

Table of Contents

Part I Introduction	1
1 Technical Support	2
2 Trial License Limitations	3
3 Information in Web and References	3
Part II Data	4
1 Business Objects in Net, Web, Silverlight	4
2 Business Objects in WinRT	12
3 Working with OData Using Business Objects	14
Part III WinForms Viewer	15
1 How to Show Report?	15
2 Dot-Matrix Viewer for WinForms	16
Setting Dot-Matrix Viewer in WinForms	16
Dot-Matrix and Escape Codes	18
Part IV WPF Viewer	19
1 How to Show Report?	19
2 Dot-Matrix Viewer for WPF	20
Dot-Matrix Viewer Settings for WPF	20
DotMatrix and Escape Codes	22
Part V Web Designer	23
1 How It Works?	23
2 How to Run Web Report Designer?	24
3 Loading Reports to Web Designer	25
4 Report Preview	27
5 Changing Report Settings Before Rendering	28
6 Saving Reports	28
7 Web Designer Settings	30
Connection	30
Main Menu	30
Zooming Static Properties	32
Viewer Static Properties	32
Additional Viewer Static Properties	35
8 Web Designer Properties	36
9 Changing Web Designer Properties From Code	37
10 Web Report Designer Localization	37
11 Visible Mode	38

Part VI Web Viewer 38

1 Caching	38
RenderMode Property	39
CacheMode Property	40
ServerTimeout Property	40
2 Printing Reports	40
3 StiReportResponse Class	40
4 Using Dialogs in Report	41
5 Images of StiWebViewer Toolbar	41
6 Localization of StiWebViewer Component	41

Part VII MVC Designer Fx 42

1 How it works?	42
2 Editing and Creating Reports	42
3 Previewing Reports	45
4 Preview Features	47
5 Previewing C#/VB.NET Report Code	49
6 Saving Reports	50
7 Localizing Designer	54
8 Additional Actions of Designer	57
9 Additional Methods	59
10 MVC Designer Fx Settings	60
Actions	62
Server	62
Appearance	63
Data Dictionary	64
Main Menu	65
Toolbar	66
Preview Panel	67
Behavior	68
Exporting	69
Send Email	71
Printing	71

Part VIII MVC Viewer 71

1 How It Works?	72
2 Showing Reports	72
3 Localizing Component	76
4 Using Themes	76
5 Using Basic Features	79
6 Printing	80
7 Export reports	81
8 Viewing Modes	83

9 Using Parameters	85
10 Bookmarks	88
11 Dynamic sorting and drill down reports	90
12 Send Email	94
13 Helpful Methods	97
14 Viewer Settings	99
Actions	101
Server	102
Appearance	103
Toolbar	104
Exporting Reports	106
Sending Email	109
Part IX MVC ViewerFx	109
1 How It Works?	109
2 How to Show Report?	110
3 Localizing Component	113
4 Using Themes	113
5 Exporting Reports	114
6 Interactive Reports	116
7 Send Email	118
8 Additional Methods	120
9 Viewer Settings	121
Actions	123
Server	124
Appearance	124
Toolbars	125
Exporting	127
Send Email	129
Printing	129
Part X Web ViewerFx	130
1 How to Show Report?	130
2 Localization of StiWebViewerFx Component	131
3 Using Themes in WebViewerFx	131
4 WebViewerFx Settings	132
Connection	132
Zooming	133
Viewer Static Properties	133
5 Properties	135
6 Export Settings	138
7 Data	140
Part XI Flex Viewer	141
1 How to Show Report?	141

2 Dialog Options	142
3 User Interface Settings	142
Part XII Web Designer in Silverlight	143
1 How It Works?	144
2 How to Run Web Report Designer?	144
3 Loading Reports to Web Designer	144
4 Report Preview	145
5 Web Designer Settings	146
Main Menu	146
Zooming	147
Viewer	148
6 Changing Web Designer Properties from Code	151
7 Web Report Designer Localization	152
8 WCF Server	152
Part XIII Silverlight Web Viewer	153
1 How to Show Report?	153
2 WebViewerSL Settings	153
Control Panel	154
Navigation Panel	155
Zooming	156
3 Saving Mode	157
Export Settings	158
Part XIV HTML5 Designer	160
1 Designer Properties	160
2 Working With Report Code	161
Part XV HTML5 Viewer	163
1 Showing Reports	163
2 Caching	164
RenderMode Property	164
CacheMode Property	165
ServerTimeout Property	165
3 StiReportResponse Class	165
4 Localization of StiMobileViewer Component	166
5 Settings	166
Zooming	166
Toolbar	166
6 Properties	167
7 Setting Export	168
8 Defining Data	169

Part XVI HTML5 MVC Designer	170
1 How It Works?	170
2 How to Run Designer?	170
3 Loading Reports	171
4 Previewing Reports	172
5 Saving Reports	173
6 Localizing HTML5 Designer	174
7 Other Actions of Designer	174
8 HTML5 Designer Settings	176
Actions	176
Server	177
File Menu	177
Interface	178
Bands	179
Components	180
Part XVII Stimulsoft Reports.JS	181
1 Connecting Library	181
2 Loading and Saving Report	184
3 Getting Access to Pages	185
4 Rendering Report	185
5 Binding Data to Report	185
6 Connect to MySQL and MS SQL Database	187
7 Synchronize Data between DataStore and Dictionary	188
8 Saving Rendered Report	189
9 Report Printing	189
10 Running Report Viewer	189
11 Viewer Events	191
12 Viewer Options	194
Appearance	195
Toolbar	196
Exports	198
13 Running Report Designer	199
14 Designer Events	201
15 Designer Options	204
Appearance	204
Toolbar	204
Bands	205
Cross-Bands	206
Components	207
Dictionary	207
16 Exporting Rendered Report	209

Part XVIII Java Viewer	212
1 Showing Reports	213
2 Custom Functions	213
Part XIX Java with Flex Client	215
1 Installation	215
2 Creating Project	215
3 Creating Server	217
4 Creating Sample	219
5 Creating Sample Page with Report Designer	222
6 Loading, Saving and Loading Custom Data	227
Part XX Java HTML5 Designer	235
1 Installation and Description HTML5 Designer	235
2 Template JDBC Coonections	240
Part XXI Java HTML5 Viewer	240
1 Installation	240
2 Creating Project	241
3 Creating a Sample Page	244
4 Create a Sample Page With Report HTML5 Viewer	248
5 Description of Webviewer Tag	251
6 Options	252
7 Template JDBC Connections	255
Part XXII WinRT Viewer	255
1 How to Show Report?	256
2 Saving Report From Code	257
Part XXIII WinRT Designer	258
1 Working with Report Code	258
Part XXIV Exports	259
1 Available File Formats	260
2 Export Reports From Code	261
ExportDocument Method	261
Export Formats	262
Export Service	263
All Export Services	264
3 Formats with Fixed Page Layout	265
PDF	265
Embedded Fonts.....	265

Digital Signature.....	266
Digital Signature from Code.....	266
Encryption.....	266
Using Parameters of Encryption from Code.....	267
Editable Fields.....	268
Export Settings.....	268
Static Options.....	269
XPS	270
Export Settings.....	271
Static Options.....	271
Microsoft Power Point 2007/2010	271
Export Settings.....	272
Static Options.....	272
4 Web Documents	272
HTML	272
Export Settings.....	273
Static Options.....	273
MHT	274
Export Settings.....	274
5 Text Formats	275
TXT	275
Export Settings.....	275
Static Options.....	276
RTF	276
Export Settings.....	277
Static Options.....	277
Word 2007/2010	278
Export Settings.....	278
Static Options.....	278
ODT	279
Export Settings.....	279
Static Options.....	280
6 Spreadsheets	280
Excel	280
Export Settings.....	281
Static Options.....	281
Excel 2007/2010	282
Export Settings.....	282
Static Options.....	282
ODS	283
Export Settings.....	283
Static Options.....	284
7 Data	284
CSV	284
Export Settings.....	284
Static Options.....	285
DBF	285
Controlling Exports.....	285
Export Settings.....	286
XML	287
Controlling Exports.....	287
DIF	288
Export Settings.....	288

SYLK	288
Export Settings.....	288
8 Images	289
Export Parameters	289
Part XXV Report Inheritance	290
1 Basic Approaches	290
Part XXVI Scripts	291
1 Programming Language of Report	291
2 Report Code	292
Part XXVII Right To Left	294
1 WinForms Report Viewer	294
2 Icons	294
3 WPF Report Designer and Viewer	299
Part XXVIII Deployment	300
1 Assemblies in Reports.Net	301
2 Assemblies in Reports.Wpf	302
3 Assemblies in Reports.Web	304
4 Assemblies in Reports Designer.Web	305
5 Assemblies in Reports.Silverlight	306
6 Assemblies in Reports Designer.Silverlight	308
7 Assemblies in Reports.Ultimate	308
8 Assemblies in Reports.Fx for Flex	312
9 Assemblies in Reports.Fx for PHP	312
10 Assemblies in Reports.Fx for Java	313
11 Working With Assemblies	313
12 Redistributable files in Reports.Net	315
13 Redistributable files in Reports.Wpf	315
14 Redistributable files in Reports.Web	315
15 Redistributable files in Report Designer.Web	316
16 Redistributable files in Reports.Silverlight	316
17 Redistributable files in Reports Designer.Silverlight	316
18 Redistributable files in Reports.Ultimate	316
19 Redistributable files in Reports.Fx for Flex	317
20 Redistributable files in Reports.Fx for PHP	317
21 Redistributable files in Reports.Fx for Java	318
22 Deployment in Windows	318
23 Deployment in Web	318

24 Deployment in Designer.Web	318
25 Deployment Reports as Files	318
26 Reports as Source Code	319
27 Reports as Assemblies	321
28 Standalone Reports	322

Index**0**

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 License Limitations](#)
- ⓘ [Information in Web and References](#)
- ⓘ [Evaluate Trial Version](#)



Using HTML5 Components:

- ⓘ [Using HTML5 Designer](#)
- ⓘ [Designer Properties](#)
- ⓘ [Using HTML5 Viewer](#)
- ⓘ [Settings](#)



Data:

- ⓘ [Business Object in Net, Web, Silverlight](#)
- ⓘ [Business Object in WinRT](#)



Reports in Java:

- ⓘ [Showing Reports in Java](#)
- ⓘ [Using Java in Web](#)



Showing Reports in WinForms:

- ⓘ [How to Show Report](#)
- ⓘ [Dot-Matrix viewer](#)



Showing Reports in WinRT:

- ⓘ [How to Show Report?](#)
- ⓘ [Saving Report From Code](#)



Using Flex Components:

- ⓘ [Using MVC ViewerFx](#)
- ⓘ [Using Web ViewerFx](#)
- ⓘ [Using_Flex_Viewer](#)



Exports:

- ⓘ [Available File Formats](#)
- ⓘ [Spreadsheets](#)
- ⓘ [More...](#)



Showing Reports in WPF:

- ⓘ [How to Show Report?](#)



Report Inheritance:

- ⓘ [Basic Approaches](#)



Reports in Web:

- ⓘ [How to Run Web Report Designer?](#)
- ⓘ [Using Web Viewer](#)



Scripts:

- ⓘ [Programming Language of Report](#)
- ⓘ [Report Code](#)



- ① [Using MVC Designer](#)
- ① [Using Mvc Viewer](#)



- ① [WinForms Report Viewer](#)
- ① [Icons](#)
- ① [WPF Report Designer and Viewer](#)



- ① [Using Web Designer in Silverlight](#)
- ① [Using Silverlight Web Viewer](#)



- ① [Working With Assemblies](#)
- ① [More...](#)

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.

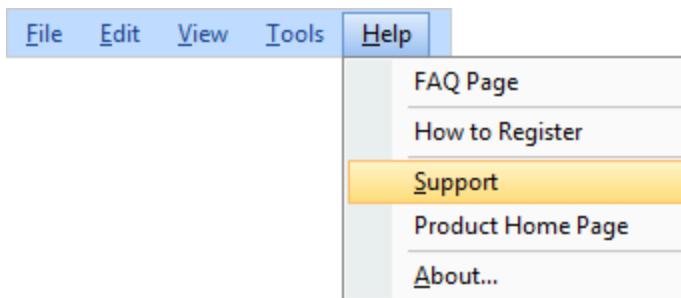
For technical questions, use the Email address: support@stimulsoft.com

For licensing, 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 <http://www.stimulsoft.com/support.aspx>

It is possible to send questions from the standard UI of the report designer. To do this, select the **Help** menu -> **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;
- 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 License Limitations

Trial Versions

The free trial versions of Stimulsoft Reports are **fully functional** and will work for an unlimited time. The only limitation is a DEMO watermark displayed on each report page.

Registered Versions

Developer licenses come with DLL's, which work without license keys. If you have a License for Stimulsoft products, you only need to ensure that you are using the Registered build. If your reports are displaying a DEMO watermark, this means that you are using a trial version of the product. Log in to your account at <http://stimulsoft.com/RegisteredUsers.aspx> and download the Registered version of the product. Also, please read the following topics:

1. [How to upgrade to a new version?](#)
2. [How to upgrade to the prerelease build?](#)
3. [I have installed new version but nothing changed. What can I do?](#)
4. [How to install registered non Demo version of Stimulsoft Reports?](#)

1.3 Information in Web and References

This topic describes how to get the information about the latest news and announcements about the software products, as well as information about known problems and issues that users are interested.

The official site of our company can be found at <http://www.stimulsoft.com> The website has a brief information of the software products <http://www.stimulsoft.com/Products.aspx> they can be downloaded at <http://www.stimulsoft.com/Downloads.aspx> where you can download a trial version of the software, as well as weekly minor prerelease builds and various database adapters.

Live demos for testing the products online can be found at:

Stimulsoft Reports.Net <http://web.stimulsoft.com>

Stimulsoft Reports.Web <http://webfx.stimulsoft.com> (Flash).

Stimulsoft Reports.Web <http://web.stimulsoft.com> (Ajax).

Stimulsoft Reports.Silverlight <http://websl.stimulsoft.com> (Client/Server).

Stimulsoft Reports.Silverlight <http://sl.stimulsoft.com> (Silverlight).

Video tutorials are available at the following link <http://www.stimulsoft.com/Videos.aspx>

Latest company and product news are available at <http://www.stimulsoft.com/AllNews.aspx>

Knowledgebase with answers on critical questions is available at http://stimulsoft.helpserve.com/index.php?_m=knowledgebase&_a=view

You may read an information on weekly prerelease builds and major versions on the Forum at the following link <http://forum.stimulsoft.com/Default.aspx?g=forum&c=2> The status of such topics is

marked as **Announcement** and is always shown above all forum topics. Also on the forum, you may read and discuss various subjects regarding reporting tools.

Short messages on the latest news of our company can be read on Twitter <http://twitter.com/Stimulsoft>, Facebook <http://www.facebook.com/pages/Stimulsoft/166319983418027> and RSS http://www.stimulsoft.com/rss/stimulsoft_rss_en.xml

You may subscribe to the newsletters on the [Home](#) page of our website.

For getting more information about the product in other online resources, please use the search engines.

2 Data

2.1 Business Objects in Net, Web, Silverlight

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

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:

```
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:

```

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";

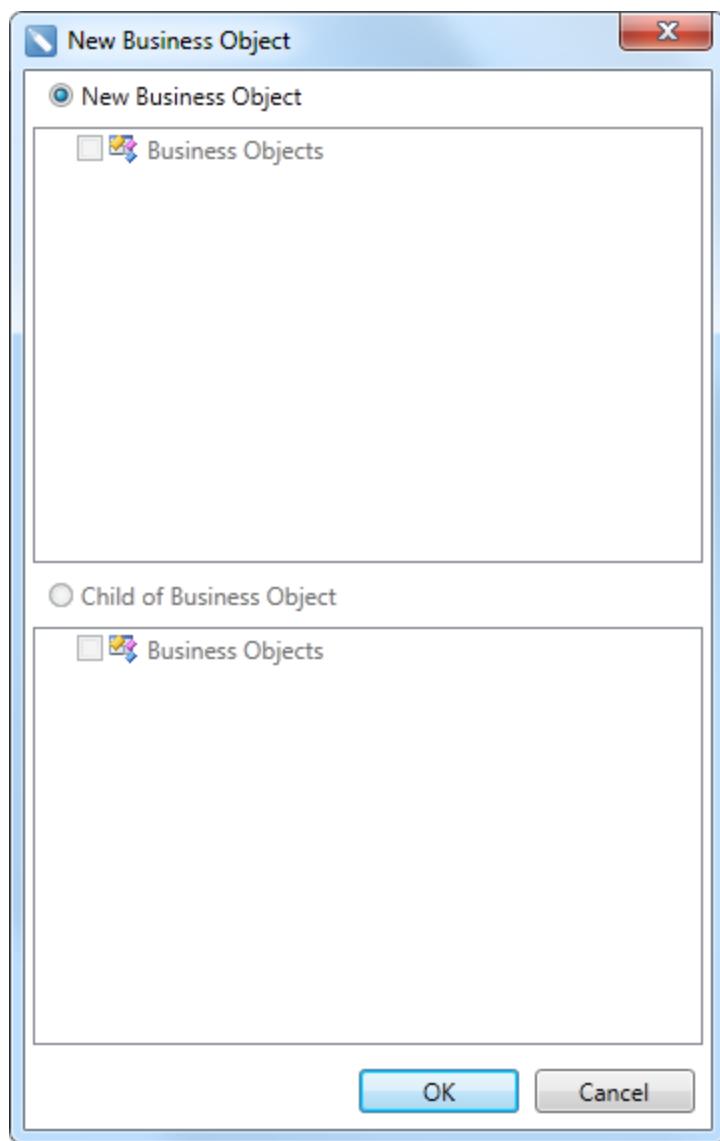
MyObject.Category c2 = new MyObject.Category();
c2.number = 2;
c2.name = "Cat2";
c2.description = "desc for n2";

obj.list[0] = c1;
obj.list[1] = c2;

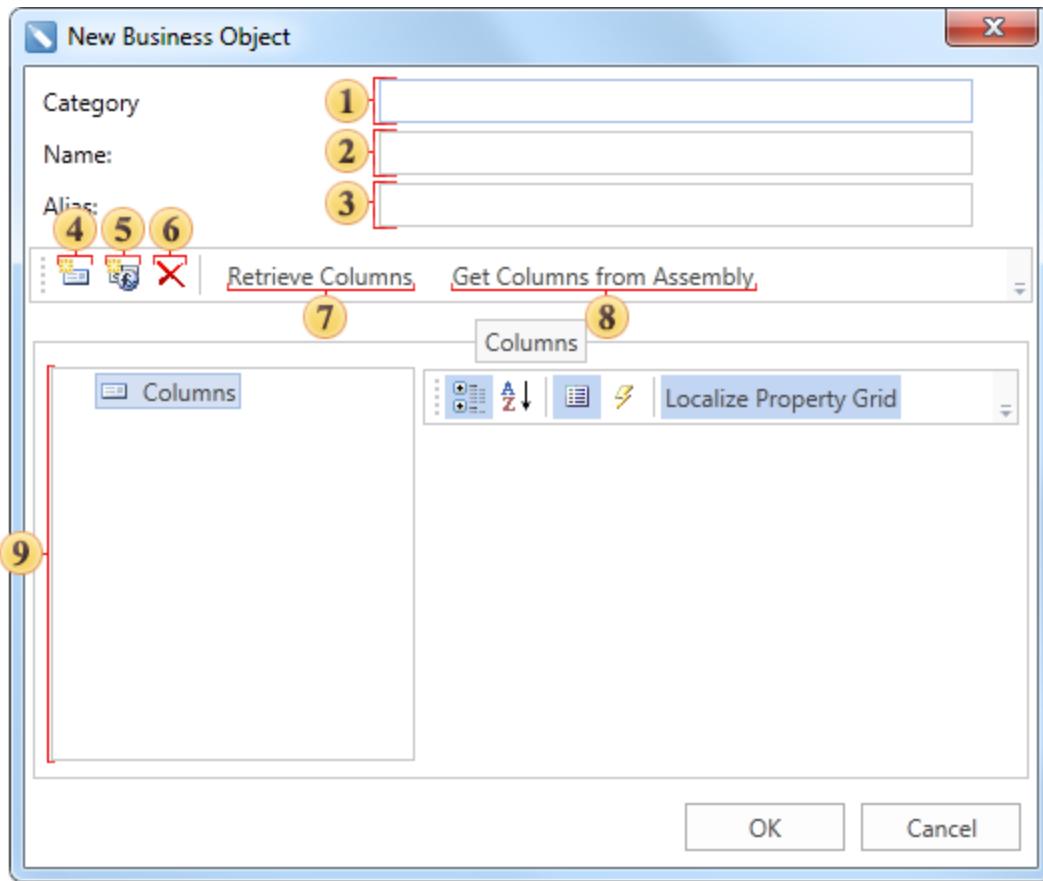
StiReport mainreport = new StiReport();
mainreport.RegBusinessObject("MyObject", obj);
mainreport.Design();

```

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**.



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**.



- 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.
- 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:

```
public class MyObject
{
    public class Category
    {
        public int categoryID;
        public int CategoryID
        {
            get
            {
                return categoryID;
            }
        }

        public string categoryName;
        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:

```
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:



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:

```
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:

The screenshot shows a rendered report with a table titled "Categories". The table has three columns: "CategoryID", "CategoryName", and "Description". The data is as follows:

CategoryID	CategoryName	Description
3	Confections	Desserts, candies, and sweetbreads
4	Dairy Products	Cheeses
5	Grains/Cereals	Breads, crackers, pasta, and cereals
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:

```

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);

```

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:

```

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:

```

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;
}

```

Business objects in Silverlight

In Silverlight creating a class of the business object is identical to the creation of a Business Object in .NET. In the Silverlight designer, as well as in the .NET one, it is possible to create a description of a business object from the data dictionary. Getting data for business objects is the same as in .NET. The difference is that, instead of the **report.Load("*.mrt")**, you should use a different code. Below is a sample code:

```

SaveFileDialog sf = new SaveFileDialog();
sf.Filter = "Files report (*.mrt)|*.mrt";

if (sf.ShowDialog() == true)
{
    Stream file = sf.OpenFile();
    report.Load(file);
}

```

2.2 Business Objects in WinRT

In order to pass the business objects in WinRT, you should use the following methods:

The method of saving a dictionary structure

The method of saving the structure of the dictionary file ***.dct**, for further opening it in the report designer and creating a report. In this case, only the structure of the **Dictionary** is remained. The structure contains a description of business objects. Here is the code that implements this method:

```

var picker = new Windows.Storage.Pickers.FileSavePicker();
picker.FileTypeChoices.Add("Files Dictionary (*.dct)", new
System.Collections.Generic.List<string>() { ".dct" });
picker.SuggestedFileName = "ReportDictionary1";
picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.ComputerFolder;

var storageFile = await picker.PickSaveFileAsync();
if (storageFile != null)
{
    StiReport report = new StiReport();
    report.RegBusinessObject("Categories", "Categories", GetData());
    report.Dictionary.SynchronizeBusinessObjects(3);
    await report.Dictionary.SaveAsync(storageFile);
}

```

The method of saving a report

The method of saving to a file ***.mrt**, with the structure of the report dictionary. The structure of the dictionary includes a description of the business object. Here is the code that implements this method:

```

var picker = new Windows.Storage.Pickers.FileSavePicker();
picker.FileTypeChoices.Add("Files report (*.mrt)", new
System.Collections.Generic.List<string>() { ".mrt" });
picker.SuggestedFileName = "Report1";
picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.ComputerFolder;

var storageFile = await picker.PickSaveFileAsync();
if (storageFile != null)

```

