FieldName",
- Column: "FieldName" "DataString".

© 2003-2024 Stimulsoft

Several elements should be separated with the semicolon. The “DataType" element should be only one and should be placed first, other elements – if necessary.

Values of the "DataType" element are shown in the table below. If the data type is not set, then the string data type is taken by default. The "FieldLength" element sets fixed width of a data field. If the field width is not set, then the width is taken from the table. For the **string** type the default width is the longest **string**. The "DecimalPartLength" element sets the number of characters after comma. If it is not set, then the default number is taken.

Data type	DBF data type (default size)	Description
int	Numeric (15 : 0)	Numeric
long	Numeric (25 : 0)	Numeric
float	Numeric (15 : 5)	Decimal
double	Numeric (20 : 10)	Decimal
string	Character (auto)	Text
date	Date (8)	Date

Sample of using elements are shown in the table below.

Type	Description
string : 25	set the column width (25 characters) and cuts all long strings
float	converts decimal digit with the length 15 characters, 5 characters after comma
float :10	converts decimal digit with the length 10 characters, 5 characters after comma
float :10 : 2	converts decimal digit with the length 10 characters, 2 characters after comma
int :10 : 2	converts integer digit with the length 10 characters; the second parameter is ignored

© 2003-2024 Stimulsoft

Information

If the integer part of a digit is long and cannot be placed into the specified field, then it is cut, so data are lost. For example, if the write «-12345,678» in the «float:8:3» field, then the «2345,678» will be output.

The "**ExportType**" element indicates for which export the field name is set. The values can be used: “dbf”, “csv”, “xml”, “default”. The "FieldName" element indicates the field name in the file (for the DBF the is automatically cut up to 10 characters). The own name can be specified to each type of export. If the name for each export is not specified then the name for the “default” type is taken. For example:

```
DBF : "Describe" ; XML : "Description" ; default: "Default name"
```

The "Column" element indicates that the additional field is added to the exported data. The "FieldName" element indicates the name of a new field. The "DataRow" element indicates the content of a new field and can be expression. For example

```
Column: "SortField" "{Products.Categories.CategoryName}"
```

Export Options

The export parameters of the DBF export are described in the **StiDbfExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
CodePage	StiDbfCodePages	a code page of a file; by default StiDbfCodePages.Default

15.4.6.3 XML

XML (eXtensible Markup Language) is a text format that is used to store structured data (in exchange for existed files of data bases), for exchange of information between programs and also to create on its base the special markup languages (for example, XHTML), sometimes called dictionaries. XML is the hierarchical structure

© 2003-2024 Stimulsoft

that is used to store any data. Visually this structure can be represented as the tree. XML supports Unicode and other encoding.

Information

Only those data (components) are exported to the XML format which are placed on data bands.

Controlling Exports

The following elements can be specified in the Tag field to control export to XML:

- DataType
- ExportType : "FieldName"
- Column: "FieldName" "DataRow"

Several elements should be separated with the semicolon. The “DataType" element should be only one and should be placed first, other elements – if necessary.

Values of the "DataType" element are shown in the table below. If the data type is not set, then the **string** data type is taken by default.

Data type	Description
int	Numeric
long	Numeric
float	Decimal
double	Decimal
string	Text
date	Date

The "**ExportType**" element indicates for which export the field name is set. The values can be used: “dbf”, “csv”, “xml”, “default”. The "FieldName" element indicates the field name in the file. The own name can be specified to each type of export. If the name for each export is not specified then the name for the “default” type is

© 2003-2024 Stimulsoft

taken. For example:

```
DBF : "Describe" ; XML : "Description" ; default: "Default name"
```

The "Column" element indicates that additional field is added to the exported data. The "FieldName" element indicates the name of a new field. The "DataRow" element indicates the content of a new field and can be expression. For example:

```
Column: "SortField" "{Products.Categories.CategoryName}"
```

15.4.6.4 DIF

DIF (Data Interchange Format) is a text format that is used to exchange sheets between spreadsheets processors (Microsoft Excel, OpenOffice.org Calc, Gnumeric, StarCalc, Lotus 1-2-3, FileMaker, dBase, Framework, Multiplan, etc). The only limitation of this format is that the DIF format may contain only one sheet in one book.