```
{
    StiReport report = new StiReport();
    report.RegBusinessObject("Categories", "Categories", GetData());
    report.Dictionary.SynchronizeBusinessObjects(3);
    await report.SaveAsync(storageFile);
}
```

Next, consider creating a report template with the description of the business objects, filling them with real data and reporting.

Creating a report template with a description of the business object

To do this, open a saved report with the structure of the dictionary or open the dictionary data in the report designer. Next, using the description, you should create a report template. For example, dragging the business object to the page. When dragging the dialogue form **Data** will be invoked, which determines the field references of the business object, the basis of the report - **Data Band** or **Table**, as well as to add a **Header Band** and **Footer Band** to the report template. You should also edit report components. The picture below shows the created report template:



The picture above shows that the report template is created. Since it was created with a description of the business object that does not contain the actual data, the report can not be rendered. For rendering a report, a business object should be filled with real data. This can be done manually by specifying values for the fields, or to connect the data source from which the data will be delivered. Created report template should be saved, for example, in the folder "**My Documents**" with the name **Report.mrt**.

Filling the business object with the real data

Filling the business object in this example, will be done from the installed database. First, we need to create a connection to this database in **Visual Studio**. After this, you should specify the filling code of the business object. Filling the actual business object data directly before the report. Here is the code to fill the business object:

Xaml:

```
<Page>
    <viewerRT:StiViewerControl x:Name="viewerControl" />
</Page>
```

C#:

```
StiReport report = new StiReport();
```

```

StorageFile file = await KnownFolders.PicturesLibrary.GetFileAsync("");
await report.LoadAsync(file);

using (NorthwindDataContext context = new NorthwindDataContext())
{
    var categories =
        from c in context.Categories
        select new { c.CategoryID, c.CategoryName, c.Description };

    report.RegBusinessObject("Categories", categories);
    await report.RenderAsync();

    viewerControl.Report = report;
}

```

After that, the report generator fills the business object with data from the specified data source, in this case from the database **Northwind**, the table **Categories**. Then, the report will be rendered by the existing template. The picture below shows the rendered report:

CategoryID	CategoryName	Description
1	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
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

2.3 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:

1. Connect the **Stimulsoft** assemblies;

2. 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>;
3. Use following code:

```
//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();

//Transfferring Data to Report via Business Objects
var report = var StiReport();
report.RegBusinessObject("Products", product);
report.Dictionary.SynchronizeBusinessObjects(2);
report.Design();
```

3 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.

3.1 How to Show Report?

Just call only one method to show a report:

C#

```
StiReport report = new StiReport();
report.Load("report.mrt");
report.Show();
```

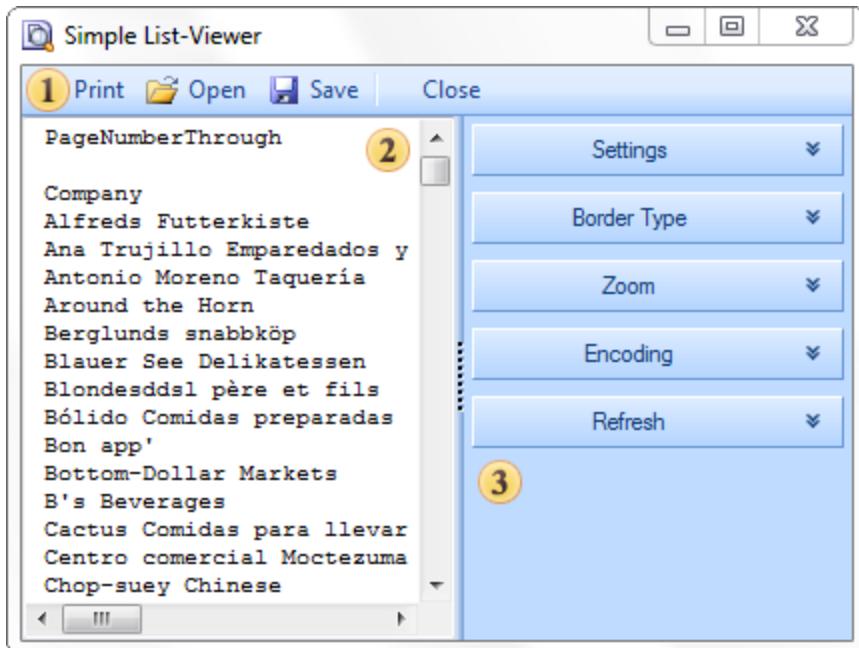
VB.NET

```
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.

3.2 Dot-Matrix Viewer for WinForms

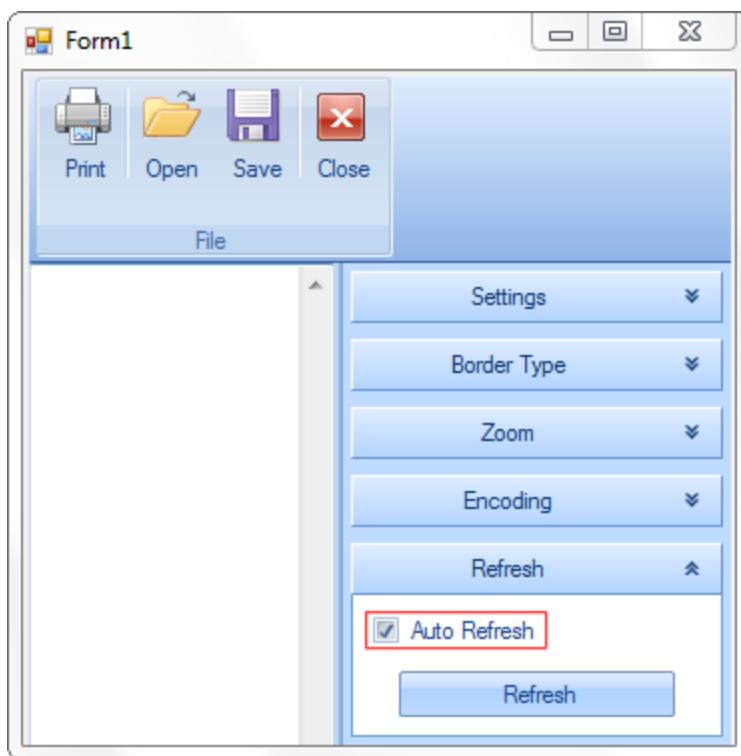
The **Dot-Matrix** viewer is designed to preview the report before printing it on dot matrix printer. The Dot matrix printer is used to print only the text and characters of pseudographics. Accordingly the viewer displays only the text and borders of objects as pseudographics characters. The picture below shows the Dot-matrix viewer dialog box:



- 1 The **Dot-matrix viewer toolbar**.
- 2 The panel displays the text of a report
- 3 The options bar of a report.

3.2.1 Setting Dot-Matrix Viewer in WinForms

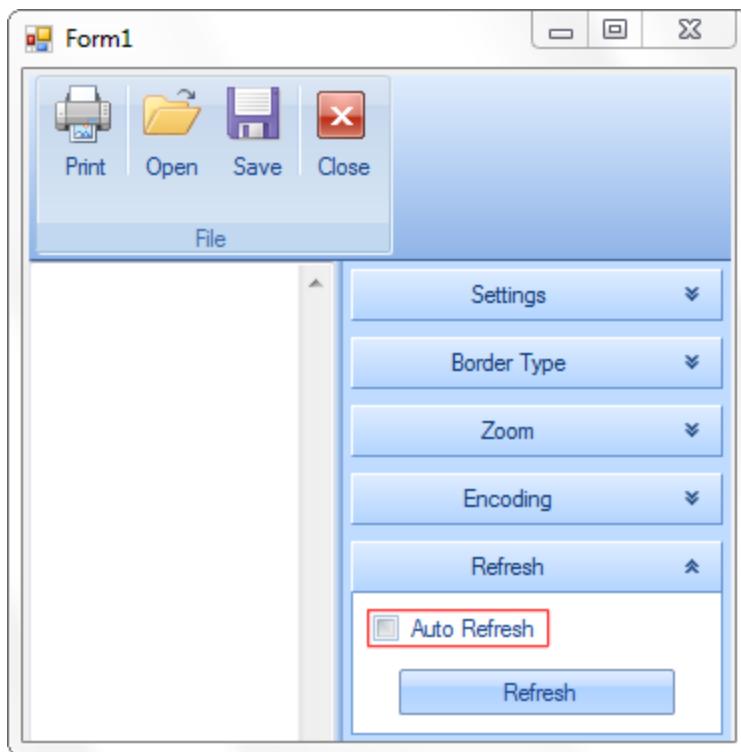
The **Dot-Matrix** viewer can be configured from code using static properties. Depending on the value of the static properties in the Dot-matrix viewer, these or that parameters will be specified. For example, the **AutoRefresh** property. The picture below shows the **Dot-Matrix** viewer dialog box:



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:

```
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.

3.2.2 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.

Command/Collection	EpsonFX	Oki ML92/93
b	ESC E	ESC T

Command/Collection	EpsonFX	Oki ML92/93
/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 or 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.

4 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.

4.1 How to Show Report?

Just call one method to show a report:

C#

```
StiReport report = new StiReport();
report.Load("report.mrt");
```

```
report.ShowWithWpf();
```

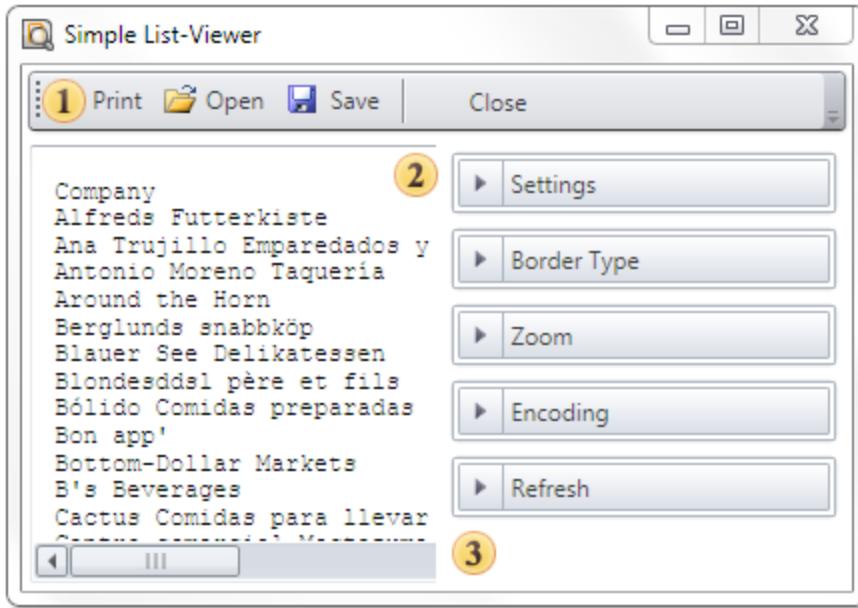
VB.NET

```
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.

4.2 Dot-Matrix Viewer for WPF

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:

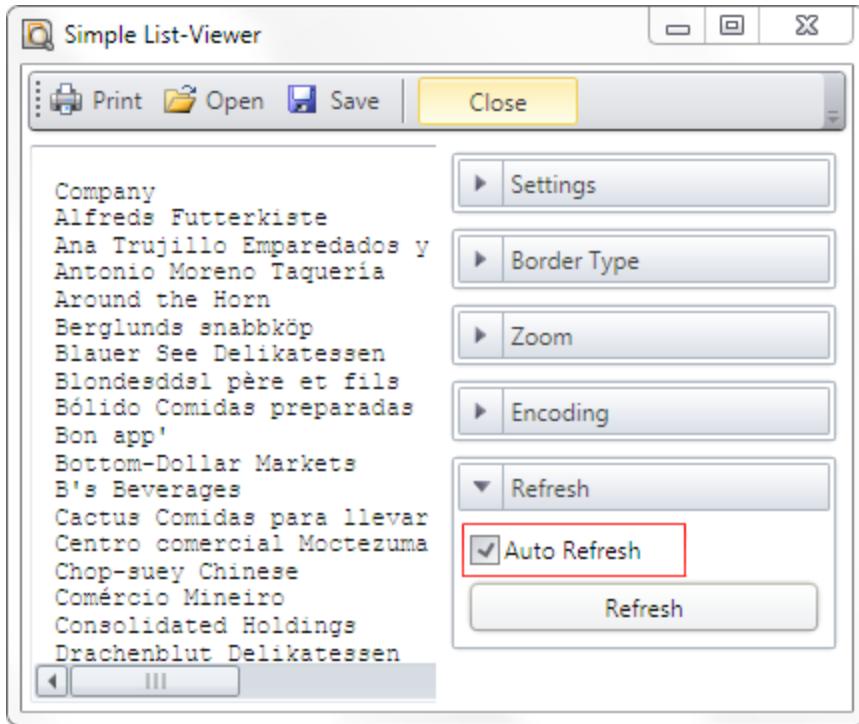


- ① The **Dot-Matrix viewer toolbar**.
- ② The panel displays the text of a report
- ③ The options bar of a report.

4.2.1 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,

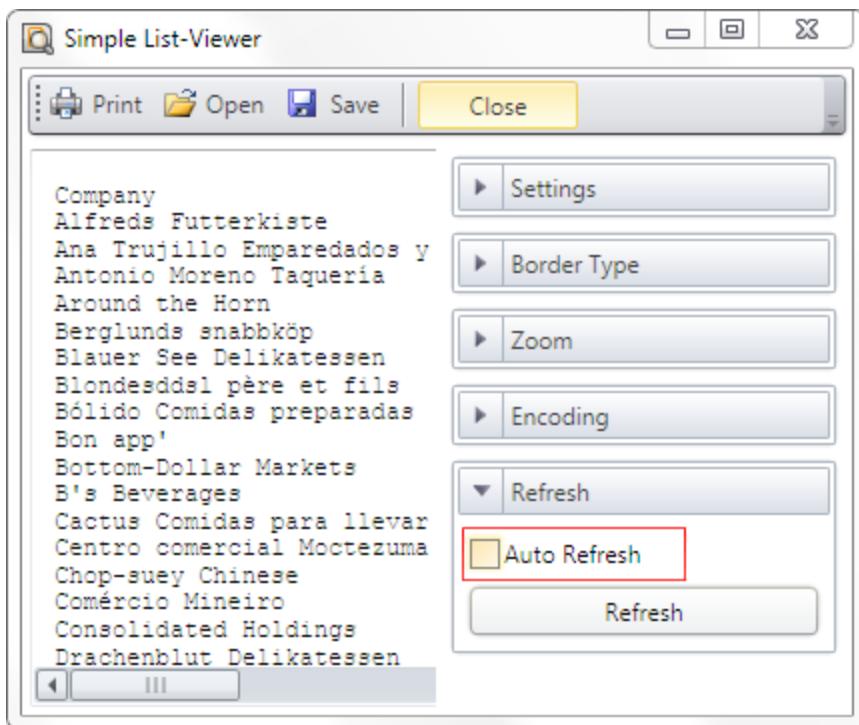
the **AutoRefresh** property. The picture below shows the **Dot-matrix** viewer dialog box:



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:

```
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.

4.2.2 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
```

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	

Command/Collection	EpsonFX	Oki ML92/93
/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 or 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.

5 Web Designer

The **StiWebDesigner** component is used to edit reports in the window of a browser. And there is no need to install the **.NET Framework**, **ActiveX** components and other special plug-ins on the client machine. The only requirements are a web browser and the **Flash** player. Using **StiWebDesigner** it is possible to create, edit, save, view, and print reports on any computer, on any OS, where there is an internet connection, and where there is the installed web browser and **Flash Player 11.1** and higher. **StiWebDesigner** is non visual **ASP.NET** component. It can be divided in two parts: client and server. The client side is the graphic wrapping of the designer that is realized on Flex. The server side is the report generator engine and, also, a module that has functions to get queries and send data on the client side. These two parts are collected into one **DLL** library and represented as a non visual component.

5.1 How It Works?

To run the web report designer, it is required to put the StiWebDesigner component on the **ASP.NET** page and call the Design method of this component. When running the Web report designer the

following actions occur:

- » The **.NET** component reads to the memory the client Flash application from resources and runs it.
- » When it is loaded, the client side, in the **AJAX** mode, requests from the server side all necessary settings and the report file. The server side passes all this.
- » When saving a report in the preview mode, the client side sends the report file as **XML** in the **AJAX** mode and the server side makes the premature processing of the report, and either sends it for saving or compile it in the window of a browser.

5.2 How to Run Web Report Designer?

For running the Web report designer it is necessary to put non visual StiWebDesigner component on the form and, in the event handler of a control, to call the Design() method:

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

C#:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiWebDesigner1.Design();
}
```

VB.NET:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
EventArgs)
    StiWebDesigner1.Design()
End Sub
```

For loading a report in the Web designer, the method of calling can be slightly modified:

C#:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\\\SimpleList.mrt");
    StiWebDesigner1.Design(report);
}
```

VB.NET:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
EventArgs)
    Dim report As StiReport = New StiReport()
    report.Load("D:\\\\SimpleList.mrt")
```

```
    StiWebDesigner1.Design(report)
End Sub
```

It requires a bit more complicated code to call the report designer automatically when loading a page. It is necessary to exclude service messages which are sent by the client part of the designer to the server part:

C#:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Page != null)
    {
        string keyValue =
        Page.Request.QueryString.Get("stimulsoft_webdesigner");

        if (!IsPostBack && keyValue == null)
        {
            StiReport report = new StiReport();
            report.Load("D:\\SimpleList.mrt");
            StiWebDesigner1.Design(report);
        }
    }
}
```

VB.NET:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
    If Not Page Is Nothing Then
        String keyValue =
        Page.Request.QueryString.Get("stimulsoft_webdesigner")

        If Not IsPostBack And keyValue Is Nothing Then
            Dim report As StiReport = New StiReport()
            report.Load("D:\\SimpleList.mrt")
            StiWebDesigner1.Design(report)
        End If
    End If
End Sub
```

5.3 Loading Reports to Web Designer

One of the following methods can be used to load a report to the **Web designer**:

- ✓ Loading a report before loading the designer;
- ✓ Loading a report after loading the designer;
- ✓ Loading a report from the main menu of the designer.

► **Loading a report before loading the designer.** In this way the report (for example, from a file) is loaded first and then the designer is loaded. The previously loaded report is specified as a parameter of a method of calling the designer. A code below is a sample for loading a report before loading the designer:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    StiWebDesigner1.Design(report);
}
```

or, as a way, the previously loaded report is assigned to the designer. In this case designer loading is done with this report. See the code below:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    StiWebDesigner1.Report = report;
    StiWebDesigner1.Design();
}
```

► **Loading a report after loading the designer** is done using the **GetReport** event. After adding the handler to this event, it will occur each time when a report is required for the designer. In other words, after loading the **Web designer** requests a report from the server and, if the handler is added to the **GetReport** event, then in this event a report can be assigned to the designer. See the code below how to use the **GetReport** event:

```
protected void StiWebDesigner1_GetReport(object sender,
StiWebDesigner.StiGetReportEventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    e.Report = report;
}
```

► If the designer is set to the **Visual** mode, then the report should be loaded with way specified below:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    StiWebDesigner1.Report = report;
}
```

5.4 Report Preview

The preview function of the edited report, in the **Web designer**, has two modes: **HTML** (preview as HTML file) and **PDF** (preview as PDF file). After selecting one of two modes, a report will be shown in the browser window. If, when report rendering, the errors occur, then alert messages will be shown in the separate browser window. To preview a report in the web designer window can be done by switching the **Preview** tab in the designer. To preview the report data are required. By default, data are taken from the **Dictionary** of the edited report. If necessary, they can be overridden. There is a sample code below using what data can be overridden:

C#:

```
protected void StiWebDesigner1_GetPreviewDataSet(object sender,
StiWebDesigner.StiPreviewDataSetEventArgs e)
{
    DataSet data = new DataSet();
    data.ReadXml("D:\\Demo.xml");
    data.ReadXmlSchema("D:\\Demo.xsd");
    e.PreviewDataSet = data;
}
```

VB.NET:

```
Protected Sub StiWebDesigner1_GetPreviewDataSet(ByVal sender As Object,
ByVal e As StiWebDesigner.StiPreviewDataSetEventArgs)
    Dim data As DataSet = New DataSet()
    data.ReadXml("D:\\Demo.xml")
    data.ReadXmlSchema("D:\\Demo.xsd")
    e.PreviewDataSet = data
End Sub
```

As seen from code, data are taken from **XML** and **XSD** files. The same way exists for other data sources. If any connections to the database are specified in the report, they have a higher priority than the data specified in the **GetPreviewDataSet** event. In order specified data be applied when rendering the report, you should clear the list of connections:

```
protected void StiWebDesigner1_GetPreviewDataSet(object
sender,StiWebDesigner.StiPreviewDataSetEventArgs e)
{
    e.Report.Dictionary.Databases.Clear();

    DataSet data = new DataSet();
    data.ReadXml("D:\\Demo.xml");
    data.ReadXmlSchema("D:\\Demo.xsd");
    e.PreviewDataSet = data;
}
```

5.5 Changing Report Settings Before Rendering

If it is necessary before the report rendering to change the report parameters or user settings in the report, you can add a handler to the **ProcessReportBeforeRender** event. This event occurs immediately before the report rendering. For example, it is necessary to change the **Report Name** before report rendering. To do this, add the handler to the **ProcessReportBeforeRender** event, define the value of the **ReportName** property, where the value of the property is the name of the report. For example, add the following code to the code project:

```
protected void StiWebDesigner1_ProcessReportBeforeRender(object sender,
StiWebDesigner.StiProcessReportBeforeRenderEventArgs e)
{
    e.Report.ReportName = "Employees";
}
```

Now, before report rendering, the name will be changed from the existing one to **Report 2010**. Thus, using the **ProcessReportBeforeRender** event, the user can change custom options before rendering.

5.6 Saving Reports

Two events are used for the processing of a saved report in the Web-designer. These are **SaveReport** and **SaveReportAs**. The **SaveReport** event occurs when you click on the Save Report button or when you click on the Save Report item of the main menu in the designer. The **SaveReportAs** event occurs when you select the Save As item of the main menu in the designer. When the user is not signed to this event, then, by default, a built-in Web-designer dialog box to save the report is called. If the user is assigned to this event, then the dialog box will not appear but the event will appear, i.e. saving is done by the server. See the code below:

C#:

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

VB.NET:

```
Protected Sub StiWebDesigner1_SaveReport(ByVal sender As Object, ByVal e As StiWebDesigner.StiSaveReportEventArgs)
    Dim report As StiReport = e.Report
End Sub
```

After that, the ways and place to save the report are identified by the user. For example, the report can be saved to a string in the database. Below is the code to save a report to the string:

C#:

```
protected void StiWebDesigner1_SaveReport(object sender,
StiWebDesigner.StiSaveReportEventArgs e)
{
    StiReport report = e.Report;
    string str = report.SaveToString();
}
```

VB.NET:

```
Protected Sub StiWebDesigner1_SaveReport(ByVal sender As Object, ByVal e As StiWebDesigner.StiSaveReportEventArgs)
    Dim report As StiReport = e.Report
    Dim str As String = report.SaveToString()
End Sub
```

Subscription to the **SaveReportAs** event is made by analogy with the **SaveReport** event. In the event of saving a report you can use the **ErrorCode** property of the event argument. The **ErrorCode** property can take numeric values. By default, this property is set to -1. This means that, when saving a report, the report event occurs, but nothing will be output in the Web-designer. If this property is set to 0, then the user will be notified about that the report is saved successfully. The picture below shows a notification window which indicates successful report saving:



If the **ErrorCode** property will be greater than 0, then the error message is displayed with its source code where the error is the value of the **ErrorCode**. The picture below shows a window with the error code 1:



Report saving occurs in the background mode, i.e. visually it will not be displayed. If you need to visually manage the process of saving the report, you may change the **SaveMode** property of the **StiWebDesigner** component on one of two values: **Visible** or **NewWindow**:

```
<cc1:StiWebDesigner ID="StiWebDesigner1" runat="server"
```

```
OnSaveReport="StiWebDesigner1_SaveReport" SaveMode="NewWindow" />
```

If the **SaveMode** property is set to **Visible**, the process of saving the report will be displayed in the current browser window. If the property is set to **NewWindow**, then the process of saving the report will be displayed in a new browser window. By default, this property is set to **Hidden**, i.e. the process of saving the report is not displayed.

5.7 Web Designer Settings

Setting the **Web designer** can be done using the static properties, which are described in the **Stimulsoft.Report.Web.StiWebDesignerOptions** class. Static properties of the **Web designer** can be divided into following groups: Connection, Main menu, Zooming, Viewer, Additional.

5.7.1 Connection

The static properties described below belong to the **StiWebDesignerOptions.Connection** group and responsible for option of connection the client and server sides:

Name	Description
ClientRequestTimeout	property sets time (in seconds) that the client part will wait the response from the server side. The default value is 10 seconds.
ClientRepeatCount	property sets the number of repeats of requests of the server side to the client side, when getting errors of obtaining data. The default value is 2 repeats.
RelativeUrls	property allows using the relative Url . If the RelativeUrls is set to false , then the absolute Url is used. If the RelativeUrls is set to true , then the relative Url is used. By default, the value is set to false . A sample of the absolute and relative Urls is shown below: http://localhost:4444/WebDesignerDemo/WebDesigner.aspx is an absolute Url , the RelativeUrls property is set to false ; /WebDesignerDemo/WebDesigner.aspx is a relative Url , the RelativeUrls property is set to true .

5.7.2 Main Menu

The main menu of the **Web** designer can be setup according to user's requirements. This group of static properties **StiWebDesignerOptions.Menu** allows enabling/disabling main menu or submenu items.

Name	Description
------	-------------

NewEnabled	property is used to enable/disable the New menu item. If the NewEnabled property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true . The New menu contains submenus such as: New Report , New Report With Wizard , New Page .
NewReport	property is used to enable/disable the New Report submenu item. If the NewReport property is set to true , then this submenu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
NewReportWithWizard	property is used to enable/disable the New Report With Wizard submenu item. If the NewReportWithWizard property is set to true , then this submenu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
NewPage	property is used to enable/disable the New Page submenu item. If the NewPage property is set to true , then this submenu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
OpenReport	property is used to enable/disable the Open Report menu item. If the OpenReport property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
SaveReport	property is used to enable/disable the Save Report menu item. If the SaveReport property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
SaveAs	property is used to enable/disable the Save As menu item. If the SaveAs property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
DeletePage	property is used to enable/disable the Delete Page menu item. If the DeletePage property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true ;
Preview	property is used to enable/disable the Preview menu item. If the Preview property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true . The Preview

	menu item contains a submenu which contains the following items: Preview As Pdf , Preview As Html .
PreviewAsPdf	property is used to enable/disable the Preview As Pdf submenu item. If the PreviewAsPdf property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true .
PreviewAsHtml	property is used to enable/disable the Preview As Html submenu item. If the PreviewAsHtml property is set to true , then this menu item is enabled and available for a user. If this property is set to false then this submenu item is disabled and not available for a user. By default, this property is set to true .

5.7.3 Zooming Static Properties

The **Zoom** group of static properties has one static property: **StiZoomMode**. Depending on the values of this property it is possible to set options of report template zoom. This property has the following values: **PageWidth**, **PageHeight**, **OnePage**, **Zoom25**, **Zoom50**, **Zoom75**, **Zoom100**, **Zoom150**, **Zoom200**.

Name	Description
PageWidth	Sets zoom by Page Width . So the width of the report template matches the width of the window of the web designer.
PageHeight	Sets zoom by Page Height . So the height of the report template matches the height of the window of the web designer.
OnePage	Sets zoom by One Page . So the entire page of the report template fits in the window of the web designer.
Zoom25 , Zoom50 , Zoom75 , Zoom100 , Zoom150 , Zoom200	Set zoom level of the report template which is 25% , 50% , 75% , 100% , 150% , 200% .

5.7.4 Viewer Static Properties

A group of **StiWebDesignerOptions.Viewer.Toolbar** static properties is described below:

Name	Description
ShowZoom	Property is used to show/hide the zoom panel. If the ShowZoom property is set to true , then the zoom panel will be shown. If the ShowZoom property is set to false , then the zoom panel will be hidden. By default this property is set to true .

ShowPrintButton	Property is used to show/hide the Print button. If the ShowPrintButton property is set to true , then the Print button is shown. If the ShowPrintButton property is set to true , then the Print button is hidden. By default this property is set to true .
ShowOpenButton	Property is used to show/hide the Open button. If the ShowOpenButton property is set to true , then the Open button is shown. If the ShowOpenButton property is set to true , then the Open button is hidden. By default this property is set to true .
ShowSaveButton	Property is used to show/hide the Save button. If the ShowSaveButton property is set to true , then the Save button is shown. If the ShowSaveButton property is set to true , then the Save button is hidden. By default this property is set to true .
ShowSendEmailButton	Property is used to show/hide the SendEmail button. If the ShowSendEmailButton property is set to true , then the SendEmail button is shown. If the ShowSendEmailButton property is set to true , then the SendEmail button is hidden. By default this property is set to true .
ShowPageNewButton	Property is used to show/hide the Page New button. If the ShowPageNewButton property is set to true , then the Page New button is shown. If the ShowPageNewButton property is set to true , then the Page New button is hidden. By default this property is set to true .
ShowPageDeleteButton	Property is used to show/hide the Page Delete button. If the ShowPageDeleteButton property is set to true , then the Page Delete button is shown. If the ShowPageDeleteButton property is set to true , then the Page Delete button is hidden. By default this property is set to true .
ShowPageSizeButton	Property is used to show/hide the Page Size button. If the ShowPageSizeButton property is set to true , then the Page Size button is shown. If the ShowPageSizeButton property is set to true , then the Page Size button is hidden. By default this property is set to true .
ShowBookmarksButton	Property is used to show/hide the Bookmarks button. If the ShowBookmarksButton property is set to true , then the Bookmarks button is shown. If the ShowBookmarksButton property is set to true , then the Bookmarks button is hidden. By default this property is set to true .
ShowThumbnailsButton	Property is used to show/hide the Thumbnails button. If the ShowThumbnailsButton property is set to true , then the Thumbnails button is shown. If the ShowThumbnailsButton property is set to true , then the Thumbnails button is hidden. By default this property is set to true .
ShowFindButton	Property is used to show/hide the Find button. If the ShowFindButton property is set to true , then the Find button is shown. If the ShowFindButton property is set to true , then the Find button is hidden. By default this property is set to true .

ShowEditButton	Property is used to show/hide the Edit button. If the ShowEditButton property is set to true , then the Edit button is shown. If the ShowEditButton property is set to false , then the Edit button is hidden. By default this property is set to true .
ShowFirstPageButton	Property is used to show/hide the First Page button. If the ShowFirstPageButton property is set to true , then the First Page button is shown. If the ShowFirstPageButton property is set to false , then the First Page button is hidden. By default this property is set to true .
ShowPreviousPageButton	Property is used to show/hide the Previous Page button. If the ShowPreviousPageButton property is set to true , then the Previous Page button is shown. If the ShowPreviousPageButton property is set to false , then the Previous Page button is hidden. By default this property is set to true .
ShowGoToPageButton	Property is used to show/hide the Go to Page button. If the ShowGoToPageButton property is set to true , then the Go to Page button is shown. If the ShowGoToPageButton property is set to false , then the Go to Page button is hidden. By default this property is set to true .
ShowNextPageButton	Property is used to show/hide the Next Page button. If the ShowNextPageButton property is set to true , then the Next Page button is shown. If the ShowNextPageButton property is set to false , then the Next Page button is hidden. By default this property is set to true .
ShowLastPageButton	Property is used to show/hide the Last Page button. If the ShowLastPageButton property is set to true , then the Last Page button is shown. If the ShowLastPageButton property is set to false , then the Last Page button is hidden. By default this property is set to true .
ShowPageViewModeSingleButton	Property is used to show/hide the Single Page button. If the ShowPageViewModeSingleButton property is set to true , then the Single Page button is shown. If the ShowPageViewModeSingleButton property is set to false , then the Single Page button is hidden. By default this property is set to true .
ShowPageViewModeContinuousButton	Property is used to show/hide the Continuous button. If the ShowPageViewModeContinuousButton property is set to true , then the Continuous button is shown. If the ShowPageViewModeContinuousButton property is set to false , then the Continuous button is hidden. By default this property is set to true .
ShowPageViewModeMultipleButton	Property is used to show/hide the Multiple Pages button. If the ShowPageViewModeMultipleButton property is set to true , then the Multiple Pages button is shown. If the ShowPageViewModeMultipleButton property is set to false , then the Multiple Pages button is hidden. By default this property is set to true .

5.7.5 Additional Viewer Static Properties

This group of **StiWebDesignerOptions** static properties allows enabling/disabling dictionary and report template editing, showing/hiding tabs of the **Web designer**.

Name	Description
ModifyDictionary	Property is used to enable/disable Dictionary editing. If the ModifyDictionary property is set to true , then editing is enabled. If the ModifyDictionary property is set to false , then editing is disabled. By default this property is set to true .
ModifyConnections	Property is used to enable/disable Connections editing. If the ModifyDictionary property is set to true , then editing is enabled. If the ModifyDictionary property is set to false , then editing is disabled. By default this property is set to true .
ModifyDataSources	Property is used to enable/disable Data Sources editing. If the ModifyDataSources property is set to true , then editing is enabled. If the ModifyDataSources property is set to false , then editing is disabled. By default this property is set to true .
ModifyVariables	Property is used to enable/disable Variables editing. If the ModifyVariables property is set to true , then editing is enabled. If the ModifyVariables property is set to false , then editing is disabled. By default this property is set to true .
ModifyTemplate	Property is used to enable/disable loaded Template editing. If the ModifyTemplate property is set to true , then editing is enabled. If the ModifyTemplate property is set to false , then editing is disabled. By default this property is set to true .
AllowScale	Property is used to enable/disable scale changing. If the AllowScale property is set to true , then changing is enabled. If the AllowScale property is set to false , then changing is disabled. By default this property is set to true .
CodeTabVisible	Property is used to show/hide the Code tab. If the CodeTabVisible property is set to true , then the code tab will be shown. If the CodeTabVisible property is set to false , then the code tab will be hidden. By default this property is set to true .
DictionaryTabVisible	Property is used to show/hide the Dictionary tab. If the DictionaryTabVisible property is set to true , then the code tab will be shown. If the DictionaryTabVisible property is set to false , then the code tab will be hidden. By default this property is set to true .
ExitButtonVisible	Property is used to show/hide the Exit button in the main menu. If the ExitButtonVisible property is set to true , then the button tab will be shown. If the ExitButtonVisible property is set to false , then the button tab will be hidden. By default this property is set to true .

5.8 Web Designer Properties

Web designer properties are described below:

Name	Description
BrowserTitle	property is used to change titles of a browser. This property may get string values. By default, the title of a browser is Report Alias , and if it is absent, it is a Report Name .
SaveMode	If you need to show the process of saving a report, you need to set the SaveMode property of the StiWebDesigner component to one of the following values: Visible or NewWindow . If the SaveMode property is set to Visible , then process will be displayed in the current window of the designer. If the SaveMode property is set to NewWindow , then the saving process will be shown in a new window of the browser. By default this property is set to Hidden , the process of saving is not shown.
SaveAsMode	The difference of this property is that, it works when the SaveReportAs event occurs, and the SaveMode property when the SaveReport event occurs.
DataEncryption	property is used to enable/disable data encryption. If the DataEncryption property is set to false , then data are not decrypted. If the DataEncryption property is set to true , then data are decrypted. By default, this property is set to false .
DataCompression	property is used to enable/disable data compression. If the DataCompression property is set to false , then data are not compressed. If the DataCompression property is set to true , then data are compressed. By default, this property is set to true .
UseCache	property allows using caching on the server when loading a report. If the UseCache property is set to true , then caching is used when loading a report, i.e. a report is loaded to the Web designer from the server cache. If the UseCache property is set to false , then caching is not used when loading a report. In this case for loading the report in the Web designer the GetReport event should be used. By default, this property is set to true .
ServerTimeout	property is used to define the time of a report in the report cache. By default, this property is set to " 00:10:00 ", this means that the report is stored 10 minutes in the server cache and then it is removed.
ExitUrl	property is used to assign Url , to what user will be redirected when closing the Web designer via clicking the Exit button of the main menu. By default, a user will be redirected on a page from what the Web designer is run.
ShowWizardOnStartup	property is used to show the report rendering wizard window when running the Web designer . If the ShowWizardOnStartup property is set to false , then, when running the Web designer , the report rendering wizard window will not be shown. If the ShowWizardOnStartup property is set to true , then, when running the Web designer , the report rendering wizard window will be shown. By default, this property is set to false .

AppCacheDirectory	property is used to define the path to the directory on the server, in what, caching files of the Flash -application will occur. For this, you should set full access of the ASP.NET application to this directory.
ShowSelectLanguage	Shows/hides the button to invoke the menu of selecting the localization. If this property is set to true , the button is displayed. If the property is set to false , the menu button is hidden. By default, this property is set to true .

5.9 Changing Web Designer Properties From Code

The **PreInit** event is used to change **Web designer** properties. This event occurs before initialization of the designer, i.e. before passing Web designer properties to the client part of an application. In other words, to change the Web designer properties from code it is necessary to add the handler to the **PreInit** event. The code below shows how to add the handler to the PreInit event and set the Localization property to en:

```
protected void StiWebDesignerSL1_PreInit(object sender,
    StiWebDesignerSL.StiPreInitEventArgs e)
{
    e.WebDesignerSL.Localization = "en";
}
```

Now, when running the **Web designer**, it will be localized in English. For example, we need to change the browser title. By default, the browser title is the value of the **Report Alias** property. If the value is not set then the **Report Name** is taken. To change the browser title it is necessary to add the code below to the event:

```
protected void StiWebDesignerSL1_PreInit(object sender,
    StiWebDesignerSL.StiPreInitEventArgs e)
{
    e.WebDesignerSL.BrowserTitle = "Stimulsoft";
}
```

Now, when running the **Web designer, Stimulsoft** word will be shown as a report title.

5.10 Web Report Designer Localization

The Web report designer interface can be localized in any of 24 available languages. It is necessary to create the Localization folder in the root catalogue of your Web-project and copy all necessary .xml localization files. After loading the web report designer, the list of languages will be available in its menu. The path to the **Localization** folder can be changed. The DirectoryLocalization property is used for this:

```
<ccl:StiWebDesigner ID="StiWebDesigner1" runat="server"
    DirectoryLocalization="\\Files\\Languages\\" />
```

Also, it is possible to specify which language can be used right after the web report designer (default localization). The following code shows how to do this. (it is necessary to specify the .xml file in the **Localization** property):

```
<ccl:StiWebDesigner ID="StiWebDesigner1" runat="server"
Localization="ru" />
```

5.11 Visible Mode

When adding a **StiWebDesigner** component on a page of a **Web** applications, it will be added as a non-visual component and calling the designer is done using the **Design ()** method. If you want to place the Web designer in a specific location on a web page with a certain size, you should use the visual designer mode. Designer mode depends on the value of the **Visible** property of the **StiWebDesigner** component. By default, this property is set to **false**, so the designer will be a non-visual component. If the **Visible** property set to **true**, it the visual designer mode will be enabled.

```
<ccl:StiWebDesigner ID="StiWebDesigner1" runat="server" Visible="True" /
>
```

If the designer is in visual mode, then the report should be loaded with the following method:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\\\SimpleList.mrt");
    StiWebDesigner1.Report = report;
}
```

If to call the **Design ()** method in the visual mode of the designer, then loaded report is displayed on the full screen in the browser.

6 Web Viewer

Stimulsoft Reports contains full set of tools to create reports and show them in Web. Basic tools are: **StiWebViewer** and the **StiReportResponse** class. The **StiWebViewer** is used to show a report on a page of browser. The **StiReportResponse** contains a method to render a report to the specified format.

6.1 Caching

The **StiWebViewer** component can output reports in two modes: 1. Using the caching and 2. Without caching. If the cashing is not used then it is necessary, when every page refreshing, to get data from a report and render a report again. When using caching the rendered report is saved in cache on the

server. The next time when the page is refreshed, the previously rendered report is loaded from cache and its re-rendering is not required. It is important to remember that every report saved in cache takes the server memory and, if there are a lot of queries to reports, it can be a critical factor. Therefore, one should choose either low requirements to the memory but high requirements to performance or high requirements to the memory but low requirements to speed. Caching should not be used if the end user needs a report with actual data when every refreshing. The caching process can be controlled using the RenderMode, CacheMode, and ServerTimeOut properties.

If the caching of a report is not used then the report that was rendered using the last data when page refreshing will be printed but not the report that is shown on the current moment. If it is necessary to get the exact copy of a report form the browser, then it is necessary to use caching.

! Notice. It should be borne in mind that if the report caching is not used, then not the report that is shown currently will be printed, but one that will be rendered using the latest data when refreshing a page. If it is important to get an exact copy of the report from the browser, you should use caching.

6.1.1 RenderMode Property

The **RenderMode** property indicates how and when a report should be rendered. All modes of the **StiWebViewer** component can be divided in two categories: using the caching of a rendered report, and without using caching of the rendered report. The modes without caching

▷ Standard

In this mode, the report should be re-rendered every time when the page is refreshed. In addition, this mode does not use Ajax to display the controls of the **StiWebViewer** and any report refresh with help of controls of the **StiWebViewer** component leads to the page refresh on what the report is output.

▷ RenderOnlyCurrentPage

Very interesting mode of the report output. In this mode, the report is rendered only to the page that is currently displayed in the **StiWebViewer** component. For example, if the report consists of 100 pages (this is a big report to be output in the web), and the current page is the page number 5, the report will be rendered only up to the page number 5. The sever memory is saved in this mode.

▷ Ajax

This mode uses Ajax to output a report and to update the content of the **StiWebViewer** component. For example, if a user goes to the next page of a report then not the whole page of the browser on what the StiWebViewer component is placed will be refreshed but only the next page of a report will be sent to the browser using the post-back query. This increases the convenience of working with the **StiWebViewer** component.

Modes with caching are **UseCache** and **AjaxWithCache**.

▷ UseCache

With each refresh in the StiMobileViewer component page reloading from the server occurs, but the report is not re-rendered but each time is loaded from the cache.

▷ AjaxWithCache

This mode as well as the Ajax mode uses the Ajax technology to output a report and also is used for refreshing operations of the **StiWebViewer** component. However, unlike the Ajax mode, this **AjaxWithCache** mode does not re-render the report after each information request on the server. The report that was earlier saved in cache is used.

6.1.2 CacheMode Property

The **CacheMode** property indicates what cache should be used to store reports, images, and service information. There are two ways:

» **Page**

A page cache will be used.

» **Session**

A session cache will be used.

6.1.3 ServerTimeout Property

The **ServerTimeout** property indicates the amount of time on what it is necessary to save a report, pictures of a report, or other data in the cache. Do not use too much time or too little time. If the time is too large, then the used cache will be overflowed and will be automatically cleared by the server. As a result, the report (or images of a report) will not be cached, and incorrect result of the report output in the **StiWebViewer** component will occur. If the time is too small, then by the time of request to the cache, there some necessary data may not be found. It is recommended to set the time equal to 10 minutes. But the exact time can be found experimentally, considering the parameters of the server, users activity, etc.

6.2 Printing Reports

It is difficult to print a report from the browser. **Stimulsoft Reports** has three methods of printing:

- » Converting a report to the PDF file and passing it to the end-user for printing.
- » Printing a report with preview in the pop-up window.
- » Printing without preview.

The first method is the best way. It allows printing a report more precisely. But it is required to have installed Adobe Acrobat to print a report to the PDF format. Often this requirement is a big disadvantage. When printing reports with preview the report generator creates a new pop-up window. A report in the HTML format is output in this window. The end-user may format this report and print it. In printing report without preview the report generator prints a report without preview. When choosing the method of printing characteristics of each method should be considered.

! Notice: The **StiWebViewer** component cannot control page parameters (page size, page orientation, page margins) when printing using the 2 and 3 method. All parameters are controlled with the browser.

6.3 StiReportResponse Class

A report can be shown without using the **StiWebViewer** component. The special

Stimulsoft.Report.Web.StiReportResponse class is used in this case. This class is a set of methods for saving a rendered report in different formats to the stream of a page. It is possible to assign a lot of input parameters which allows controlling the saving format of a report. For example, the following code can be used to save a report to the PDF format to the stream of a page:

```
StiReport report = new StiReport();
report.Load("MyReport.mrt");
report.Render(false);
Stimulsoft.Report.Web.StiReportResponse.ResponseAsPdf(this, report);
```

In this code the report will be loaded first. Then this report will be rendered. Then the result of the report rendering will be saved to the stream of a page. The following code saves a report to the Excel 2007 format.

```
StiReport report = new StiReport();
report.Load("MyReport.mrt");
report.Render(false);
Stimulsoft.Report.Web.StiReportResponse.ResponseAsExcel2007(this,
report);
```

6.4 Using Dialogs in Report

The **StiWebViewer** component can show reports with dialogs, but some conditions should be fulfilled:

- ▷ The **AjaxWithCache** mode should be used or the **UseCache** in the **RenderMode** property;
- ▷ If data are not received from the server then it is necessary to connect with other data using the **ReportConnect** event of the **StiWebViewer** component and also to disconnect from data using the **ReportDisconnect** event;
- ▷ Besides it is impossible to run a report for rendering in the **Load** event of a page;
- ▷ It is possible to use only one dialog form in a report.

6.5 Images of StiWebViewer Toolbar

By default, images for the toolbar of the **StiWebViewer** component are saved in cache before report viewing. Then images are output using cache. This result in additional requests to the server for extracting images from cache and sending them to the client. This problem can be solved using the **ButtonImagesPath** property of the **StiWebViewer** component. The first thing that should be done is to place a folder with toolbar images on the server (they can be taken from the standard delivery of the product). Then it is necessary to specify the path to this folder using the **ButtonImagesPath** property.

6.6 Localization of StiWebViewer Component

To make the **StiWebViewer** component "speak" another language it is necessary to copy a localization file of the standard delivery to the sever. For example, select the "de.xml" file. Then define the path to this file in the **GlobalizationFile** property of the component. For example, "**Localization\de.xml**".

7 MVC Designer Fx

The **MVC DesignerFx (StiMvcDesignerFx)** component is used for editing 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 a **web browser** and **Flash Player**. With **MVC DesignerFx** you can create, edit, save, view, and print reports on any computer with any operating system, where there is access to the Internet and there is a web browser with support for Flash Player version 11.1 or higher. The **MVC DesignerFx** component can be divided into two parts - client and server. The client side is a graphical front-end wrap of the designer developed on base of the Flash technology. The server side is the report engine, as well as a module that performs the functions of receiving requests, processing and passing data arranged on the client side. These two parts are assembled into a single library and represented as the **ASP.NET MVC** component.

The **MVC DesignerFx** component is located in the **StiMvcDesignerFx** class. The minimum required set of libraries for components is
Stimulsoft.Base.dll
Stimulsoft.Report.dll
Stimulsoft.Report.Mvc.dll

7.1 How it works?

To run the designer it is required to place the **MVC DesignerFx** component on the view page, set the necessary options, and, in the view controller, to determine the appropriate action. When you run the report designer the following actions occur:

- The .NET component generates **HTML** and **Javascript** code needed to display reports and work with the report designer.
- When the component is output the Flash application runs. This requests a report template on the server side and displays it in the edit window.
- Various actions such as previewing the report, saving the report template, exporting it, using parameters, sorting and drill down, cause a specific action on the server side, where you can perform the necessary manipulations with the report.