Export Settings

The export parameters of the DIF export are described in the **StiDifExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ExportDataOnly	bool	export data only. e.g. only components placed on data bands; by default false
Encoding	Encoding	file encoding; by default Encoding.ASCII
UseDefaultSystemEncoding	bool	use the default system encoding; if false then use encoding that is set by the Encoding property; by default true

© 2003-2024 Stimulsoft

15.4.6.5 SYLK

SYLK (Symbolic Link) format- this text format is used to exchange data between applications, specifically spreadsheets. Files of SYLK have «.slk» extension. Microsoft does not publish a SYLK specification, therefore work with this format in different applications can be different.

Information

A SYLK file can be written in Unicode and read by some applications but anyway many applications which do support Unicode writes SYLK files into ANSI but not Unicode. Therefore, symbols which do not have representation in the system code page will be written as ('?') symbols.

Export Settings

The export parameters of the SYLK export are described in the **StiSylkExportSettings** class. The description of all class properties are in the table below.

Name	Type	Description
ExportDataOnly	bool	export data only. e.g. only components placed on data bands; by default false
Encoding	Encoding	file encoding; by default Encoding.ASCII
UseDefaultSystemEncoding	bool	use the default system encoding; if false then use encoding that is set by the Encoding property; by default true

15.4.7 Images

Export groups to graphic formats. All graphic formats can be divided in to types: bitmapped images and vector formats. Notice. On the current moment the export of

© 2003-2024 Stimulsoft

monochrome image is supported only to the BMP, GIF, PCX, PNG, TIFF format. So the DitheringType property works only for these exports.

Export Parameters

All exports of images have the same export settings. They are described in the table below. But each format has its own ExportSettings class. For BMP, GIF, PNG, TIFF, JPEG, PCX, and EMF the following classes are used in exports. The StiBmpExportSettings is used for export to BMP, **StiGifExportSettings** is used for export to GIF, **StiPngExportSettings** is used for export to PNG, **StiTiffExportSettings** is used for export to TIFF, **StiJpegExportSettings** is used for export to JPEG, **StiPcxExportSettings** is used for export to PCX, and **StiEmfExportSettings** is used for export to EMF.

Name	Type	Description
ImageZoom	double	zoom factor. By default a value is 1.0 what is equal 100% in export settings window
CutEdges	bool	cut page edges; by default false
ImageFormat	StiImageFormat	Image format - colored, tint of grey or monochrome; by default StiImageFormat.Color
MultipleFiles	bool	saves pages of a report into separate files; can be used for TIFF only, because it can save some pages into one file, other formats save pages into separate files; by default false
DitheringType	StiMonochromeDitheringType	a type of image dithering to get monochrome image; by default

© 2003-2024 Stimulsoft

		StiMonochromeDitheringType.FloydSteinberg

15.5 Scripts

Stimulsoft Reports supports a choice of languages for report generation.

- Programming Language of Report,
- Report Code.

15.5.1 Programming Language of Report

The report generator uses a single specified programming language to generate the report code and handle report events. If the current programming language of a report does not suit your requirements you can change it. The options are currently C# or VB.NET.

Changing The Language Of The Current Report

To do this select **File | Report Setup**. A new dialog will be displayed

In the **Language** group select a new programming language and press **Ok**. The current programming language will then be changed.