7.2 Editing and Creating Reports

For editing a report template, you must add the **MVC DesignerFx** component on a report page and set the minimum required options, and in the view controller to determine the appropriate action. The **GetReportTemplate** action returns the report template to the designer.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {  
    Actions =  
    {  
        GetReportTemplate = "GetReportTemplate"  
    }  
}) %>
```

```
}
```

```
) %>
```

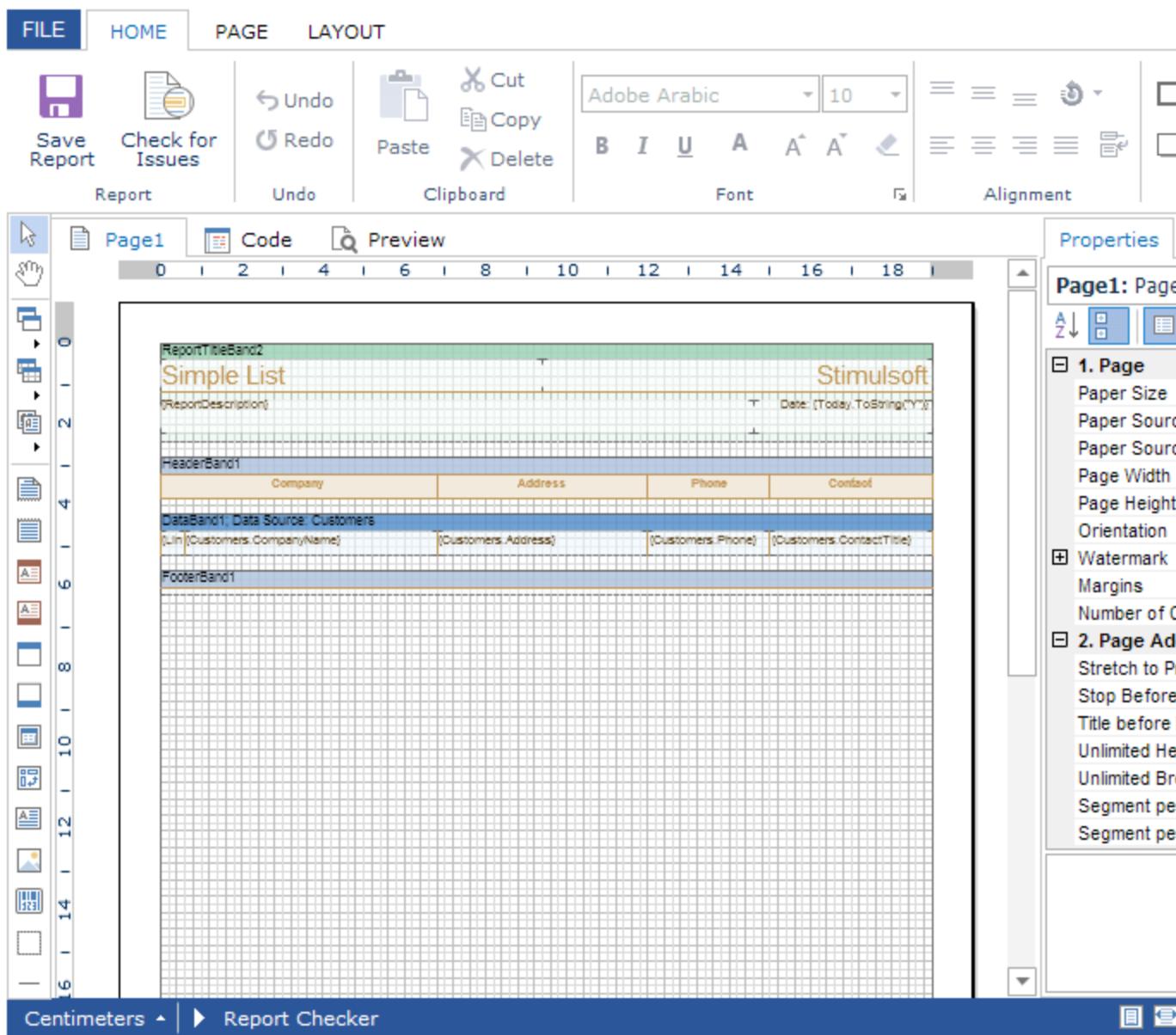
Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        GetReportTemplate = "GetReportTemplate"
    }
})
```

Controller:

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

    return StiMvcDesignerFx.GetReportTemplateResult(report);
}
```



This action is automatically called after the report designer is loaded. It is also possible to return the report template when creating a new report by the relevant item of the main menu of the designer. The **CreateReportTemplate** action is used for this.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        CreateReportTemplate = "CreateReportTemplate"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        CreateReportTemplate = "CreateReportTemplate"
    }
})
```

Controller:

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

    StiReport report = new StiReport();
    report.RegData("Demo", ds);
    report.Dictionary.Synchronize();

    return StiMvcDesignerFx.GetReportTemplateResult(report);
}
```

7.3 Previewing Reports

The **MVC DesignerFx** component has a preview mode of edited reports. To preview the report, you should simply navigate to the appropriate tab in the designer. The report template will be transferred to the server side, rendered and displayed in the embedded viewer. To preview the report, you should define a special action - **GetReportSnapshot**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        GetReportTemplate = "GetReportTemplate",
        GetReportSnapshot = "GetReportSnapshot"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        GetReportTemplate = "GetReportTemplate",
        GetReportSnapshot = "GetReportSnapshot"
    }
})
```

})

Controller:

```
public ActionResult GetReportSnapshot()
{
    return StiMvcDesignerFx.GetReportSnapshotResult();
}
```

Also, before previewing the report it is possible to carry out any manipulation with it (for example, connect the data). To do this, you should use the method - **GetReportObject()**, which returns the

currently editable report template.

Controller:

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

    StiReport report = StiMvcDesignerFx.GetReportObject();
    report.RegData("Demo", ds);

    return StiMvcDesignerFx.GetReportSnapshotResult(report);
}
```

7.4 Preview Features

The preview window of the **MVC DesignerFx** report component is an interactive report viewer that can print and export reports, and also supports sending data via Email. For working with the built-in report viewer it is necessary to determine the following.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        ExportReport = "ExportReport",
        EmailReport = "EmailReport",
        DesignerEvent = "DesignerEvent"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        ExportReport = "ExportReport",
        EmailReport = "EmailReport",
        DesignerEvent = "DesignerEvent"
    }
})
```

Controller:

```
public ActionResult ExportReport()
{
```

```
        return StiMvcDesignerFx.ExportReportResult();
    }

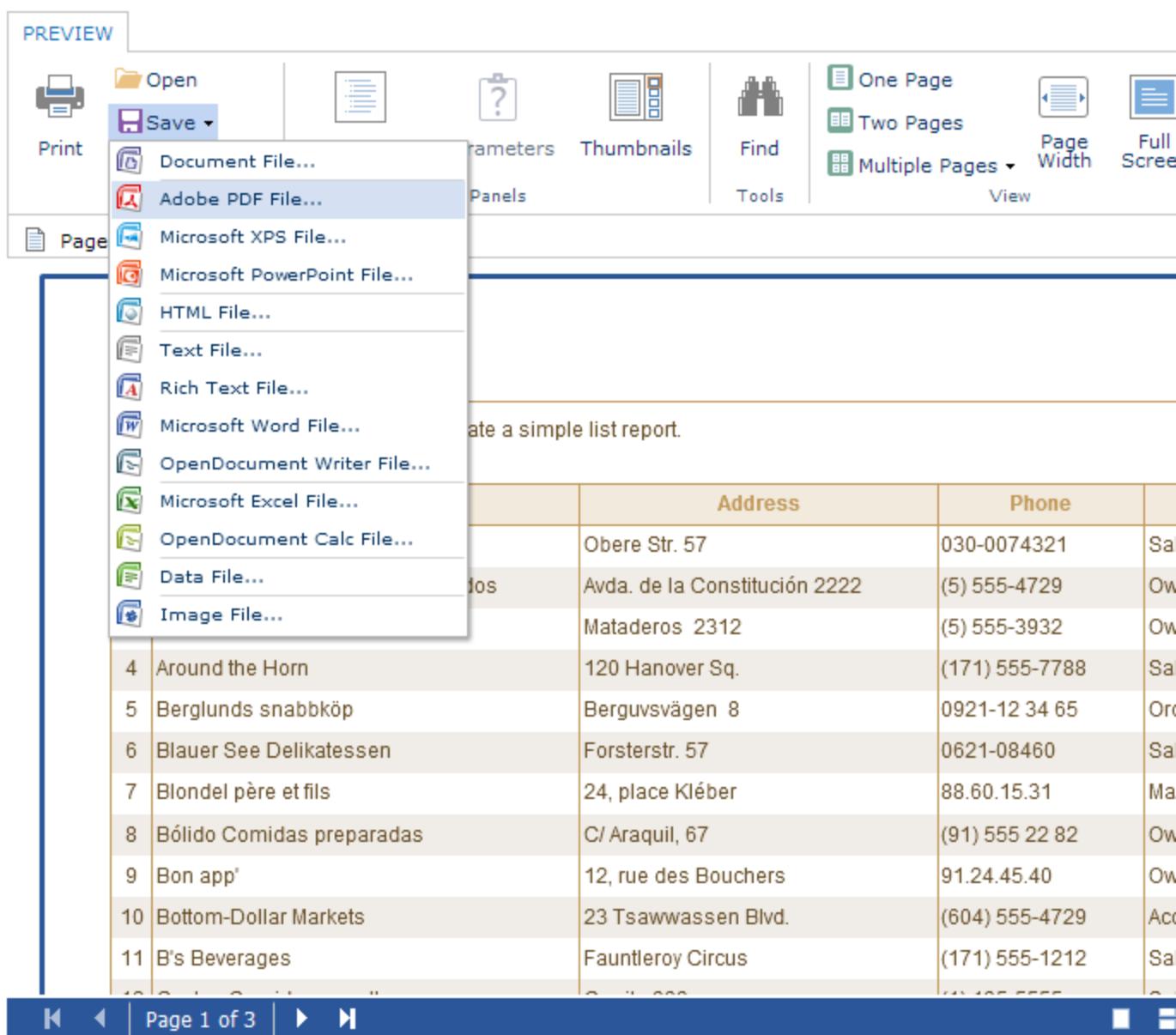
    public ActionResult EmailReport()
    {
        StiEmailOptions options = StiMvcDesignerFx.GetEmailOptions();

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

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

        return StiMvcDesignerFx.EmailReportResult(options);
    }

    public ActionResult DesignerEvent()
    {
        return StiMvcDesignerFx.DesignerEventResult();
    }
}
```



In any of the above mentioned actions it is accepted to modify a report template, if required. If one of these actions is absent, then the corresponding one will not be available in the preview window.

7.5 Previewing C#/VB.NET Report Code

The **MVC DesignerFx** component has the ability to view **C#/VB.NET** code of the editable report template. By default, this option is disabled. In order to enable it you should set the **ShowCodeTab** option to true. The special action - **DesignerEvent** should be defined.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
```

```

Actions =
{
    DesignerEvent = "DesignerEvent"
},

Toolbar =
{
    ShowCodeTab = true
}
}) %>

```

Razor:

```

@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        DesignerEvent = "DesignerEvent"
    },

    Toolbar =
    {
        ShowCodeTab = true
    }
})

```

Controller:

```

public ActionResult DesignerEvent()
{
    return StiMvcDesignerFx.DesignerEventResult();
}

```

Also, the window with the code of the report will be displayed in case of an error compiling the report. For this it is necessary to select one of the errors in the proposed list.

7.6 Saving Reports

To process the saved report in the **MVC DesignerFx** component two special actions are used - **SaveReportTemplate** and **SaveReportTemplateAs**. The **SaveReportTemplate** action is triggered by clicking the Save Report button on the quick access toolbar, or by selecting the Save Report of the main menu in the designer. The **SaveReportTemplateAs** action is called when you select Save As of the main menu in the designer. If the second action is not defined in the settings of the report designer, the default command is invoked dialog box to save the report on the local computer. If the action is specified, the dialog box does not appear. Below is an example of the action code to save a report - **SaveReportTemplate** (action code **SaveReportTemplateAs** will be similar).

ASPX:

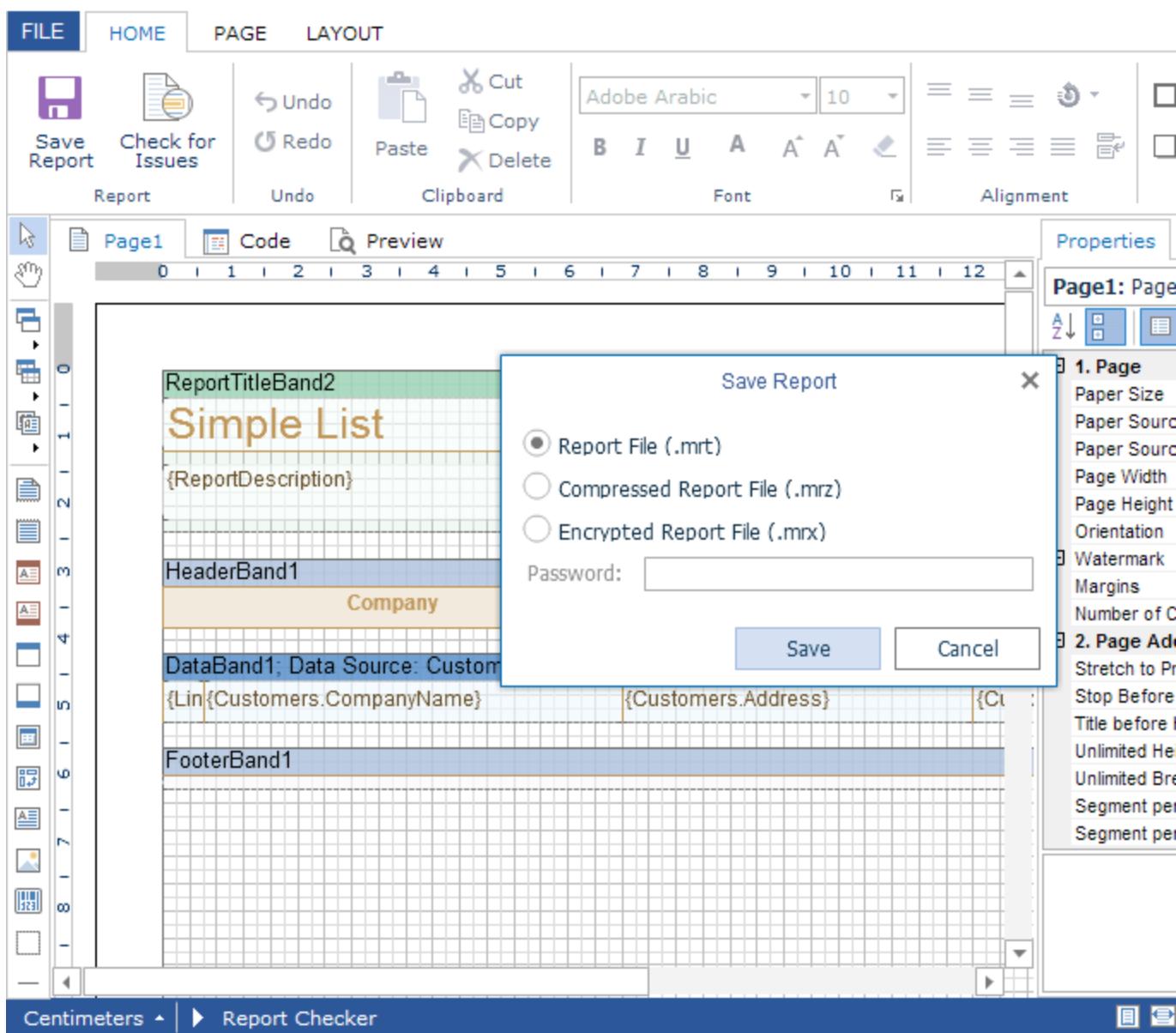
```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {  
    Actions =  
    {  
        SaveReportTemplate = "SaveReportTemplate",  
        SaveReportTemplateAs = "SaveReportTemplateAs"  
    }  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {  
    Actions =  
    {  
        SaveReportTemplate = "SaveReportTemplate",  
        SaveReportTemplateAs = "SaveReportTemplateAs"  
    }  
})
```

Controller:

```
public ActionResult SaveReportTemplate()  
{  
    StiReport report = StiMvcDesignerFx.GetReportObject();  
  
    string packedReport = report.SavePackedReportToString();  
    // ...  
    // Here the save report code  
    // ...  
  
    return StiMvcDesignerFx.SaveReportResult();  
}
```



The user identifies options and the location to save the report in the action to save a report. For example, the report can be saved in the database or as a file on the server side. This action should return a response to the designer of the result of saving the report. The answer may be of several types - standard, Boolean, integer, and string.

No action will occur when there will be a standard reply (without parameters). After saving the user will continue to work with the report template.

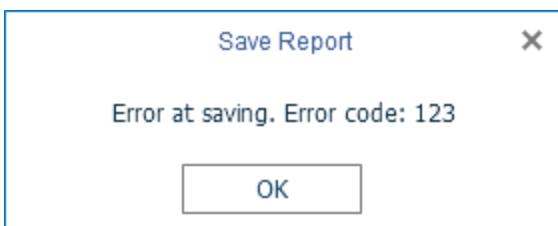
If the action returns true, then after saving the web designer window displays the successful saving of the report.



When returning a string value, you will see a dialog box indicating that the report error occurred. As the error text the transmitted string value will be displayed.



When returning an integer value, the user sees an error message of saving the report and an error code, where the error code is a passed integer value.



Saving a report goes in the background mode, without reloading the page in a browser window. If you need some way to visually manage the process of saving the report, you should change the value of the **SaveReportTemplateMode** option of the designer on one of three values - **Hidden**, **Visible** or **NewWindow**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        SaveReportTemplate = "SaveReportTemplate"
    },
    Behavior =
    {
        SaveReportTemplateMode = StiSaveMode.Visible
    }
})%>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {  
    Actions =  
    {  
        SaveReportTemplate = "SaveReportTemplate"  
    },  
  
    Behavior =  
    {  
        SaveReportTemplateMode = StiSaveMode.Visible  
    }  
})
```

If the **SaveReportTemplateMode** option is set to **Visible**, then the action of saving the report will be called in the current browser window in the visible mode. If the **SaveReportTemplateMode** option is set to **NewWindow**, the action of saving the report will be called in the new browser window. By default, this option is set to **Hidden**, i.e. the action to save a report is called in the **AJAX** mode and does not appear in the browser window. For the **SaveReportTemplateAsMode** option the same values and behavior can be applied.

You should use the additional **GetNewReportFlag()** method if you need to know whether the previously loaded report or a new report created in the designer is saved.

Controller:

```
public ActionResult SaveReportTemplate()  
{  
    StiReport report = StiMvcDesignerFx.GetReportObject();  
    string packedReport = report.SavePackedReportToString();  
  
    string isNewReport = StiMvcDesignerFx.GetNewReportFlag();  
  
    if (isNewReport)  
    {  
        // Here the save new report code  
    }  
    else  
    {  
        // Here the save edited report code  
    }  
  
    return StiMvcDesignerFx.SaveReportResult();  
}
```

7.7 Localizing Designer

To localize the interface of the **MVC DesignerFx** component, you should use the **Localization** option, as well as a special action - **GetLocalization**.

ASPX:

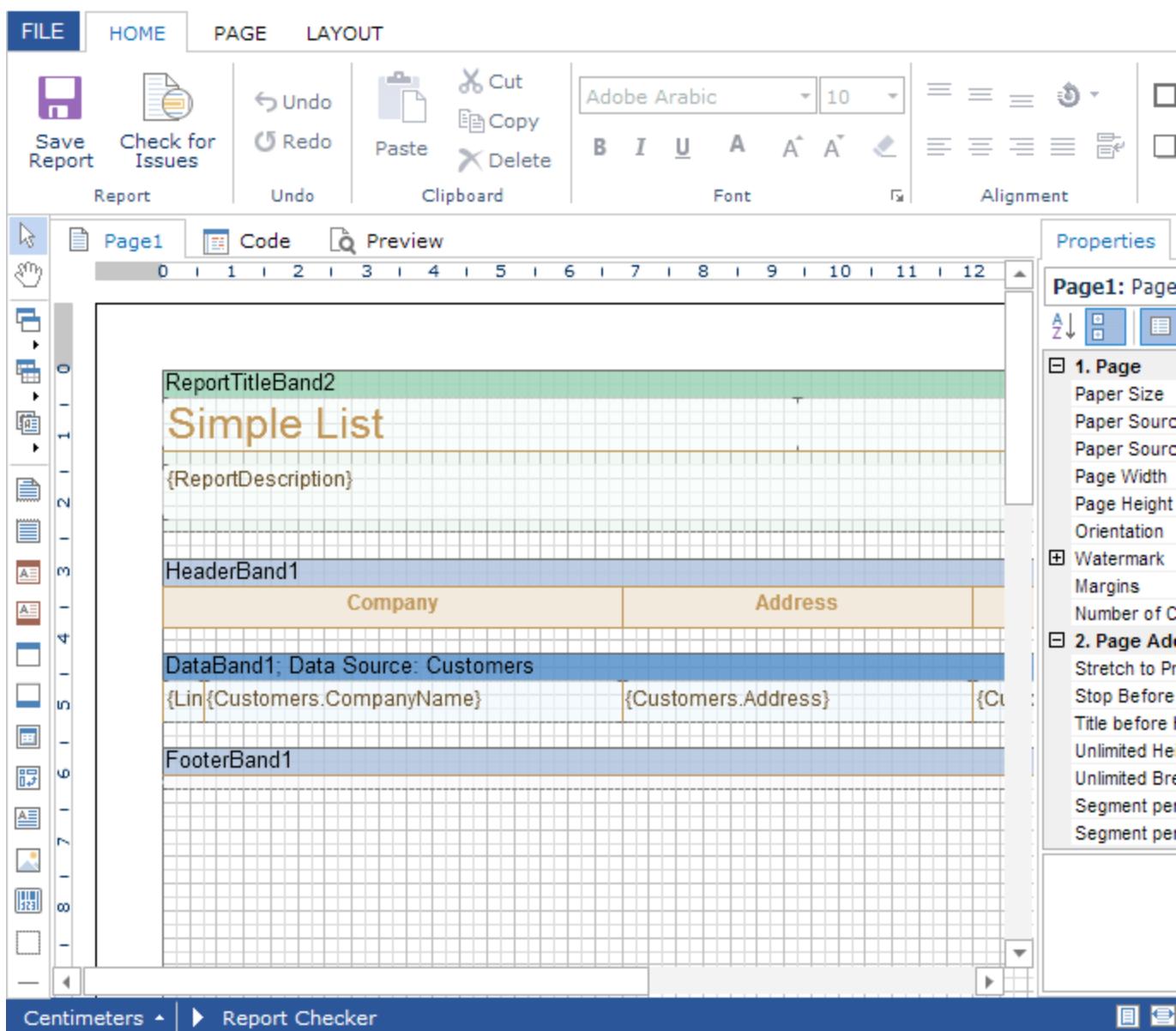
```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {  
    Actions =  
    {  
        GetLocalization = "GetLocalization"  
    },  
  
    Localization = "~/Content/Localization/en.xml"  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {  
    Actions =  
    {  
        GetLocalization = "GetLocalization"  
    },  
  
    Localization = "~/Content/Localization/en.xml"  
})
```

Controller:

```
public ActionResult GetLocalization()  
{  
    return StiMvcDesignerFx.GetLocalizationResult();  
}
```



The user interface of the report designer allows you to select the desired location from the available list. Use the **LocalizationDirectory** option to specify the location where you stored the localization files.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        GetLocalization = "GetLocalization"
    },
    LocalizationDirectory = "~/Content/Localization"
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        GetLocalization = "GetLocalization"
    },
    LocalizationDirectory = "~/Content/Localization"
})
```

If the **Localization** option is set then when you run the report designer the specified localization option will be applied. If the option is not specified, then the localization selected from a list of available ones on the panel of the report designer will be loaded.

7.8 Additional Actions of Designer

There are some additional actions of the of the **MVC DesignerFx** component. These are the actions - **DesignerEvent** and **Exit**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        DesignerEvent = "DesignerEvent",
        Exit = "Exit"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        DesignerEvent = "DesignerEvent",
        Exit = "Exit"
    }
})
```

The **DesignerEvent** action is used to process the data requested by the designer on the server side. These data are processing queries associated with testing database connections, obtaining the data source columns, obtaining images for some components of the report, work with the **Google Docs** cloud storage, as well as for the preparation of the report code. If this action is not specified, the above functionality will not be available. This action is defined in the view controller as follows.

Controller:

```
public ActionResult DesignerEvent()
{
    return StiMvcDesignerFx.DesignerEventResult();
}
```

The **Exit** action is called when you click the Exit button in the main menu of the designer. If this action is not specified, pressing the button will switch to the URL address specified in the **ExitUrl** option of the designer. If this option is not specified, then the Exit button will be hidden.

An example of using the **Exit** action.

Controller:

```
public ActionResult ExitDesigner()
{
    return View(" MainPage");
}
```

An example of using the **ExitUrl** option.

ASPX:

```
<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        DesignerEvent = "DesignerEvent"
    },
    Behavior =
    {
        ExitUrl = "http://www.stimulsoft.com"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Actions =
    {
        DesignerEvent = "DesignerEvent"
    },
    Behavior =
    {
        ExitUrl = "http://www.stimulsoft.com"
    }
})
```

7.9 Additional Methods

There are several helper methods for working with the report designer. Additional methods of the **MVC DesignerFx** component include the following methods.

GetReportObject()

Returns a report object with which the designer is currently working. It is allowed to make the necessary actions with this - register new data sets, change report properties, assign parameters or load another report into the object.

```
public ActionResult ExportReport()
{
    StiReport report = StiMvcDesignerFx.GetReportObject();
    report.ReportName = "MyReportName";

    return StiMvcDesignerFx.ExportReportResult(report);
}
```

GetRouteValues()

Returns the value for the URL, with which the page of the designer was opened. So, it is possible to obtain a collection of page parameters of running in any action of the designer.

```
public ActionResult ExportReport()
{
    RouteValueDictionary routeValues = StiMvcDesignerFx.GetRouteValues();

    return StiMvcDesignerFx.ExportReportResult(report);
}
```

Also, the URL value can be obtained by the name parameter, specifying it in the called action of the designer.

```
public ActionResult ExportReport(string id)
{
    return StiMvcDesignerFx.ExportReportResult(report);
}
```

The difference between this method of obtaining the parameter values from the additional method `GetRouteValues()` is that the name of the controller and action will depend on the settings of the designer and the called action.

GetNewReportFlag()

Gets a value indicating whether the previously loaded report or a new report created in the designer is saved. This method is used in the action to save a report.

Controller:

```

public ActionResult SaveReportTemplate()
{
    StiReport report = StiMvcDesignerFx.GetReportObject();
    string packedReport = report.SavePackedReportToString();

    string isNewReport = StiMvcDesignerFx.GetNewReportFlag();

    if (isNewReport)
    {
        // Here the save new report code
    }
    else
    {
        // Here the save edited report code
    }

    return StiMvcDesignerFx.SaveReportResult();
}

```

7.10 MVC Designer Fx Settings

Setting the **MVC DesignerFx** is performed with the options that are in the **StiMvcDesignerFxOptions** class. Options are divided into the following groups - Actions, Server, Appearance, Data Dictionary, Main Menu, Toolbar, Preview Panel, Behavior, Exporting, Send Email, Printing.

An example of applying options of the designer.

ASPX:

```

<%= Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Theme = StiTheme.Office2013,
    Localization = "~/Content/Localization/en.xml",
    LocalizationDirectory = "~/Content/Localization",

    Actions =
    {
        GetReportTemplate = "GetReportTemplate",
        GetReportSnapshot = "GetReportSnapshot",
        ExportReport = "ExportReport"
    },
    Exports =
    {
        ShowExportDialog = true,
        ShowExportToDbf = false,
        ShowExportToDif = false
    },
    Print =
    {

```

```

        AutoPageOrientation = true,
        AutoPageScale = true
    }
}) %>

```

Razor:

```

@Html.Stimulsoft().StiMvcDesignerFx(new StiMvcDesignerFxOptions() {
    Theme = StiTheme.Office2013,
    Localization = "~/Content/Localization/en.xml",
    LocalizationDirectory = "~/Content/Localization",

    Actions =
    {
        GetReportTemplate = "GetReportTemplate",
        GetReportSnapshot = "GetReportSnapshot",
        ExportReport = "ExportReport"
    },
    Exports =
    {
        ShowExportDialog = true,
        ShowExportToDbf = false,
        ShowExportToDif = false
    },
    Print =
    {
        AutoPageOrientation = true,
        AutoPageScale = true
    }
})

```

Main settings.

Name	Description
Theme	Changes the theme of the designer. The list of available themes is in the StiThemeFx class.
Localization	Sets the path to the localization XML file. The path can be absolute or relative. If the option is not specified, then the localization will be loaded, selected from a list of available locations on the panel of the report designer.
LocalizationDirectory	Specifies the path to the directory with the localization XML files. Localization files will be loaded in the selection list of the localization on the designer panel.
Width	Specifies the width of the component to the required units that are defined in the Unit class. Available values are - Unit.Pixel() ,

	Unit.Point() and Unit.Percentage() . By default, the width is 100% .
Height	Specifies the height of a component in the required units that are defined in the Unit class. Available values are - Unit.Pixel() , Unit.Point() and Unit.Percentage() . By default, the height is 800 pixels .

7.10.1 Actions

Name	Description
GetReportTemplate	Specifies the name of the action method for preparing the report template when loading the designer.
CreateReportTemplate	Specifies the name of the action method for preparing the report template when creating a new report in the designer.
SaveReportTemplate	Specifies the name of the action method for saving the report template on the server side.
SaveReportTemplateAs	Specifies the name of the action method for saving the report template on the server side, when you select Save As. If the action is not specified, the built-in method of saving a report template to a local disk will be used.
GetReportSnapshot	Specifies the name of the action method for preparing the rendered report in the preview window.
ExportReport	Specifies the name of the action method to export the report to the required format.
EmailReport	Specifies the name of the action method to send the report by Email.
GetLocalization	Specifies the name of the action method for loading a localization XML file of the designer. If the action is not specified, the English localization will be set.
DesignerEvent	Specifies the name of the action method of the report designer that will process requested data on the server side.
Exit	Specifies the name of the action method to navigate to the presentation by clicking the Exit button in the main menu of the designer.

7.10.2 Server

Name	Description
Controller	Specifies the name of the controller to process queries of the report designer. If this option is not specified, then the query will be used by

	the current controller.
RequestTimeout	Waiting time for the response from the server in seconds, after which an error is thrown. The default setting is 20 seconds . For large reports it is recommended to increase this value.
RepeatCount	Sets the number of repeats of the request when any error of connection with the server occur. The default value is 1 .
EnableDataLogger	Enables saving of all requests/responses of the designer into a special log file that can be stored on a local disk. If there are any errors when working with the designer, it is recommended to enable this option and analyze the log file.
UseRelativeUrls	Sets the mode of the designer, in which relative links are used for queries on the server. By default, this option is set to false .

7.10.3 Appearance

Name	Description
AutoHideScrollbars	Hides the scroll bar in the preview window if the entire page fits on the screen. By default, this option is set to false .
OpenLinksTarget	The target window to open the link contained in the rendered report. By default, this option is set to " _blank " (new window).
OpenExportedReportTarget	The target window to open the export file from the designer. By default, this option is set to " _blank " (new window).
ShowToolipsHelp	Displays links to online help for controls of the designer on mouse over. By default, this option is set to true .
ShowFormsHelp	Displays links to online help in titles of dialog windows of the designer. By default, this option is set to true .
ShowFormsHint	Shows hints when you hover on elements of the interface in dialogs windows of the designer. By default, this option is set to true .
ParametersPanelColumnsCount	Sets the number of columns to display the report. The default partition is made on 2 columns.
ParametersPanelEditorWidth	Sets the width of the elements of the parameters panel in pixels. The default width is 200 pixels.
DateFormat	Sets the format of date and time for the designer dialogs, as well as for the variables of the appropriate type in the parameters panel. The default date and time format set on the server is used.
FlashWMode	Specifies the mode to display Flash applications on a browser page. The default it is set to " window ".

AboutDialogTextLine1	Specifies your own text to display in the About box. The given text will be displayed in the first line of the box.
AboutDialogTextLine2	Specifies your own text to display in the About box. The given text will be displayed in the second line of the box.
AboutDialogUrl	Sets a URL to display in the About box.
AboutDialogUrlText	Specifies the text for a URL to display in the About box.
ShowCancelButton	Shows/hides the Cancel button in the load data window from the server side. By default, this option is set to false .
GridSizeInCentimetres	Sets the size of the grid on a report page in centimeters. The default value is set to 0.2 inches .
GridSizeInHundredthsOfInches	Sets the size of the grid on a report page in hundredths of an inch. The default value is 10 hundredths of an inch .
GridSizeInInches	Sets the size of the grid on a report page in inches. The default value is 0.1 inches .
GridSizeInMillimeters	Sets the size of the grid on a report page in millimeters. The default value is 2 millimeters .
BrowserTitle	Sets the text of the title of the current browser window.

7.10.4 Data Dictionary

Name	Description
AllowModifyDictionary	Allows/forbids editing data dictionary of the designer. By default this option is set to true .
AllowModifyConnections	Allows/forbids editing connections in the data dictionary of the designer. By default this option is set to true .
AllowModifyDataSources	Allows/forbids editing data sources in the data dictionary of the designer. By default this option is set to true .
AllowModifyRelations	Allows/forbids editing relations in the data dictionary of the designer. By default this option is set to true .
AllowModifyVariables	Allows/forbids editing variables in the data dictionary of the designer. By default this option is set to true .
ShowConnectionType	Shows/hides the type of a connection in the data dictionary of the designer. By default this option is set to true .
ShowActionsMenu	Shows/hides the menu Actions in the data dictionary of the designer. By default this option is set to true .
ShowTestConnectionButt	Shows/hides the button Test in the dialog window of editing the

on	connection. By default this option is set to true .
ShowOnlyAliasForBusinessObjects	Shows/hides aliases for business objects in the designer. By default this option is set to true .
ShowOnlyAliasForConnections	Shows/hides aliases for connections in the designer. By default this option is set to true .
ShowOnlyAliasForDataSources	Shows/hides aliases for data sources in the designer. By default this option is set to true .
ShowOnlyAliasForDataRelations	Shows/hides aliases for connections in the designer. By default this option is set to true .
ShowOnlyAliasForDataColumns	Shows/hides aliases for data columns in the designer. By default this option is set to true .
ShowOnlyAliasForVariables	Shows/hides aliases for variables in the designer. By default this option is set to true .

7.10.5 Main Menu

Name	Description
ShowNew	Shows/hides the main menu item New . By default this option is set to true .
ShowNewReport	Shows/hides the main menu item New Report . By default this option is set to true .
ShowNewReportWithWizard	Shows/hides the main menu item New Report With Wizard . By default this option is set to true .
ShowNewPage	Shows/hides the main menu item New Page . By default this option is set to true .
ShowOpenReportFromGoogleDocs	Shows/hides the main menu item Open Report from Google Docs . By default this option is set to true .
ShowOpenReport	Shows/hides the main menu item Open Report . By default this option is set to true .
ShowSaveReport	Shows/hides the main menu item Save Report . By default this option is set to true .
ShowSaveAs	Shows/hides the main menu item Save Report As . By default this option is set to true .
ShowSaveAsToGoogleDocs	Shows/hides the main menu item Save Report As to Google Docs . By default this option is set to true .
ShowDeletePage	Shows/hides the main menu item Delete Page . By default this option is set to true .

ShowPreview	Shows/hides the main menu item Preview . By default this option is set to true .
ShowPreviewAsPdf	Shows/hides the main menu item Preview as PDF . By default this option is set to true .
ShowPreviewAsHtml	Shows/hides the main menu item Preview as HTML . By default this option is set to true .
ShowPreviewAsXps	Shows/hides the main menu item Preview as XPS . By default this option is set to true .
ShowClose	Shows/hides the main menu item Close . By default this option is set to true .
ShowAbout	Shows/hides the main menu item About . By default this option is set to true .
ShowExitButton	Shows/hides the main menu item Exit . By default this option is set to true .
Caption	Sets the text of the main menu title in the designer.

7.10.6 Toolbar

Name	Description
ShowSelectLanguage	Shows/hides the list of designer localizations. By default this option is set to true .
ShowCodeTab	Shows/hides the code preview window of the edited report. By default this option is set to true .
ShowPreviewReportTab	Shows/hides the preview window of the edited report. By default this option is set to true .
ShowDictionaryTab	Shows/hides the data dictionary of the designer. By default this option is set to true .
ShowEventsTab	Shows/hides the tab of the report events on the panel of the property editor. By default this option is set to true .
Zoom	Sets the zoom of the report template. By default it is set to 100%. The value varies from 10 to 500%. One of the following values are available: StiZoomModeFx.Default – when you run the designer, the previously used zoom value is used (set by default). StiZoomModeFx.OnePage – when you run the designer one page entirely is shown in the designer window. StiZoomModeFx.PageWidth – when you run the designer the report page is zoomed by the page width. StiZoomModeFx.PageHeight – when you run the designer the

	report page is zoomed by the page height.
--	---

7.10.7 Preview Panel

Name	Description
Visible	Shows/hides the preview panel. By default this option is set to true .
ShowNavigatePanel	Shows/hides the navigation panel of preview. By default this option is set to true .
ShowViewModePanel	Shows/hides the panel of selecting the preview mode. By default this option is set to true .
ShowPrintButton	Shows/hides the button Print on the preview panel toolbar. By default this option is set to true .
ShowOpenButton	Shows/hides the button Open on the preview panel toolbar. By default this option is set to true .
ShowSaveButton	Shows/hides the button Save on the preview panel toolbar. By default this option is set to true .
ShowSendEmailButton	Shows/hides the button Send Email on the preview panel toolbar. By default this option is set to true .
ShowBookmarksButton	Shows/hides the button Bookmarks on the preview panel toolbar. By default this option is set to true .
ShowParametersButton	Shows/hides the button Parameters on the preview panel toolbar. By default this option is set to true .
ShowThumbnailsButton	Shows/hides the button Thumbnails on the preview panel toolbar. By default this option is set to true .
ShowFindButton	Shows/hides the button Find on the preview panel toolbar. By default this option is set to true .
ShowFullScreenButton	Shows/hides the button Full Screen on the preview panel toolbar. By default this option is set to true .
ShowAboutButton	Shows/hides the button About on the preview panel toolbar. By default this option is set to true .
ShowFirstPageButton	Shows/hides the button First Page on the preview panel toolbar. By default this option is set to true .
ShowPreviousPageButton	Shows/hides the button Previous Page on the preview panel toolbar. By default this option is set to true .
ShowGoToPageButton	Shows/hides the button Go To Page on the preview panel toolbar. By default this option is set to true .

ShowNextPageButton	Shows/hides the button Next Page on the preview panel toolbar. By default this option is set to true .
ShowLastPageButton	Shows/hides the button Last Page on the preview panel toolbar. By default this option is set to true .
ShowSinglePageViewModeButton	Shows/hides the button Single Page on the preview panel toolbar. By default this option is set to true .
ShowContinuousPageViewModeButton	Shows/hides the button Continuous Page on the preview panel toolbar. By default this option is set to true .
ShowMultiplePageViewModeButton	Shows/hides the button Multiple Page on the preview panel toolbar. By default this option is set to true .
ShowZoomButtons	Shows/hides the button to select the preview zoom. By default this option is set to true .

7.10.8 Behavior

Name	Description
ShowSaveFileDialog	Shows/hides the dialog to enter the name of the report when it is saved. By default this option is set to false .
SetReportModifiedWhenOpening	Mark the loaded report as changed when running the designer. By default this option is set to false .
AllowModifyTemplate	Enables/disables editing loaded elements of the report template. Creating and editing new items will be allowed regardless of the value of this option. By default this option is set to true .
AutoSaveInterval	Sets time (in minutes) after which the designer will automatically call the action to save a report. If the interval is 0 minutes, the autosave is disabled. The default value is 0 minutes .
SaveReportTemplateMode SaveReportTemplateAsMode	Controls the process of saving the report. It can take one of three values: StiSaveMode.Hidden - the action to save a report is called in the AJAX mode and does not appear in the browser window (default). StiSaveMode.Visible - the action to save a report will be called in the current browser window in the visible mode. StiSaveMode.NewWindow - the action to save a report will be called in a new window (tab) of the browser.
RunWizardAfterLoad	Displays the report wizard when you run the report designer. By default this option is set to false .

DesignerEventFunction	Specifies the name of the Javascript function to be called when certain actions of the designer are done. The function takes a string parameter in which the identifier of the action of the designer is passed.
ExitUrl	Sets the URL for the transition when you click the Exit button in the main menu of the designer.

7.10.9 Exporting

Name	Description
ShowExportDialog	Shows/hides the dialog windows of export parameters. If this option is disabled the exporting will go with parameters set by default. By default this option is set to true .
ShowExportToDocument	Shows/hides the menu item Document File . By default this option is set to true .
ShowExportToPdf	Shows/hides the menu item Adobe PDF File . By default this option is set to true .
ShowExportToXps	Shows/hides the menu item Microsoft XPS File . By default this option is set to true .
ShowExportToPowerPoint	Shows/hides the menu item Microsoft PowerPoint 2007/2010 File . By default this option is set to true .
ShowExportToHtml	Shows/hides the menu item HMTL File . By default this option is set to true .
ShowExportToHtml5	Shows/hides the menu item HMTL5 File . By default this option is set to true .
ShowExportToMht	Shows/hides the menu item MHT Web Archive . By default this option is set to true .
ShowExportToText	Shows/hides the menu item Text File . By default this option is set to true .
ShowExportToRtf	Shows/hides the menu item Rich Text File . By default this option is set to true .
ShowExportToWord2007	Shows/hides the menu item Microsoft Word 2007/2010 File . By default this option is set to true .
ShowExportToOpenDocument Writer	Shows/hides the menu item OpenDocument Writer File . By default this option is set to true .
ShowExportToExcel	Shows/hides the menu item Microsoft Excel File . By default this option is set to true .

ShowExportToExcelXml	Shows/hides the menu item Microsoft Excel Xml File . By default this option is set to true .
ShowExportToExcel2007	Shows/hides the menu item Microsoft Excel 2007/2010 File . By default this option is set to true .
ShowExportToOpenDocumentCalc	Shows/hides the menu item OpenDocument Calc File . By default this option is set to true .
ShowExportToCsv	Shows/hides the menu item CSV File . By default this option is set to true .
ShowExportToDbf	Shows/hides the menu item DBF File . By default this option is set to true .
ShowExportToXml	Shows/hides the menu item XML File . By default this option is set to true .
ShowExportToDif	Shows/hides the menu item Data Interchange Format (DIF) File . By default this option is set to true .
ShowExportToSylk	Shows/hides the menu item Symbolic Link (SYLK) File . By default this option is set to true .
ShowExportToImageBmp	Shows/hides the menu item BMP Image . By default this option is set to true .
ShowExportToImageGif	Shows/hides the menu item GIF Image . By default this option is set to true .
ShowExportToImageJpeg	Shows/hides the menu item JPEG Image . By default this option is set to true .
ShowExportToImagePcx	Shows/hides the menu item PCX Image . By default this option is set to true .
ShowExportToImagePng	Shows/hides the menu item PNG Image . By default this option is set to true .
ShowExportToImageTiff	Shows/hides the menu item TIFF Image . By default this option is set to true .
ShowExportToImageMetafile	Shows/hides the menu item Windows Metafile . By default this option is set to true .
ShowExportToImageSvg	Shows/hides the menu item Scalable Vector Graphics (SVG) File . By default this option is set to true .
ShowExportToImageSvgz	Shows/hides the menu item Compressed SVG (SVGZ) File . By default this option is set to true .

7.10.10 Send Email

Name	Description
ShowEmailDialog	Shows/hides the options dialog box to send the report by Email. If the dialog box is disabled, then sending it by Email will be made with settings on the server side in the EmailReport action. By default, this option is set to true.
ShowExportDialog	Shows/hides the export options dialog boxes when sending Email. If this option is disabled, the export will be carried out with the default settings. By default, this option is set to true.
DefaultEmailAddress	Defines the Email recipient, i.e. an address to which an email with an attached report will be sent.

7.10.11 Printing

Name	Description
ShowPrintDialog	Shows/hides the options dialog box to print a report. If the dialog box is disabled, printing will be done with the default settings. By default this option is set to true .
AutoPageOrientation	Enables automatic page orientation when printing a report, if the correct format is set in the printer settings. By default this option is set to true .
AutoPageScale	Enables automatic scaling of the page when printing a report, if wrong size is set in the printer settings. By default this option is set to true .
AllowDefaultPrint	Shows/hides the Default item in the print dialog of the report. By default this option is set to true .
AllowPrintToPdf	Shows/hides the Print as PDF item in the print dialog of the report. By default this option is set to true .
AllowPrintToHtml	Shows/hides the Print as HTML item in the print dialog of the report. By default this option is set to true .

8 MVC Viewer

The **MVC Viewer** (StiMvcViewer) component is designed to view reports in the web browser. You do not need to install .NET Framework, ActiveX components or other special plug-ins on the client side. All you need is a web browser. With help of the **StiMvcViewer** one can view, print, and export reports on any computer with any operating system. Since the **StiMvcViewer** uses only HTML and Javascript, it can be run on devices that do not support Flash or **Silverlight** - tablet PC's, smartphones.

The **StiMvcViewer** component uses **AJAX** for executing all actions (loading reports, scrolling pages, zooming, interactivity etc.), that allows you to get rid of reloading the whole page and saves web traffic. The **StiMvcViewer** supports themes, beautiful menu with animation, full screen mode etc.

The **MVC Viewer** component is located in the **StiMvcViewer** class. The minimum required set of libraries for components is

Stimulsoft.Base.dll
Stimulsoft.Report.dll
Stimulsoft.Report.Mvc.dll

8.1 How It Works?

To run the viewer you need to place the **StiMvcViewer** component on the view page. Set the necessary properties to the **StiMvcViewer** component, and, in the view controller, define the necessary actions.

The following actions occur when running the viewer:

- The .NET component generates **HTML** and **Javascript** code that is necessary for showing and working the viewer.
- When the viewer component is output, the Javascript method is running. It requests the first report page or the entire report (depending on the selected mode) on the server side.
- Each action in the viewer (for example, scrolling pages or printing) calls the action on the server side in which one can execute certain manipulations with the report.
- For faster work the viewer saves the report in the cache or a server session that allows you to remove the excess report rendering.