Information

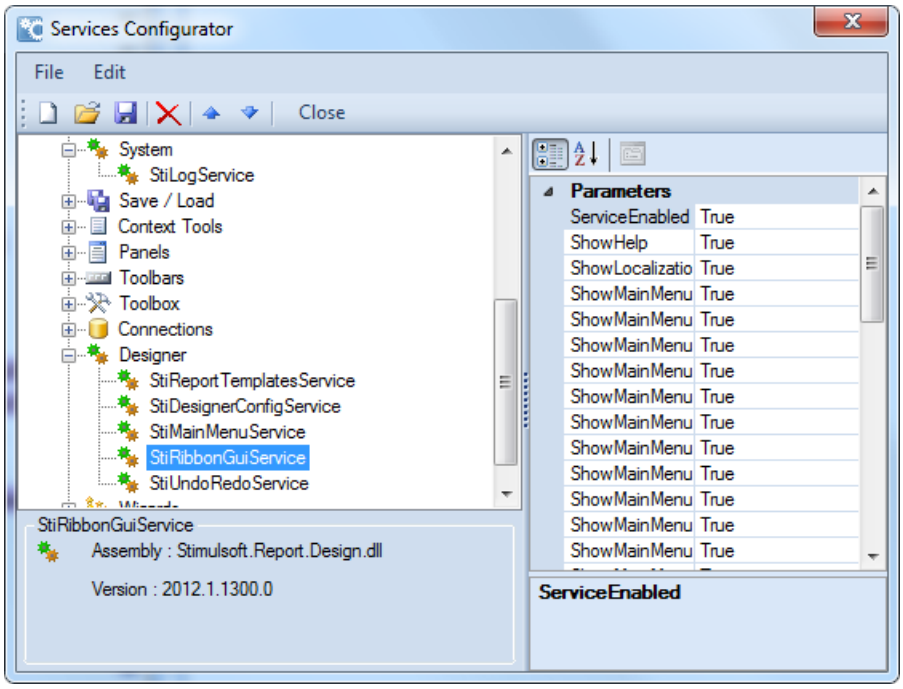
The underlying report code will have to be regenerated for the entire report, and any changes which have been made within the report code will be lost.

It is not convenient to change the programming language each time a report is rendered, so Stimulsoft Reports allows you to set the default programming language used by all new reports.

To do this you should use the Services Configurator utility. All programming languages in Stimulsoft Reports are located under the Languages node. The language which is shown first in the list is the default programming language. For example, on the picture below the C# programming language is the default programming language. To make VB.Net the default programming language simply

© 2003-2024 Stimulsoft

drag the **StiCSharpLanguage** service down one position with the mouse or use the up and down buttons to re-order the languages.



15.5.2 Report Code

When you create a new report its source code, often called the report script, is generated automatically using either C# or VB.NET programming language depending on the currently selected default. You can use only one of these programming languages at a time

In the report code the structure and initialization of the report class, which itself inherits from the StiReport class, are described. When adding new pages, components or changing any parameters of a report, those changes are automatically recorded within the class. The report class therefore contains a description of all components, data, events, report properties, and data source structures for the report. Any events specified by the user are also added to the report code.

For the ultimate in power and flexibility Stimulsoft Reports allows direct editing of the report code - if you want something not provided by the available properties and features of the designer you can actually code your own features within the report. When writing events or another code in the report, you use the standard syntax of the selected .Net Framework programming language i.e. if the language is set to C# you write code using C# syntax.