8.2 Showing Reports

In order to preview the report it is necessary to add the component **StiMvcViewer** on the view page and set to it the minimum required properties, and set the appropriate action in the view controller. In order to preview the report it is necessary to add the component StiMvcViewer on the view page and set to it the minimum required properties, and set the appropriate action in the view controller:

```
<system.web>
  <pages>
    <namespaces>
      <add namespace="Stimulsoft.Report.Mvc" />
    </namespaces>
  </pages>
</system.web>
```

The action **GetReportSnapshot** returns the report prepared for the viewer and the action **ViewerEvent** handles the viewer events.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Actions =
        {
            GetReportSnapshot = "GetReportSnapshot",
            ViewerEvent = "ViewerEvent"
        }
    }) %>
```

Razor:

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

Controller:

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

    return StiMvcViewer.GetReportSnapshotResult(report);
}

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



Notice: These actions are mandatory. The viewer will not work correctly without them.

If the report was not rendered before being displayed, the **StiMvcViewer** component will build it automatically. Thus, to display the report, it is allowed to use report templates, rendered reports and report classes.

Controller:

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

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

Controller:

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

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

The sample demonstrates how to create a simple list report.

	Company	Address	Phone	Contact
1	Alfreds Futterkiste	Obere Str. 57	030-0074321	Sales Repres...
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 Repres...
5	Berglunds snabbköp	Berguvsvägen 8	0921-12 34 65	Order Adminis...
6	Blauer See Delikatessen	Forsterstr. 57	0621-08460	Sales Repres...
7	Blondel père et fils	24, place Kléber	88.60.15.31	Marketing Ma...
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 M...
11	B's Beverages	Fauntleroy Circus	(171) 555-1212	Sales Repres...
12	Cactus Comidas para llevar	Cerrito 333	(1) 135-5555	Sales Agent
13	Centro comercial Moctezuma	Sierras de Granada 9993	(5) 555-3392	Marketing Ma...
14	Chop-suey Chinese	Hauptstr. 29	0452-076545	Owner
15	Comércio Mineiro	Av. dos Lusíadas, 23	(11) 555-7647	Sales Associa...
16	Consolidated Holdings	Berkeley Gardens 12 Brewery	(171) 555-2282	Sales Repres...
17	Drachenblut Delikatessen	Walserweg 21	0241-039123	Order Adminis...
18	Du monde entier	67, rue des Cinquante Otages	40.67.88.88	Owner

Also before showing the report it is allowed to change its properties, assign parameters and register data sets, for example:

Controller:

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

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

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

8.3 Localizing Component

In order to localize the interface of the report viewer the **Localization** property is used. The value of this property should point the path to the localization XML file (relative or absolute):

ASPX:

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

Razor:

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

When the report viewer is loaded the localization file will be loaded automatically.

8.4 Using Themes

The **StiMvcViewer** component has the ability to change the theme of visual controls. The **Theme** property is used to change the theme, which can take one of the enumeration values - **StiTheme**:

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(new StiMvcViewerFxOptions() {  
    Theme = StiTheme.Office2013  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {  
    Theme = StiTheme.Office2013
```

{ })

Currently 20 themes are available.

Print Save Send Email Page 1 of 3 100% One Page

Document File... Adobe PDF File... Microsoft XPS File...

Print Save Send Email Page 1 of 3 100% One Page

Document File... Adobe PDF File... Microsoft XPS File...

Print Save Send Email Page 1 of 3 100% One Page

Simple List

The sample demonstrates how to create a simple list.

Company	
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y helados
3	Antonio Moreno Taquería
4	Around the Horn
5	Berglunds snabbköp
6	Blauer See Delikatessen
7	Blondel père et fils
8	Bólido Comidas preparadas
9	Bon app'
10	Bottom-Dollar Markets
11	B's Beverages
12	Cactus Comidas para llevar
13	Centro comercial Moctezuma
14	Chop-suey Chinese
15	Comércio Mineiro
16	Consolidated Holdings
17	Drachenblut Delikatessen
18	Du monde entier

Export Settings

Page Range

All Current Page Pages:

Settings

Image Resolution: 100
Image Compression Method: Jpeg
Allow Editable: No
Image Quality: 75%

Standard PDF Fonts
 Embedded Fonts
 Use Unicode
 Compressed
 Export Rich Text as Image
 PDF/A Compliance
Document Security: 75%
Digital Signature

Open After Export

OK Cancel

It is possible to set color parameters of basic viewer items. Below is a list of available options and their values by default.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(new StiMvcViewerFxOptions() {
    Appearance =
    {
        BackgroundColor = Color.White,
        PageBorderColor = Color.Blue
    },
    Toolbar =
    {
        BackgroundColor = Color.White,
        BorderColor = Color.Gray,
        FontColor = Color.Black,
        FontFamily = "Arial"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Appearance =
    {
        BackgroundColor = Color.White,
        PageBorderColor = Color.Blue
    },
    Toolbar =
    {
        BackgroundColor = Color.White,
        BorderColor = Color.Gray,
        FontColor = Color.Black,
        FontFamily = "Arial"
    }
})
```

8.5 Using Basic Features

The main features of the report viewer are switching between report pages, report zooming, displaying modes of reports. All these operations are performed in the **AJAX**-mode without restarting the browser page. For correct work of the operations it is necessary to define a specific action - **ViewerEvent**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(
```

```
"MvcViewer1", new StiMvcViewerOptions() {
    Actions =
    {
        ViewerEvent = "ViewerEvent"
    }
}) %>
```

Razor:

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

Controller:

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



Notice: These actions are mandatory. The viewer will not work correctly without them.

The **ViewerEvent** action returns the **HTML** report page (or a set of **HTML** pages), rendered with the new settings on the viewer panel. In case of exporting the report, the action returns **FileResult** with the file in the format selected in the Export menu of the report viewer.

8.6 Printing

The component has several ways of printing the report.

Print to PDF

Printing will be done by exporting the report to the PDF format. The advantages include more accurate location and printing of report components compared to other printing options. Disadvantages - required installed browser plug-in to view the PDF.

Print with Preview

Print a report will be made in the pop-up separate browser window. The report can be viewed and, if

necessary, send to a printer or copy to another location as a text or HTML code.

Print without Preview

The report will be sent directly to the printer without preview. After selecting this menu item, a printing dialog box is shown.

Important: When printing to the HTML format, make sure that report page parameters and page setup of the printer (paper size, orientation, margins, padding), and check the print settings of the browser - margins, headers and footers, background images printing.

To work correctly of the printing function it is necessary to define a special action **PrintReport**:

ASPX:

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

Razor:

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

Controller:

```
public ActionResult PrintReport()
{
    // Some code before print
    // ...
    return StiMvcViewer.PrintReportResult();
}
```

8.7 Export reports

StiMvcViewer allows you to export reports to about 30 different formats - **PDF, Microsoft Word,**

Excel, XPS, RTF, image formats, and many others. No additional configuration for the viewer is required to work with exports. If you want to spend some actions before exporting the report, you can define a special action **ExportReport**.

ASPX:

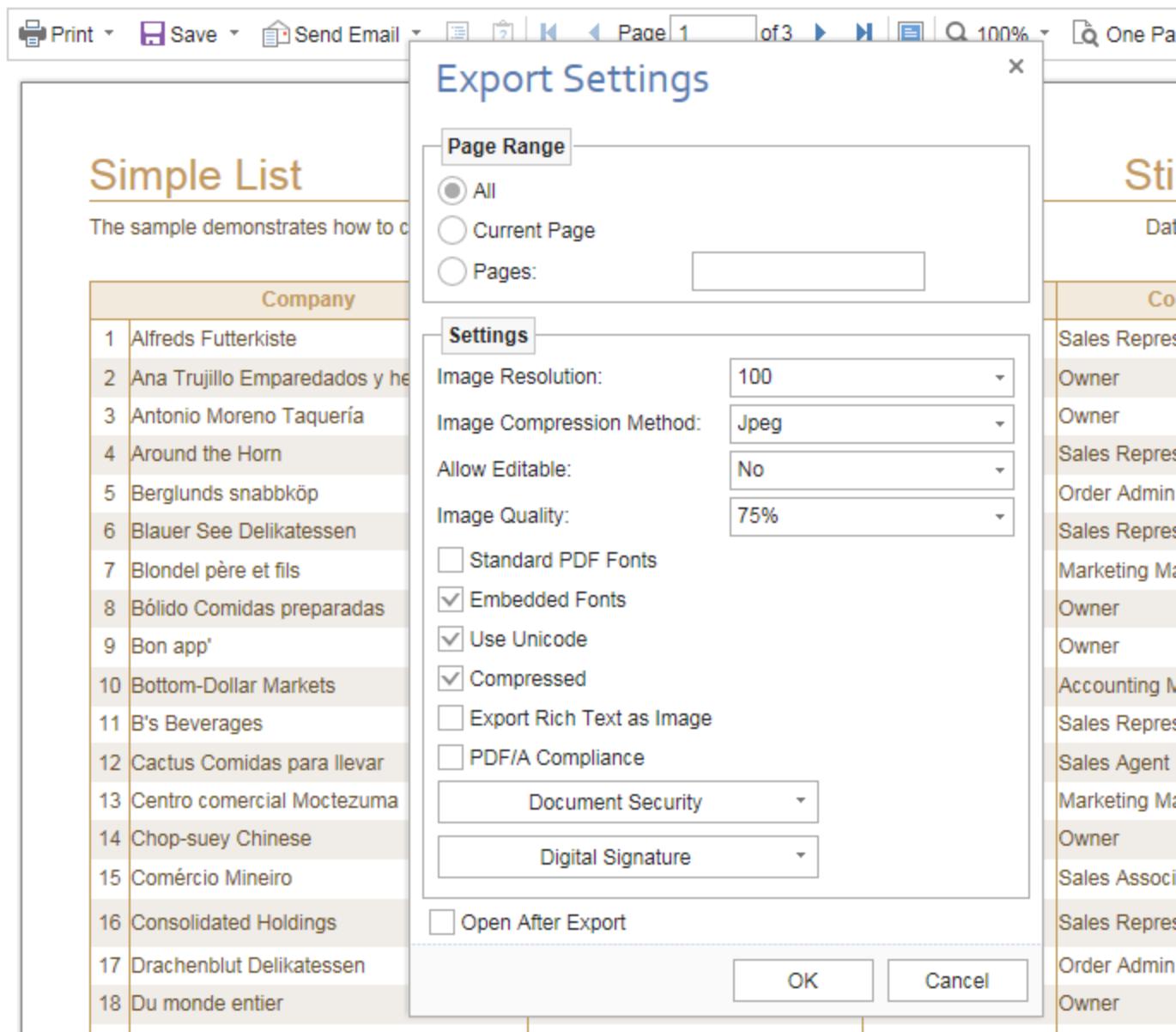
```
<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Actions =
        {
            ExportReport = "ExportReport"
        }
}) %>
```

Razor:

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

Controller:

```
public FileResult ExportReport()
{
    // Some code before export
    // ...
    return StiMvcViewer.ExportReportResult();
}
```



8.8 Viewing Modes

The **StiMvcViewer** component has two modes of displaying reports: with scrollbars and without scrollbars. The **ScrollbarsMode** option operates this and can take values **True** or **False**.

ASPX:

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

```
}
```

Razor:

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

Without scrollbars (this mode is set by default), the viewer displays the page or the whole report, automatically stretching the viewing area. If you specify width and height, then the viewer will cut everything that goes out of the page. In the second mode, unlike the first, when the page goes our of the viewer size, nothing is cut off but the scrollbar appear.

Information: When viewing a report with scrollbars it is necessary to set the height of the viewer, otherwise the default height of 600 pixels will be set.

In the **MVC Viewer** component you can find a full screen mode of displaying reports. The **FullScreenMode** option is used for this. It takes the value true or false.

ASPX:

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

Razor:

```
@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 **MVC Viewer** component supports work with mobile devices. When you switch to the mobile mode, the control mode from the mouse to the touch was switched in the viewer. The size of interface items increased to facilitate control using the touch screen. To manage this mode the **InterfaceType** option is used. It takes the following values.

- **StiInterfaceType.Auto** - the type of the viewer interface is automatically selected depending on the device used (set by default).
- **StiInterfaceType.Mouse** - force use of the interface to manage the viewer using the mouse.
- **StiInterfaceType.Touch** - force use of the touch interface to manage the viewer using the touch screen.

ASPX:

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

Razor:

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

8.9 Using Parameters

A special **Parameters** panel is implemented for working with report parameters in **StiMvcViewer**. To add a parameter to the panel you need to define a variable requested by the user in a report. When previewing a report in the viewer, such a variable will be automatically added to the Parameters panel. It supports all types of report variables (normal variables, date and time, borders, lists, and so on). No additional viewer settings are required. To work correctly with parameters you need to define the specific action **Interaction**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Actions =
        {
            Interaction = "Interaction"
        }
}) %>
```

```
    }
}) %>
```

Razor:

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

Controller:

```
public ActionResult Interaction()
{
    // Some code before apply parameters
    // ...
    return StiMvcViewer.InteractionResult();
}
```

The screenshot shows the Stimulsoft Reports ASPX interface. At the top, there is a toolbar with various icons: Print, Save, Send Email, Help, Page, and a search bar set to 100% zoom. Below the toolbar, there are several input fields for parameters:

- InvoiceNumber: A text input field.
- InvoiceDate: A date input field showing "17.03.2015 17:42:17".
- CustomerID: A text input field.
- Bill To - Name: A text input field.
- Bill To - Address: A text input field.
- Bill To - Address 2: A text input field.
- Bill To - City: A text input field.
- Bill To - State: A text input field.

On the right side, there is a "Ship To" section with fields for Name, ZIP CODE, Street Address, and City. Below these is a calendar control showing the month of March 2015, with the 17th highlighted. There is also a "Reset" button.

At the bottom left, the word "Invoice" is displayed in a large, bold, orange font. To its right, the word "Sti" is partially visible. Below "Invoice", a message reads: "This sample requests parameters for printing an invoice." To the right of this message, the word "Data" is partially visible.

Below the parameter inputs, there is a table for "BILL TO" and "SHIP TO" details. The "BILL TO" section includes fields for Name, Street Address, Address 2, and City, ZIP CODE. The "SHIP TO" section includes fields for Name, Street Address, Address 2, and City, ZIP CODE. To the right of these sections, there are three rows of text: "Invoice #0", "Invoice date 3/17/2015", and "Customer ID 0".

Below the "SHIP TO" section, there is a table for an invoice item. The columns are Unit Name, Description, Qty, Item Price, and Total. The first row shows "Alice Mutton" as the unit name, "20 - 1 kg tins" as the description, "0.00" as the quantity, "\$39.00" as the item price, and "0.00" as the total.

If work with parameters is not required, you can disable this feature. The **ShowParametersButton** option from the **Toolbar** section is used for this. Set this option to false.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Toolbar =
        {
            ShowParametersButton = false
        }
    }) %>
```

Razor:

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

In this case, the settings panel will not be shown, even if the parameters are present in the displayed report.

8.10 Bookmarks

The **MVC Viewer** component supports bookmarks in reports. When displaying such a report on the left side of the page the panel with tabs will be displayed. When you select report bookmark the viewer will open the desired page, and the report item with a bookmark will be highlighted:

The screenshot shows a Stimulsoft Reports interface. On the left, there is a navigation tree with the following categories:

- Detalized Categories
- Beverages
 - Chai
 - Chang
 - Chartreuse verte
 - Côte de Blaye
 - Guaraná Fantástica**
 - Ipo Coffee
 - Lakkaliköori
 - Laughing Lumberjack Lager
 - Outback Lager
 - Rhönbräu Klosterbier
 - Sasquatch Ale
 - Steeleye Stout
- Condiments
- Confections
- Dairy Products
- Grains/Cereals
- Meat/Poultry
- Produce
- Seafood

On the right, there are two tables of product details:

Product Name	Description	Price
3.Chartreuse verte	750 cc per bottle	\$18.00
4.Côte de Blaye	12 - 75 cl bottles	\$263.50
5.Guaraná Fantástica	12 - 355 ml cans	\$4.50
6.Ipo Coffee	18 - 500 g tins	\$46.00
7.Lakkaliköori	500 ml	\$18.00
8.Laughing Lumberjack Lager	24 - 12 oz bottles	\$14.00
9.Outback Lager	24 - 355 ml bottles	\$15.00
10.Rhönbräu Klosterbier	24 - 0.5 l bottles	\$7.75
11.Sasquatch Ale	24 - 12 oz bottles	\$14.00
12.Steeleye Stout	24 - 12 oz bottles	\$18.00

2.Condiments		
1.Aniseed Syrup	12 - 550 ml bottles	\$10.00
2.Chef Anton's Cajun Seasoning	48 - 8 oz jars	\$22.00
3.Chef Anton's Gumbo Mix	36 boxes	\$21.35
4.Genen Shouyu	24 - 250 ml bottles	\$15.50
5.Grandma's Boysenberry Spread	12 - 8 oz jars	\$25.00
6.Gula Malacca	20 - 2 kg bags	\$19.45
7.Louisiana Fiery Hot Pepper Sauce	32 - 8 oz bottles	\$21.05
8.Louisiana Hot Spiced Okra	24 - 8 oz jars	\$17.00
9.Northwoods Cranberry Sauce	12 - 12 oz jars	\$40.00
10.Original Frankfurter grüne Soße	12 boxes	\$13.00
11.Sirop d'éralbe	24 - 500 ml bottles	\$28.50
12.Vegie-spread	15 - 625 g jars	\$43.90

3.Confections		

4.Dairy Products		

5.Grains/Cereals		

6.Meat/Poultry		

7.Produce		

8.Seafood		

If work with bookmarks report is not required, you can disable this function. The **ShowBookmarksButton** option from the Toolbar section of options is sued for this. Set this option to false.

ASPX:

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

Razor:

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

In this case, report bookmarks will not be displayed, even if they are present in the displayed report.

When printing a report with bookmarks, the tree of bookmarks will be hidden. If, in addition to the report, you need to print bookmarks as well, you need to set the **BookmarksPrint** option from the Appearance section of options to true.

ASPX:

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

Razor:

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

8.11 Dynamic sorting and drill down reports

The **MVC Viewer** component supports dynamic sorting, collapsing and detailed reports. Dynamic sorting provides the ability to change the sorting direction in the rendered report. To do this, click the component in which was dynamic sorting is set. Dynamic sorting can be done in the following directions - **Ascending** and **Descending**. Each time you click the component the direction is reversed.

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

The screenshot shows a report titled "Interactive Sorting". At the top, there is a toolbar with icons for Print, Save, Send Email, and navigation buttons. Below the toolbar, the title "Interactive Sorting" is displayed in a large, bold font. A subtitle below it states, "The sample demonstrates how to use interactive sorting." The main content is a table with four columns: Company, Address, Phone, and Contact. The "Company" column has a dropdown arrow icon. The table lists 19 rows of data, each representing a different company with its address and phone number. The companies listed include: Alfreds Futterkiste, Ana Trujillo Emparedados y helados, Antonio Moreno Taquería, Around the Horn, Berglunds snabbköp, Blauer See Delikatessen, Blondel père et fils, Bólido Comidas preparadas, Bon app', Bottom-Dollar Markets, B's Beverages, Cactus Comidas para llevar, Centro comercial Moctezuma, Chop-suey Chinese, Comércio Mineiro, Consolidated Holdings, Die Wandernde Kuh, and Drachenblut Delikatessen.

Company ▾	Address	Phone	Contact
Alfreds Futterkiste	Obere Str. 57	030-0074321	Sales Repres...
Ana Trujillo Emparedados y helados	Avda. de la Constitución 2222	(5) 555-4729	Owner
Antonio Moreno Taquería	Mataderos 2312	(5) 555-3932	Owner
Around the Horn	120 Hanover Sq.	(171) 555-7788	Sales Repres...
Berglunds snabbköp	Berguvsvägen 8	0921-12 34 65	Order Admini...
Blauer See Delikatessen	Forsterstr. 57	0621-08460	Sales Repres...
Blondel père et fils	24, place Kléber	88.60.15.31	Marketing Ma...
Bólido Comidas preparadas	C/ Araquil, 67	(91) 555 22 82	Owner
Bon app'	12, rue des Bouchers	91.24.45.40	Owner
Bottom-Dollar Markets	23 Tsawwassen Blvd.	(604) 555-4729	Accounting M...
B's Beverages	Fauntleroy Circus	(171) 555-1212	Sales Repres...
Cactus Comidas para llevar	Cerrito 333	(1) 135-5555	Sales Agent
Centro comercial Moctezuma	Sierras de Granada 9993	(5) 555-3392	Marketing Ma...
Chop-suey Chinese	Hauptstr. 29	0452-076545	Owner
Comércio Mineiro	Av. dos Lusíadas, 23	(11) 555-7647	Sales Associ...
Consolidated Holdings	Berkeley Gardens 12 Brewery	(171) 555-2282	Sales Repres...
Die Wandernde Kuh	Adenauerallee 900	0711-020361	Sales Repres...
Drachenblut Delikatessen	Walserweg 21	0241-039123	Order Admini...

The report with dynamic collapsing is an interactive report that looks like blocks which can collapse/expand its contents by clicking on the block header. Report items that can expanded/collapsed, marked with a special icon sign -/+.

Print Save Send Email Page 1 of 4 100% One Page

Group with Collapsing

This sample demonstrates how to use a GroupHeaderBand with collapsing.

<input type="checkbox"/> A				
	Company	Address	Phone	Co
<input type="checkbox"/> B				
	Company	Address	Phone	Co
<input type="checkbox"/> C				
	Company	Address	Phone	Co
	Cactus Comidas para llevar	Cerrito 333	(1) 135-5555	Sales Agent
	Centro comercial Moctezuma	Sierras de Granada 9993	(5) 555-3392	Marketing Ma
	Chop-suey Chinese	Hauptstr. 29	0452-076545	Owner
	Comércio Mineiro	Av. dos Lusíadas, 23	(11) 555-7647	Sales Associ
	Consolidated Holdings	Berkeley Gardens 12 Brewery	(171) 555-2282	Sales Repres
<input type="checkbox"/> D				
	Company	Address	Phone	Co
	Die Wandernde Kuh	Arlenauerallee 900	0711-020261	Sales Repres

When you drill down data in your report you will see report tabs. The currently displayed report will be highlighted.

The screenshot shows a Stimulsoft Reports viewer interface. At the top, there are navigation icons for Print, Save, Send Email, and various page controls (Page 1 of 1). Below the toolbar, a breadcrumb navigation bar shows 'List of Products > Beverages > Dairy Products'. The main content area has a title 'List of Products in Category "Dairy Products"'. A table displays ten rows of product information:

Product	Supplier	Quantity Per Unit	Price	Unit
Camembert Pierrot	Gai pâturage	15 - 300 g rounds	34	
Fløtemysost	Norske Meierier	10 - 500 g pkgs.	21.5	
Geitost	Norske Meierier	500 g	2.5	
Gorgonzola Telino	Formaggi Fortini s.r.l.	12 - 100 g pkgs	12.5	
Gudbrandsdalsost	Norske Meierier	10 kg pkg.	36	
Mascarpone Fabioli	Formaggi Fortini s.r.l.	24 - 200 g pkgs.	32	
Mozzarella di Giovanni	Formaggi Fortini s.r.l.	24 - 200 g pkgs.	34.8	
Queso Cabrales	Cooperativa de Quesos 'Las Cabras'	1 kg pkg.	21	
Queso Manchego La Pastora	Cooperativa de Quesos 'Las Cabras'	10 - 500 g pkgs.	38	
Raclette Courdavault	Gai pâturage	5 kg pkg.	55	

No additional viewer settings required for the dynamic sorting, folding and drill down reports work properly. If you want to carry out any action before sorting, folding etc, you can define a special action - **Interaction**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Actions =
        {
            Interaction = "Interaction"
        }
}) %>
```

Razor:

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

Controller:

```
public ActionResult Interaction()
{
    // Some code before apply sorting, collapsing or drill down
    // ...
    return StiMvcViewer.InteractionResult();
}
```

8.12 Send Email

The **MVC Viewer** component provides the ability to send a report by Email. To activate this feature, you should set the **ShowSendEmailButton** option of the viewer to true and define the **EmailReport** action.

ASPX:

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

Razor:

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

```
    Toolbar =
    {
        ShowSendEmailButton = true
    }
})
```

Controller:

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

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

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

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

After selecting the report format to be sent by Email, the dialog displays the input options for sending such as Email recipient, subject and message.

Print Save Send Email Page 1 of 3 100% One Page

Simple List

The sample demonstrates how to create a simple list report.

	Company
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y helados
3	Antonio Moreno Taquería
4	Around the Horn
5	Berglunds snabbköp
6	Blauer See Delikatessen
7	Blondel père et fils
8	Bólido Comidas preparadas
9	Bon app'
10	Bottom-Dollar Markets
11	B's Beverages
12	Cactus Comidas para llevar
13	Centro comercial Moctezuma
14	Chop-suey Chinese
15	Comércio Mineiro
16	Consolidated Holdings
17	Drachenblut Delikatessen
18	Du monde entier

Email Options

Email:

Subject:

Message:

Attachment: Simple List.pdf

For sending a report by Email it is necessary to fill sending options such as address which will be used to send Email, as well as the sending server settings. If necessary, you can check and correct the form options to send Email specified in the viewer. The exported file will be automatically attached to the Email. Also it is allowed to set default values for the form to send Email. For this purpose you can use the following options.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Email =
        {
            DefaultEmailAddress = "recipient_address@gmail.com",
        }
    }
)%>
```

```

        DefaultEmailSubject = "New Invoice",
        DefaultEmailMessage = "Please check the new invoice in
        the attachment"
    }
}) %>

```

Razor:

```

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

```

8.13 Helpful Methods

There are several helpful methods to work with the report viewer. Applying the methods will be reviewed on the example of the Interaction action, but they can be used in any other actions of the MVC Viewer component. These methods are

GetReportObject()

Gets the report object with which the viewer is currently running. It is allowed to make necessary manipulations with it - register new data sets, modify report properties, assign parameters or load another report into the object. Then the report can be returned to the viewer by specifying it as a parameter in the resulting action method.