© 2003-2024 Stimulsoft

Information

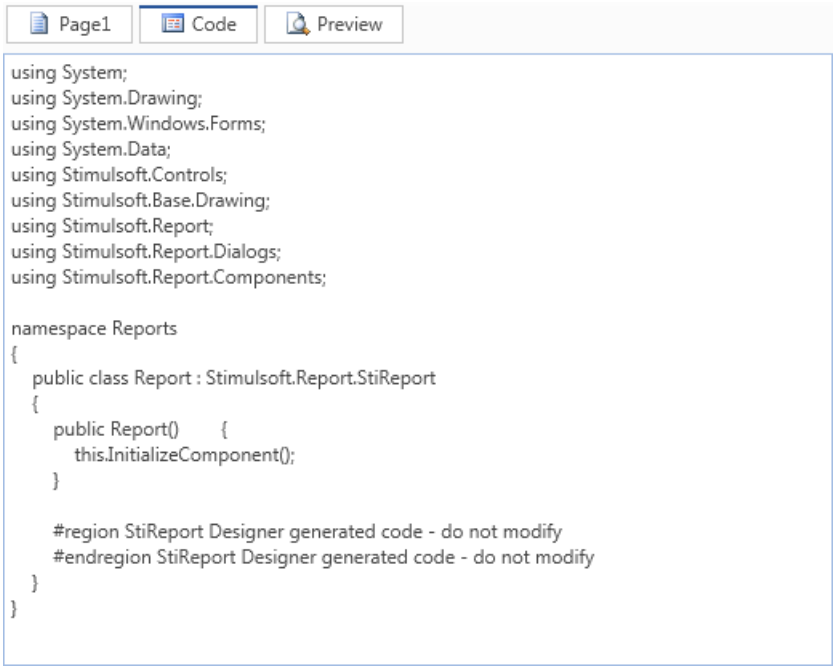
The report code is generated in **C#** or **VB.Net** programming language. All events and any another code in this report must be written in the currently selected language.

When rendering reports, compilation of the report class occurs first. After that the compiled report is executed.

Information

The report code is compiled using the .NET Framework compiler.

To see the report code click the Code tab in the designer. The code will then be displayed:



© 2003-2024 Stimulsoft

To edit the code, simply start typing in the appropriate place.

Infromation

Do not change preprocessor directives or automatically updated code.

Whilst Stimulsoft Reports allows you to directly edit the report code, it is important to remember that it is impossible to make changes in the parts of the report code which are automatically updated - such changes will be lost when the next update takes place. The automatically updated report code is enclosed in Region preprocessor directives:

VB.NET

```
...
//At the beginning of the automatically updated code
#region StiReport Designer generated code - do not modify

//Automatically updated code goes here

//the end of the automatically updated code
#endregion StiReport Designer generated code - do not modify
...
```

Any code that you write within the report must be written outside these Regions.

16 Forms.WEB

Stimulsoft Forms is a tool for creating, editing, filling, publishing, distributing interactive forms, and collecting results.

16.1 Get Started for ASP.NET Core 3.1

1. Create the **ASP.NET Core Empty** application;
2. Install the **Stimulsoft.Forms.Web** component into the application from the **NuGet** package manager;
3. In **Startup.cs**, configure the application to use controllers:

Startup.cs

© 2003-2024 Stimulsoft

```
public void ConfigureServices(IServiceCollection services)
{
   services.AddControllers();
}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
   if (env.IsDevelopment())
   {
      app.UseDeveloperExceptionPage();
   }

   app.UseDefaultFiles();
   app.UseStaticFiles();

   app.UseRouting();

   app.UseCors(builder =>
   builder.AllowAnyOrigin().AllowAnyHeader().AllowAnyMethod());

   app.UseEndpoints(endpoints =>
   {
      endpoints.MapControllerRoute(
         name: "default",
         pattern: "{controller=Forms}/{action=Action}");
   });
}
```

4. Create the **Controllers** folder;
5. Add **FormsController.cs** to the **Controllers** folder, which will process requests from the **Stimulsoft Forms** component:

FormsController.cs

```
public class FormsController : Controller
{
   [HttpPost]
   public IActionResult Action()
   {
      try
      {
         var data = JObject.Parse(this.HttpContext.Request.Form["data"]);
         var action = data["action"].ToString();
         switch (action)
         {
            case "Initialize":
               var initData = StiWebForm.Initialize(data, null);
               return Json(initData.Content);

            default:
               var result = StiWebForm.ProcessRequest(data);
```