```

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

    return StiMvcViewer.InteractionResult(report);
}

```

GetRouteValues()

Returns the value for the URL with which the page viewer is open. So there is an opportunity to get a collection of page parameters for running the viewer in any action.

```

public ActionResult Interaction()

```

```

{
    RouteValueDictionary routeValues = StiMvcViewer.GetRouteValues();

    return StiMvcViewer.InteractionResult(report);
}

```

Also, the URL values can be obtained by the name of the parameter specifying it in the called action of the viewer

```

public ActionResult Interaction(string id)
{
    return StiMvcViewer.InteractionResult(report);
}

```

The difference between this method of obtaining the parameter values and the GetRouteValues() method is that the name of the controller and action will depend on the viewer settings and the action called.

GetFormValues()

Returns the value of the form, which initiates (opened by POST query) the viewer page. There is an opportunity to get a collection of form parameters in any action of the viewer.

```

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

    return StiMvcViewer.InteractionResult(report);
}

```

By default, this feature is disabled. To activate it, set the PassFormValues option to true.

ASPX:

```

<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Server =
        {
            PassFormValues = true
        }
    }) %>

```

Razor:

```

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

```

```

        Server =
    {
        PassFormValues = true
    }
})

```

GetViewerParameters()

Returns all the parameters required to prepare a report (view, export, print, interactive actions, etc.) passed by the report viewer to the server side. Returned parameters are read-only. They can be useful for determining the parameters of the action that is currently performing the viewer - for example, to determine the type of export.

```

public FileResult ExportReport()
{
    StiJavascriptParameters parameters =
    StiMvcViewer.GetViewerParameters();
    if (parameters.Options.ExportFormat == StiExportFormat.Pdf)
    {
        StiReport report = StiMvcViewer.GetReportObject();
        // Some action with report
    }

    return StiMvcViewer.ExportReportResult();
}

```

8.14 Viewer Settings

Setting the **StiMvcViewer** is performed with properties that are in the **StiMvcViewerOptions** class. Properties of this class can be divided into the following categories: Action, Working with the server, Interface, Toolbar, Buttons, Export. A sample of setting viewer options.

ASPX:

```

<%= Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Theme = StiTheme.Office2013WhiteTeal,
        Localization = "~/Content/Localization/en.xml",
        Actions =
        {
            GetReportSnapshot = "GetReportSnapshot",
            ViewerEvent = "ViewerEvent"
        },
        Appearance =
        {
            InterfaceType = StiInterfaceType.Auto,
            ScrollbarsMode = true,

```

```

        ShowTooltips = false
    },
    Exports =
    {
        DefaultSettings =
        {
            ExportToPdf =
            {
                CreatorString = "Company Name"
            }
        },
        ShowExportToDbf = false,
        ShowExportToDif = false
    }
} ) %>

```

Razor:

```

@Html.Stimulsoft().StiMvcViewer(
    "MvcViewer1", new StiMvcViewerOptions() {
        Theme = StiTheme.Office2013WhiteTeal,
        Localization = "~/Content/Localization/en.xml",
        Actions =
        {
            GetReportSnapshot = "GetReportSnapshot",
            ViewerEvent = "ViewerEvent"
        },
        Appearance =
        {
            InterfaceType = StiInterfaceType.Auto,
            ScrollbarsMode = true,
            ShowTooltips = false
        },
        Exports =
        {
            DefaultSettings =
            {
                ExportToPdf =
                {
                    CreatorString = "Company Name"
                }
            },
            ShowExportToDbf = false,
            ShowExportToDif = false
        }
    })

```

Name	Description
------	-------------

Theme	Changes the viewer theme. The list of available themes is located in the StiTheme class.
Localization	Sets the path to the XML file of the localization. The path can be absolute and relative.
Width	Specifies the width of the component to the required units that are defined in the Unit class. Values are available in pixels – Unit.Pixel() , points – Unit.Point() and percentage – Unit.Percentage() . By default, the width is 100%.
Height	Specifies the height of the component to the required units that are defined in the Unit class. Values are available in pixels – Unit.Pixel() , points – Unit.Point() and percentage – Unit.Percentage() . By default, the automatic height is set depending on the size of the report page, or 600 pixels in the display mode of the viewer with scroll bars.

8.14.1 Actions

This group includes properties that define the names of actions in the view controller, which will cause the client side report viewer using the appropriate functions.

Name	Description
GetReportSnapshot	Specifies the name of the action method of preparation of a rendered report. If the report caching is enabled, this action will be called only once during the first request of the report. This action is mandatory.
ViewerEvent	Specifies the name of the action method of major viewer events, such as flipping through pages of the report, scaling, change the display mode of pages, printing and exporting the report, working with parameters and interactivity. This action is mandatory.
PrintReport	Specifies the name of the action method of the report printing.
ExportReport	Specifies the name of the action method of the report export into the required format.
EmailReport	Specifies the name of the action method to send the report by Email.

Interaction	Specifies the name of the action method for working the viewer with interactive operations such as the use of parameters, dynamic sorting, drill down and detailing the report.
--------------------	---

8.14.2 Server

This group includes the properties that are responsible for the connecting parameters of the client and server sides.

Name	Description
Controller	Specifies the name of the controller for processing report viewer queries. If this option is not specified, then the query will be used by the current controller.
RequestTimeout	Waiting for the response from the server in seconds, after which an error is thrown. By default it is set to 20 seconds. For big reports it is recommended to increase this value.
CacheTimeout	Sets time in minutes the rendered report will be stored on the server since the last action of the viewer. By default value is set to 20 minutes.
CacheMode	Sets the cache mode of the report. It can have one of the following values: StiServerCacheMode.None – caching is disabled, the report will be loaded again using the GetReportSnapshot action. StiServerCacheMode.ObjectCache – the server cache is used as a repository (by default). The report is stored as an object. StiServerCacheMode.ObjectSession – the session is used as a repository. The report is stored as an object. StiServerCacheMode.StringCache – the server cache is used as a repository. The report is serialized into the packed string. StiServerCacheMode.StringSession – the session is used as a repository. The report is serialized into the packed string.
GlobalReportCache	Sets the mode of the common report cache for all clients, otherwise a separate object cache will be kept for each client. It must be disabled, if the same report uses different data bases. By default the cache is set to common.
CacheItemPriority	Sets the priority of the report stored in the server cache. This option affects the automatic cleaning of the server's memory in case of lack of it. The lower the priority is, the greater is the chance of removing information from memory.
UseRelativeUrls	Sets the viewer mode, in which the server uses relative links for AJAX requests. By default it is set to false .
PassFormValues	Enables sending values of the POST form to the client side, if these values are required to be used in the actions of the viewer. When you enable this

feature, the **GetFormValues()** method will return a collection of form parameters. By default it is set to **false**.

8.14.3 Appearance

This group includes properties that affect the appearance of the viewer.

Name	Description
BackgroundColor	Changes the color background of the viewer. By default it is set to Color.White .
PageBorderColor	Sets the border color of report pages. By default it is set to Color.Gray .
RightToLeft	Sets the Right to Left mode for viewer controls. By default it is set to false .
FullScreenMode	Sets the full screen mode of the report viewer. By default it is set to false .
ScrollbarsMode	Sets the mode of displaying the report viewer without scroll bars or with them. By default it is set to false .
OpenLinksTarget	Target window to open the link contained in the report. By default it is set to " _self " (current window).
OpenExportedReport Target	Target window to open the export file from the viewer. By default it is set to " _blank " (new window).
ShowToolips	Displays tips for the viewer controls when hovering the mouse. By default it is set to true .
PageAlignment	Sets the position of the report page in the viewer window. StiContentAlignment.Left – the page is aligned by the left side of the viewer. StiContentAlignment.Center – the page is aligned by the center (by default); StiContentAlignment.Right – the page will be aligned by the right side of the viewer.
ShowPageShadow	Hides the report page shadow. By default it is set to true .
BookmarksPrint	Prints report bookmarks on the printer (in addition to the report). By default it is set to false .
BookmarksTreeWidth	Sets the width of the bookmark panel in pixels. By default the width is 180 pixels.
ParametersPanelMaxHeight	Sets the maximum height of the parameter panel in pixels. By default the maximum height is set to 300 pixels.
ParametersPanelColumnsCount	Sets the number of columns for showing report parameters. By default partition is made on 2 columns.
ParametersPanelDateFormat	Sets the date format and time for variables of the corresponding type on the

eFormat	parameters panel. By default the date and time format set by the browser is used.
InterfaceType	Sets the type of the interface viewer. Can take the following values: StiInterfaceType.Auto – the interface type of the report viewer is selected automatically depending on the device used (by default). StiInterfaceType.Mouse – forced use of the interface to manage viewer using the mouse device. StiInterfaceType.Touch – force the use of the touch interface to manage viewer using the touch screen (mobile devices).
ChartRenderType	Sets the type of drawing charts on the report page. It has the following values: StiChartRenderType.AnimatedVector – charts are drawn in the vector mode with animation (default); StiChartRenderType.Vector – charts are drawn as vector images; StiChartRenderType.Image – charts are drawn as images.

8.14.4 Toolbar

This group includes properties that affect displaying the toolbar and its elements.

Name	Description
Visible	Shows/hides the toolbar of the report viewer. By default the option is set to true .
BackgroundColor	Changes the toolbar color. By default, the color palette of the selected theme is used.
BorderColor	Changes the color of the toolbar borders. By default, the color palette of the selected theme is used.
FontColor	Changes the font color for the toolbar and menu. By default, the color palette of the selected theme is used.
FontFamily	Changes the font family for the toolbar and menu. By default, the font of selected theme is used.
Alignment	Sets the position of the toolbar controls. StiContentAlignment.Default – the items are placed depending on the value of the RightToLeft option (by default); StiContentAlignment.Left – the items are aligned by the left side of the toolbar. StiContentAlignment.Center – the items are aligned by the center of the toolbar; StiContentAlignment.Right – the items are aligned by the right side of the toolbar.
ShowButtonCaptions	Shows/hides the buttons of the viewer toolbar. By default the option is set to true .

ShowPrintButton	Shows/hides the button Print on the toolbar of the report viewer. By default the option is set to true .
ShowSaveButton	Shows/hides the button Save on the toolbar of the report viewer. By default the option is set to true .
ShowSendEmailButton	Shows/hides the button Send Email on the toolbar of the report viewer. By default the option is set to false .
ShowBookmarksButton	Shows/hides the button Bookmarks on the toolbar of the report viewer. By default the option is set to true . If this button is hidden then the toolbar panel will not be shown even if bookmarks are present in the report.
ShowParametersButton	Shows/hides the button Parameters on the toolbar of the report viewer. By default the option is set to true . If this button is hidden then the toolbar panel will not be shown even if bookmarks are present in the report.
ShowEditorButton	Shows/hides the Editor button on the report viewer toolbar. By default, this option is set to true . If the report does not have editable components, the button will be hidden regardless of the property value.
ShowFullScreenButton	Shows/hides the button Full Screen on the toolbar of the report viewer. By default the option is set to true .
ShowFirstPageButton	Shows/hides the button First Page on the toolbar of the report viewer. By default the option is set to true .
ShowPreviousPageButton	Shows/hides the button Previous Page on the toolbar of the report viewer. By default the option is set to true .
ShowCurrentPageControl	Shows/hides the indicator of the current report page. By default the option is set to true .
ShowNextPageButton	Shows/hides the button Next Page on the toolbar of the report viewer. By default the option is set to true .
ShowLastPageButton	Shows/hides the button Last Page on the toolbar of the report viewer. By default the option is set to true .
ShowZoomButton	Shows/hides the button for selecting the report zoom. By default the option is set to true .

ShowViewModeButton	Shows/hides the button for selecting viewing modes of report pages. By default the option is set to true .
ShowDesignButton	Shows/hides the button Design on the toolbar of the report viewer. By default the option is set to false .
ShowAboutButton	Shows/hides the button About on the toolbar of the report viewer. By default the option is set to true .
PrintDestination	Sets the printing mode. StiPrintDestination.Default – the menu to choose the printing option will be shown (set by default). StiPrintDestination.Pdf – printing to the PDF file. StiPrintDestination.Direct – sending the report directly to the printer. The system printing dialog will be shown. StiPrintDestination.PopupWindow – printing the report will be done through the pop up window of preview.
ViewMode	Sets the mode of showing report pages. StiWebSizeMode.OnePage – shows one page of the report selected in the toolbar of the viewer (set by default); StiWebSizeMode.WholeReport – displays all report pages at once.
Zoom	Sets zoom of report pages. By default, it is 100 percent. Values vary from 10 to 500 percent. StiZoomMode.PageWidth – when you run the viewer the zoom will be set to display the report by the page width. StiZoomMode.PageHeight – when you run the viewer the zoom will be set to display the report by the page height.
MenuAnimation	Disables animation when showing/hiding the menu of the report viewer. By default the option is set to true .
ShowMenuMode	Sets the mode of showing the menu. StiShowMenuMode.Click – showing the menu by the mouse click (set by default); StiShowMenuMode.Hover – showing the menu by hovering the mouse.

8.14.5 Exporting Reports

With this group of properties, you can configure the menu to exporting reports.

Name	Description
DefaultSettings	Sets default export settings for each export type. These settings will be applied to dialog windows of exports when calling them or to reports in

	case of disabling showing dialog forms of these exports.
ShowExportDialog	Shows/hides the export parameters dialog. If the dialog window is disabled then exporting will be processed with the standard settings. It is enabled by default.
ShowExportToDocument	Shows/hides the menu item Document File . By default the option is set to true .
ShowExportToPdf	Shows/hides the menu item Adobe PDF File . By default the option is set to true .
ShowExportToXps	Shows/hides the menu item Microsoft XPS File . By default the option is set to true .
ShowExportToPoint	Shows/hides the menu item Microsoft PowerPoint 2007/2010 File . By default the option is set to true .
ShowExportToHtml	Shows/hides the menu item HMTL File . By default the option is set to true .
ShowExportToHtml5	Shows/hides the menu item HMTL5 File . By default the option is set to true .
ShowExportToMht	Shows/hides the menu item MHT Web Archive . By default the option is set to true .
ShowExportToText	Shows/hides the menu item Text File . By default the option is set to true .
ShowExportToRtf	Shows/hides the menu item Rich Text File . By default the option is set to true .
ShowExportToWord2007	Shows/hides the menu item Microsoft Word 2007/2010 File . By default the option is set to true .
ShowExportToOpenDocumentWriter	Shows/hides the menu item OpenDocument Writer File . By default the option is set to true .
ShowExportToExcel	Shows/hides the menu item Microsoft Excel File . By default the option is set to true .
ShowExportToExcelXml	Shows/hides the menu item Microsoft Excel Xml File . By default the option is set to true .

ShowExportToExcel2007	Shows/hides the menu item Microsoft Excel 2007/2010 File . By default the option is set to true .
ShowExportToOpenDocumentCalc	Shows/hides the menu item OpenDocument Calc File . By default the option is set to true .
ShowExportToCsv	Shows/hides the menu item CSV File . By default the option is set to true .
ShowExportToDbf	Shows/hides the menu item DBF File . By default the option is set to true .
ShowExportToXml	Shows/hides the menu item XML File . By default the option is set to true .
ShowExportToDif	Shows/hides the menu item Data Interchange Format (DIF) File . By default the option is set to true .
ShowExportToSylk	Shows/hides the menu item Symbolic Link (SYLK) File . By default the option is set to true .
ShowExportToImageBmp	Shows/hides the menu item BMP Image . By default the option is set to true .
ShowExportToImageGif	Shows/hides the menu item GIF Image . By default the option is set to true .
ShowExportToImageJpeg	Shows/hides the menu item JPEG Image . By default the option is set to true .
ShowExportToImagePcx	Shows/hides the menu item PCX Image . By default the option is set to true .
ShowExportToImagePng	Shows/hides the menu item PNG Image . By default the option is set to true .
ShowExportToImageTiff	Shows/hides the menu item TIFF Image . By default the option is set to true .
ShowExportToImageMetafile	Shows/hides the menu item Windows Metafile . By default the option is set to true .
ShowExportToImageSvg	Shows/hides the menu item Scalable Vector Graphics (SVG) File . By default the option is set to true .
ShowExportToImageSvgz	Shows/hides the menu item Compressed SVG (SVGZ) File . By default the option is set to true .

8.14.6 Sending Email

This group includes properties responsible for the default values for the send **Email** dialog form.

Name	Description
ShowEmailDialog	Shows/hides the dialog box options for sending the report by Email. If the dialog box is disabled, then sending by Email will be made by the settings set on the server side in the EmailReport action. By default the option is set to true .
ShowExportDialog	Shows/hides dialog boxes of export parameters when sending Email. If this option is disabled, the export will be carried out with the default settings. By default the option is set to true .
DefaultEmailAddress	Defines the recipient Email, the address which an email with an attached report will be sent to.
DefaultEmailSubject	Defines the e-mail subject, the value of this option will be a subject of this Email.
DefaultEmailMessage	Specifies a message (text) of the Email.

9 MVC ViewerFx

The **StiMvcViewerFx** component is used to edit reports in the window of a browser. And there is no need to install the .NET Framework, **ActiveX** components and other special plug-ins on the client machine. The only requirements are a web browser and the Flash player. Using **StiMvcViewerFx** it is possible to create, edit, save, view, and print reports on any computer, on any OS, where there is an internet connection, and where there is the installed web browser and **Flash Player 10.2** and higher. The **StiMvcViewerFx** component can be divided in two parts: client and server. The client side is the graphic wrapping of the designer that is realized on Flex. The server side is the report generator engine and, also, a module that has functions to get queries and send data on the client side. These two parts are collected into one DLL library and represented as a non visual **ASP.NET MVC** component.

The **MVC ViewerFx** component is located in the **StiMvcViewerFx** class. The minimum required set of libraries for components is

Stimulsoft.Base.dll
Stimulsoft.Report.dll
Stimulsoft.Report.Mvc.dll

9.1 How It Works?

To run the web report designer, it is required to put the **StiMvcViewerFx** component on the view page,

set the required properties, and define necessary actions in the view controller. When running the report viewer the following actions occur:

- The .NET component generates **HTML** and **Javascript** code for displaying and working the report viewer.
- When it is loaded, the **Flash** application is run. It requests the whole report on the server side and displays it.
- Interactive actions, such as exporting reports, applying parameters, sorting and drill down of reports, cause a specific action on the server side, where you can perform the necessary manipulations with the report.

9.2 How to Show Report?

In order to display reports, you must add the **MVC ViewerFx** component on a page and set the minimum required options to it. In the view controller you should determine appropriate actions. The **GetReportSnapshot** action returns the prepared report to the report viewer.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Actions =
    {
        GetReportSnapshot = "GetReportSnapshot"
    }
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Actions =
    {
        GetReportSnapshot = "GetReportSnapshot"
    }
})
```

Controller:

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

    return StiMvcViewerFx.GetReportSnapshotResult(report);
}
```

If the report was not rendered before being displayed, the **MVC ViewerFx** component will build it automatically. Thus, to display the report it is allowed to use report templates, rendered reports and reports as classes.

Controller:

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

    return StiMvcViewerFx.GetReportSnapshotResult(report);
}
```

Controller:

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

    return StiMvcViewerFx.GetReportSnapshotResult(report);
}
```

Simple List

The sample demonstrates how to create a simple list report.

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

◀ | Page 1 of 3 | ▶ ■

Also, before displaying the report it is allowed to change its properties, assigning parameters and registering data sets. For example:

Controller:

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

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

```

        report.RegData("Demo", ds);

        return StiMvcViewerFx.GetReportSnapshotResult(report);
    }
}

```

9.3 Localizing Component

The special **Localization** option is used in order to localize the interface of the report viewer to the required language. The value of this option will be a path (relative or absolute) to the XML localization file.

ASPX:

```

<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Localization = "~/Content/Localization/en.xml"
}) %>

```

Razor:

```

@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Localization = "~/Content/Localization/en.xml"
})

```

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

9.4 Using Themes

The **MVC ViewerFx** component has the ability to change the appearance of visual controls. The **Theme** option is used to change the theme of the report viewer. The option can be one of the enumeration values of **StiThemeFx**.

ASPX:

```

<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Theme = StiThemeFx.Office2013
}) %>

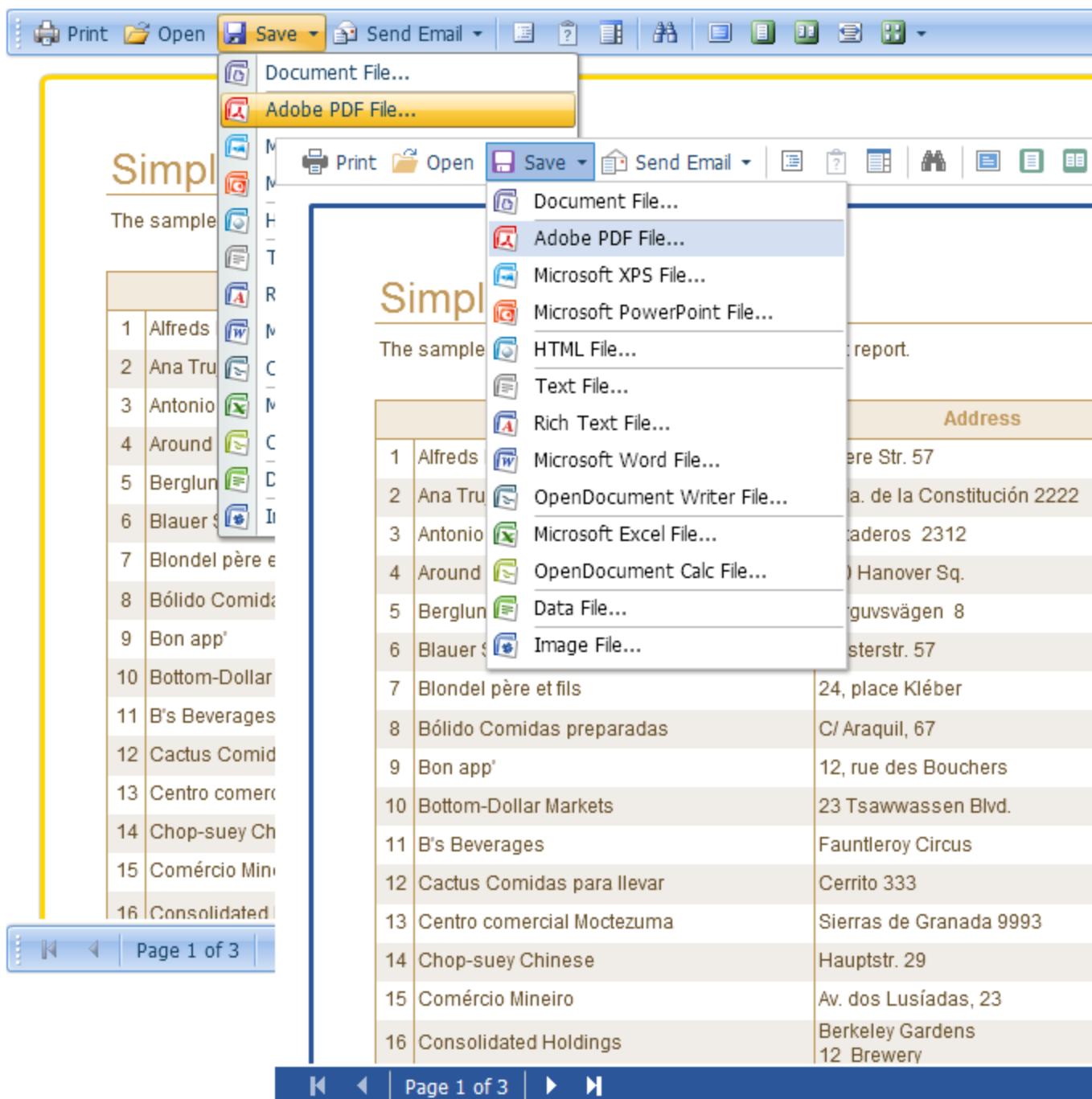
```

Razor:

```

@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Theme = StiThemeFx.Office2013
})

```



9.5 Exporting Reports

The **MVC ViewerFx** component allows you to export the displayed report to three dozen different formats such as PDF, Microsoft Word, Excel, XPS, RTF, image formats and many others.

No additional configuration is required for the viewer to operate the export function. If you want to carry out any action before exporting the report, you can define a special action `ExportReport`.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {  
    Actions =  
    {  
        ExportReport = "ExportReport"  
    }  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {  
    Actions =  
    {  
        ExportReport = "ExportReport"  
    }  
})
```

Controller:

```
public FileResult ExportReport()  
{  
    // Some code before export  
    // ...  
    return StiMvcViewerFx.ExportReportResult();  
}
```

The screenshot shows the MVC ViewerFx application interface. On the left, there is a report titled "Simple List" with the sub-instruction "The sample demonstrates how to create". Below this is a table with 16 rows, each containing a number and a company name. The first row is highlighted in orange. The table has a header row labeled "Company". The companies listed are: 1. Alfreds Futterkiste, 2. Ana Trujillo Emparedados y helados, 3. Antonio Moreno Taquería, 4. Around the Horn, 5. Berglunds snabbköp, 6. Blauer See Delikatessen, 7. Blondel père et fils, 8. Bólido Comidas preparadas, 9. Bon app', 10. Bottom-Dollar Markets, 11. B's Beverages, 12. Cactus Comidas para llevar, 13. Centro comercial Moctezuma, 14. Chop-suey Chinese, 15. Comércio Mineiro, and 16. Consolidated Holdings.