© 2003-2024 Stimulsoft

```
            return result.ContentType switch
            {
               "application/pdf" => new FileContentResult(result.Content as
               byte[], result.ContentType),
               _ => Json(result.Content),
            };
         }
      }
      catch (Exception e)
      {
         return new ContentResult()
         {
            Content = e.Message,
            ContentType = "text/plain"
         };
      }
   }
}
```

6. Inside the application folder, run the command in the terminal to create a new **Angular** application with no `routing` and CSS styling:

console

```
ng new sti-forms-designer --routing false --style css
```

7. Rename the **sti-forms-designer** folder to **ClientApp**;
8. In the **package.json** file, add dependencies for **Stimulsoft Forms** and the necessary components:

package.json

```
"stimulsoft-forms": "^2023.2.4",
"ngx-color-picker": "^13.0.0",
"primeicons": "^6.0.1",
"primeng": "^14.1.2",
"resize-observer-polyfill": "^1.5.1",
"rxjs": "~6.5.0"
```

9. Add styles to the **angular.json** file and improve the `budgets`:

© 2003-2024 Stimulsoft

angular.json

```
"build": {
   "options": {
      "styles": [
         "src/styles.css",
         "node_modules/primeicons/primeicons.css",
         "node_modules/primeng/resources/primeng.min.css",
         "node_modules/stimulsoft-forms/theme.css"
      ]},

      "configurations": {
         "production": {
            "budgets": [
               {
                  "type": "initial",
                  "maximumWarning": "500kb",
                  "maximumError": "5mb"
               }]
            }
         }
      }
   }
}
```

10. Update the imports in **ClientApp\src\app\app.module.ts**:

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from "@angular/forms";
import { HttpClientModule } from "@angular/common/http";
import { BrowserAnimationsModule } from "@angular/platform-browser/
animations";
import { DropdownModule } from "primeng/dropdown";
import { StimulsoftFormsModule } from 'stimulsoft-forms';
import { AppComponent } from './app.component';


@NgModule({
   declarations: [
      AppComponent
   ],
   imports: [
      BrowserModule,
      HttpClientModule,
      FormsModule,
      BrowserAnimationsModule,
      DropdownModule,
      StimulsoftFormsModule
```

© 2003-2024 Stimulsoft

```
  ],
  providers: [],
  bootstrap: [AppComponent]
})
```

11. Update **ClientApp\src\app\app.component.ts**:

app.component.ts

```
import { Component } from '@angular/core';
import { StimulsoftFormsService } from 'stimulsoft-forms';

@Component({
  selector: 'app-root',
  template: `
    <stimulsoft-forms
      [requestUrl]="'http://localhost:7536/Forms/Action'" /*URL to Forms
      controller, you can find this url in launchSettings.json file of
      the project*/
      [form]="form" /*StiForm object to use to create form object using
      StimulsoftFormsService*/
      [style.width]="'100%'"
      [style.height]="'100%'">
    </stimulsoft-forms>
`
})

export class AppComponent {
  public form!: any;

  constructor(public formService: StimulsoftFormsService) {
    this.form = this.formService.createElement("Form");
  }
}
```

12. Update **ClientApp\src\app\index.html**:

index.html

```
<!doctype html>
<html lang="en" style="position: fixed;width:100%;height:100%">
<head>
  <meta charset="utf-8">
  <title>StiFormsDesigner</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
```

© 2003-2024 Stimulsoft

```
</head>
<body style="width:100%;height:100%">
  <app-root></app-root>
</body>
</html>
```

13. Go to the **ClientApp** folder;
14. Run the command to install **Angular** components:

console

```
npm i --force
```

15. Run the command to build an **Angular** application:

console

```
ng build --output-hashing none
```

16. Create the **wwwroot** folder in the project's root folder;
17. Copy all files from **ClientApp\dist\sti-forms-designer** into **wwwroot**;
18. Run the project.

© 2003-2024 Stimulsoft