On the right, a modal dialog titled "Export Settings" is open. It contains two sections: "Page Range" and "Settings". In the "Page Range" section, there are three radio buttons: "All" (selected), "Current Page", and "Pages:" followed by an input field. A note below says "Enter page number and/or pages ranges separated by commas. For example: 1, 3, 5-12". In the "Settings" section, there are four dropdown menus: "Type" (set to "Html"), "Image Format" (set to "Jpeg"), "Scale" (set to "100%"), and "Export Mode" (set to "Table"). There are also four checkboxes: "Compress to Archive", "Embedded Image Data", "Add Page Breaks", and "Open After Export". At the bottom of the dialog are "OK" and "Cancel" buttons. The footer of the application displays the text "Berkeley Gardens 12 Brewery (171) 555-2282".

9.6 Interactive Reports

The **MVC ViewerFx** component supports working with interactive reports, such as reports with parameters, dynamic sorting and drill down. For adding a parameter to the panel you should define the variable requested by the user. When viewing a report in the viewer such a variable is automatically added to the panel of parameters. All types of report variables (normal variables, date and time, borders, lists, etc.) are supported. Dynamic sorting provides the ability to change the sorting direction in the rendered report. Reports with drill down below the main panel of the viewer will be displayed. The currently displayed report will be highlighted.

Interactive reports do not require additional configuration of the viewer. If it is necessary to apply report settings, do some sorting or drill down, the **GetReportSnapshot** action will be called. In this

action the required report template will be loaded again. The parameters passed from the client side will be automatically applied to the report.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {  
    Actions =  
    {  
        GetReportSnapshot = "GetReportSnapshot"  
    }  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {  
    Actions =  
    {  
        GetReportSnapshot = "GetReportSnapshot"  
    }  
})
```

Controller:

```
public ActionResult GetReportSnapshot()  
{  
    StiReport report = new StiReport();  
    report.Load(Server.MapPath("~/Content/ReportWithParameters.mrt"));  
  
    return StiMvcViewerFx.GetReportSnapshotResult(report);  
}
```

List of Products in Category "Condiments"

Product	Supplier	Quantity Per Unit	Price
Aniseed Syrup	Exotic Liquids	12 - 550 ml bottles	
Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	48 - 6 oz jars	
Chef Anton's Gumbo Mix	New Orleans Cajun Delights	36 boxes	
Genen Shouyu	Mayumi's	24 - 250 ml bottles	
Grandma's Boysenberry Spread	Grandma Kelly's Homestead	12 - 8 oz jars	
Gula Malacca	Leka Trading	20 - 2 kg bags	
Louisiana Fiery Hot Pepper Sauce	New Orleans Cajun Delights	32 - 8 oz bottles	
Louisiana Hot Spiced Okra	New Orleans Cajun Delights	24 - 8 oz jars	
Northwoods Cranberry Sauce	Grandma Kelly's Homestead	12 - 12 oz jars	
Original Frankfurter grüne Soße	Plutzer Lebensmittelgroßmärkte AG	12 boxes	
Sirop d'éable	Forêts d'éables	24 - 500 ml bottles	
Vegie-spread	Pavlova, Ltd.	15 - 625 g jars	

◀ ▶ | Page 1 of 1 | ▶ ▷

9.7 Send Email

The **MVC ViewerFx** component provides the ability to send a report by Email. Set the ShowSendEmailButton option of the viewer to true and define the EmailReport action to activate this ability.

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Actions =
    {
```

```

        EmailReport = "EmailReport"
    },
    Toolbar =
    {
        ShowSendEmailButton = true
    }
}) %>

```

Razor:

```

@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Actions =
    {
        EmailReport = "EmailReport"
    },
    Toolbar =
    {
        ShowSendEmailButton = true
    }
})

```

Controller:

```

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

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

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

    return StiMvcViewerFx.EmailReportResult(options);
}

```

After selecting the report format to be sent by Email, the opened dialog displays the input parameters for sending Email such as recipient, subject and message text.

The screenshot shows the MVC ViewerFx application interface. At the top, there is a toolbar with various icons for printing, opening files, saving, sending emails, and navigating. Below the toolbar, the main content area has a title "List of Products" and a subtitle "The sample demonstrates how to use drill-down reports." To the left, there is a vertical list of categories: 1. Beverages, 2. Condiments, 3. Confections, 4. Dairy Products, 5. Grains/Cereals, 6. Meat/Poultry, 7. Produce, and 8. Seafood. Overlaid on the right side of the page is a modal dialog titled "Email Options". The dialog contains fields for "Email" (recipient_address@gmail.com), "Subject" (New Products), and "Message" (Please check the new list of products in the attachment). It also shows the "Attachment" as "List of Products.pdf" and has a checked checkbox for "Save form values". At the bottom right of the dialog are "OK" and "Cancel" buttons.

Fill the sending options such as an Email address and server settings. If necessary, you can check and adjust the options of the form to send Email, specified in the viewer. The exported file will be automatically attached to the letter.

9.8 Additional Methods

There are several additional methods for working with the report viewer. The methods of the **MVC ViewerFx** component are following:

GetReportObject()

Returns an object of the report, with which the viewer is currently working. It is allowed to make necessary manipulations with it - register new data sets, change report properties, assign parameters

or load an object into another report.

```
public ActionResult ExportReport()
{
    StiReport report = StiMvcViewerFx.GetReportObject();
    report.ReportName = "MyReportName";

    return StiMvcViewerFx.ExportReportResult(report);
}
```

GetRouteValues()

Returns values for the **URL** with which the viewer page is opened. Thus, it is possible to obtain a collection of parameters of the running page in any action of the viewer.

```
public ActionResult ExportReport()
{
    RouteValueDictionary routeValues = StiMvcViewerFx.GetRouteValues();

    return StiMvcViewerFx.ExportReportResult(report);
}
```

Also, **URL** values can be obtained by the name of the parameter specifying it in the called action of the viewer.

```
public ActionResult ExportReport(string id)
{
    return StiMvcViewerFx.ExportReportResult(report);
}
```

The difference between this method of obtaining the parameter of values and the additional method **GetRouteValues()** is that the name of the controller and action will depend on the settings of the viewer and the called action.

9.9 Viewer Settings

Configuring the **MVC ViewerFx** component is performed using the options that are located in the **StiMvcViewerFxOptions** class. Options are divided into eight groups - no groups, Actions, Server, Appearance, Toolbar, Export, Email, Print. Here is an example below of setting options viewer:

ASPX:

```
<%= Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Theme = StiTheme.Office2013,
    Localization = "~/Content/Localization/en.xml",
    Actions =
    {
        GetReportSnapshot = "GetReportSnapshot",
    }
}) %>
```

```

        ExportReport = "ExportReport"
    },
    Exports =
    {
        ShowExportDialog = true,
        ShowExportToDbf = false,
        ShowExportToDif = false
    },
    Print =
    {
        AutoPageOrientation = true,
        AutoPageScale = true
    }
}) %>

```

Razor:

```

@Html.Stimulsoft().StiMvcViewerFx(new StiMvcViewerFxOptions() {
    Theme = StiTheme.Office2013,
    Localization = "~/Content/Localization/en.xml",
    Actions =
    {
        GetReportSnapshot = "GetReportSnapshot",
        ExportReport = "ExportReport"
    },
    Exports =
    {
        ShowExportDialog = true,
        ShowExportToDbf = false,
        ShowExportToDif = false
    },
    Print =
    {
        AutoPageOrientation = true,
        AutoPageScale = true
    }
})

```

Name	Description
Theme	Changes the viewer theme. The list of available themes can found in the StiThemeFx class.

Localization	Sets the path to the XML file of the localization. The path can be absolute and relative.
Width	Specifies the width of the component to the required units that are defined in the Unit class. Values are available in pixels – Unit.Pixel() , points – Unit.Point() and percentage – Unit.Percentage() . By default, the width is 100%.
Height	Sets the height of the component to the required units that are defined in the Unit class. Values are available in pixels – Unit.Pixel() , points – Unit.Point() and percentage – Unit.Percentage() . By default, the automatic height is set to 600 pixels.

9.9.1 Actions

This group includes properties that define the name of actions in the view controller, which will call the client side report viewer using the appropriate functions.

Name	Description
GetReportSnapshot	Defines the action method name of preparing the rendered report.
GetLocalization	Defines the action method name of loading the XML file with the viewer localization. If this action is not defined the English localization is set by default.
ExportReport	Defines the action method name of exporting a report to the required format.
DesignReport	Defines the action method name to navigate to the desired view by pressing the Design button on the viewer panel.
EmailReport	Defines the action method name of sending the report by Email.
Exit	Defines the action method name to navigate to the desired view by pressing the Exit button on the viewer panel.

9.9.2 Server

This group includes properties that define parameters of working with server.

Name	Description
Controller	Specifies the name of the controller to process queries of the report viewer. If this option is not specified then the current controller will be used to process queries.
RequestTimeout	The waiting time of the report from the server in seconds after which an error is thrown. The default setting is 20 seconds. For big reports it is recommended to increase this value.
RepeatCount	Sets the number of repeats of the request in the event of any error of connection with the server. The default value is 1.
EnableDataLogger	Provides an opportunity to enable saving of all requests/responses in the viewer to a special log file that can be stored on a local disk. If there is any error in the report viewer, it is recommended to enable this option and analyze the log file.
UseRelativeUrls	Sets the mode of the viewer in which relative links are used for requests to the server. By default, this option is set to false .

9.9.3 Appearance

This group includes properties that define the viewer appearance.

Name	Description
BackgroundColor	Changes the color background of the viewer. By default it is set to Color.White .
CurrentPageBorderColor	Sets color of a report page border. By default it is set to Color.Gold (Color.Blue for the theme Office2013).
AutoHideScrollbars	Hides scrollbars in the viewer if the page does not entirely fit the page. By default, this property is set to false .
OpenLinksTarget	Target window to open links contained in the report. By default, this property is set to " _blank " (new window).
OpenExportedReportTarget	Target window to open an export file from the viewer. By default, this property is set to " _blank " (new windows).
ShowToolipsHelp	Showing a link to the online help in pop-up hints when hovering a mouse over viewer controls. By default, this property is set to true .
ShowFormsHelp	Showing links to the online help in the header of the viewer dialogs. By default, this property is set to true .

ShowFormsHint	Showing a hint when hovering over the interface items in dialogs of the viewer. By default, this property is set to true .
ParametersPanelColumnsCount	Sets the number of columns for showing report parameters. 2 columns are set by default.
ParametersPanelEditWidth	Sets the width of items of the parameters panel in pixels. By default it is set to 200 pixels.
ParametersPanelDateFormat	Sets the format of date and time for variables of the corresponding type on the panel of parameters. By default, the data and time format set on the server is used.
FlashWMode	Set the mode of showing a Flash application in the browser page. By default, this property is set to " window ".
AboutDialogTextLine1	Sets a text for the About box. The text will be shown in the first line of the box.
AboutDialogTextLine2	Sets a text for the About box. The text will be shown in the second line of the box.
AboutDialogUrl	Sets the URL to show it in the About box in the viewer.
AboutDialogUrlText	Set the text of the URL to show it in the About box.
ShowCancelButton	Shows/hides the Cancel button in the loading data window from the server side. By default, this property is set to false .

9.9.4 Toolbars

This group includes properties that are used to show or hide the viewer items.

Name	Description
Visible	Shows/hides the toolbar of the viewer. By default the option is set to true .
ShowNavigatePanel	Shows/hides the navigation panel of the viewer. By default the option is set to true .
ShowViewModePanel	Shows/hides the viewing mode panel of the viewer. By default the option is set to true .
ShowPrintButton	Shows/hides the button Print on the toolbar of the report viewer. By default the option is set to true .
ShowOpenButton	Shows/hides the button Open on the toolbar of the report viewer. By default the option is set to true .
ShowSaveButton	Shows/hides the button Save on the toolbar of the report viewer. By default the option is set to true .
ShowSendEmailButton	Shows/hides the button Send Email on the toolbar of the report viewer. By default the option is set to false .

ShowBookmarksButton	Shows/hides the button Bookmarks on the toolbar of the report viewer. By default the option is set to true .
ShowParametersButton	Shows/hides the button Parameters on the toolbar of the report viewer. By default the option is set to true .
ShowThumbnailsButton	Shows/hides the button Thumbnails on the toolbar of the report viewer. By default the option is set to true .
ShowFindButton	Shows/hides the button Find on the toolbar of the report viewer. By default the option is set to true .
ShowFullScreenButton	Shows/hides the button Full Screen on the toolbar of the report viewer. By default the option is set to true .
ShowExitButton	Shows/hides the button Exit on the toolbar of the report viewer. By default the option is set to true .
ShowAboutButton	Shows/hides the button About on the toolbar of the report viewer. By default the option is set to true .
ShowDesignButton	Shows/hides the button Design on the toolbar of the report viewer. By default the option is set to true .
ShowFirstPageButton	Shows/hides the button First Page on the navigation panel of the report viewer. By default the option is set to true .
ShowPreviousPageButton	Shows/hides the button Previous Page on the navigation panel of the report viewer. By default the option is set to true .
ShowGoToPageButton	Shows/hides the button Go To Page on the navigation panel of the report viewer. By default the option is set to true .
ShowNextPageButton	Shows/hides the button Next Page on the navigation panel of the report viewer. By default the option is set to true .
ShowLastPageButton	Shows/hides the button Last Page on the navigation panel of the report viewer. By default the option is set to true .
ShowSinglePageViewModeButton	Shows/hides the button Single Page on the viewing mode panel of the report viewer. By default the option is set to true .
ShowContinuousPageViewModeButton	Shows/hides the button Continuous Page on the viewing mode panel of the report viewer. By default the option is set to true .
ShowMultiplePageViewModeButton	Shows/hides the button Multiple Page on the viewing mode panel of the report viewer. By default the option is set to true .
ShowZoomButtons	Shows/hides the zoom buttons. By default the option is set to true .
Zoom	Sets zoom for report pages. By default, it is 100 percent. Values vary from 10 to 500 percent. The following values are available: StiZoomModeFx.Default – when you run the viewer the zoom that was used previously will be set (set by default). StiZoomModeFx.OnePage – when you run the viewer the zoom will be set to display a whole page of the report.

	<p>StiZoomModeFx.TwoPages – when you run the viewer the zoom will be set to display 2 pages of the report.</p> <p>StiZoomModeFx.PageWidth – when you run the viewer the zoom will be set to display the report by the page width.</p>
--	---

9.9.5 Exporting

With this group of properties one can modify exporting options.

Name	Description
ShowExportDialog	Shows/hides the exporting dialog. If the property is set to true, the dialog will be displayed. If it is set to false, then the dialog will be hidden. By default, this property is set to true .
ShowExportToDocument	Shows/hides the menu item Document File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToPdf	Shows/hides the menu item Adobe PDF File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToXps	Shows/hides the menu item Microsoft XPS File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToPowerPoint	Shows/hides the menu item Microsoft PowerPoint 2007/2010 File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToHtml	Shows/hides the menu item HMTL File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToHtml5	Shows/hides the menu item HMTL5 File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToMht	Shows/hides the menu item MHT Web Archive . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToText	Shows/hides the menu item Text File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToRtf	Shows/hides the menu item Rich Text File . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .

ShowExportToWord2007	Shows/hides the menu item Microsoft Word 2007/2010 File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToOpenDocumentWriter	Shows/hides the menu item OpenDocument Writer File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToExcel	Shows/hides the menu item Microsoft Excel File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToExcelXml	Shows/hides the menu item Microsoft Excel Xml File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToExcel2007	Shows/hides the menu item Microsoft Excel 2007/2010 File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToOpenDocumentCalc	Shows/hides the menu item OpenDocument Calc File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToCsv	Shows/hides the menu item CSV File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToDbf	Shows/hides the menu item DBF File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToXml	Shows/hides the menu item XML File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true.
ShowExportToDif	Shows/hides the menu item Data Interchange Format (DIF) File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToSylk	Shows/hides the menu item Symbolic Link (SYLK) File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImageBmp	Shows/hides the menu item BMP Image. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImageGif	Shows/hides the menu item GIF Image. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImageJpeg	Shows/hides the menu item JPEG Image. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be

	hidden. By default, this property is set to true .
ShowExportToImage Pcx	Shows/hides the menu item PCX Image. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImage Png	Shows/hides the menu item PNG Image. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImage Tiff	Shows/hides the menu item TIFF Image. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImage Metafile	Shows/hides the menu item Windows Metafile . If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImage Svg	Shows/hides the menu item Scalable Vector Graphics (SVG) File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .
ShowExportToImage Svgz	Shows/hides the menu item Compressed SVG (SVGZ) File. If the property is set to true, the menu item will be displayed. If it is set to false, then the menu item will be hidden. By default, this property is set to true .

9.9.6 Send Email

With this group of properties one can modify options to send Email.

Name	Description
ShowEmailDialog	Shows/hides the dialog window of sending reports by Email. If the option is disabled then sending by Email will be done with settings set on the server side in the EmailReport action. By default, this property is set to true .
ShowExportDialog	Hides/shows dialog windows of exports when sending a report by Email. If this option is disabled then exporting will be done with settings set by default. By default, this property is set to true .
DefaultEmailAddress	Defines the Email of a recipient.

9.9.7 Printing

With this group of properties one can modify printing options.

Name	Description
ShowPrintDialog	Shows/hides the dialog box of printing parameters. If the dialog is hidden

	then printing will be done by default settings. By default the option is set to true .
AutoPageOrientation	Enables automatic page orientation when printing a report, if the incorrect printing format is set. By default the option is set to true .
AutoPageScale	Enables automatic scaling when printing the report, if incorrect size is set in the printer settings. By default the option is set to true .
AllowDefaultPrint	Shows/hides the printing item Default in the printing dialog. By default the option is set to true .
AllowPrintToPdf	Shows/hides the printing item Print as PDF in the printing dialog. By default the option is set to true .
AllowPrintToHtml	Shows/hides the printing item Print as HTML in the printing dialog. By default the option is set to true .

10 Web ViewerFx

The **StiWebViewerFx** component is delivered in **Stimulsoft Reports.Web**. This component is used to show reports in a web browser.

10.1 How to Show Report?

Put the **StiWebViewerFx** component on a web page. Then you need to use the following code to show a report:

C#:

```
StiReport report = new StiReport();
report.Load("report.mrt");
StiWebViewerFx1.Report = report;
```

VB.NET:

```
Dim Report As StiReport = New StiReport()
Report.Load("report.mrt")
StiWebViewerFx1.Report = Report
```

If the report was not rendered before showing, then the WebViewerFx component renders it automatically. Loading a report after loading the viewer is done using the **OnGetReport** event. After subscription to this event, it will occur each time when the viewer needs a report. In other words, after loading **WebViewerFx** requests the report from the server and, if the subscription to the **OnGetReport** event is done, then, in this event, the report can be assigned to the designer. Below is a code example, using the **OnGetReport** event:

```
protected void StiWebViewerFx1_GetReport(object sender,
StiWebViewerFx.StiOnGetReportEventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    e.Report = report;
}
```

If the report has not been rendered before being displayed, the **WebViewerFx** component renders it automatically.

If it is required to show **WebViewerFx** to the entire browser window, you can use the following method:

```
StiReport report = new StiReport();
report.Load("report.mrt");
StiWebViewerFx1.View(report);
```

In this case, in the entire region in the current browser window a viewer will be displayed. All other items are located on the .aspx page will not be displayed.

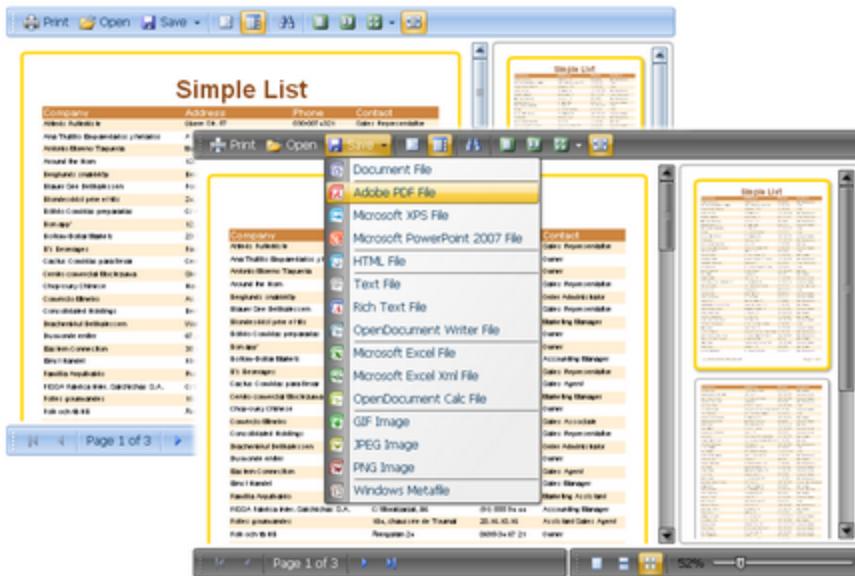
10.2 Localization of StiWebViewerFx Component

To make the StiWebViewerFx component to "speak" another language it is necessary to copy a localization file of the standard delivery to the server to "Localization" folder. For example, select the "de.xml" file. Then define the name of default localization:

```
<ccl:StiWebViewerFx ID="StiWebViewerFx1" runat="server"
Localization="en" />
```

10.3 Using Themes in WebViewerFx

The **StiWebViewerFx** component has the ability to change the themes of the viewer. The theme can be changed using the **ThemeName** property. For example, **ThemeName="Black"**.



10.4 WebViewerFx Settings

Setting the **WebViewerFx** can be done using the static properties, which are described in the **StiWebViewerFxOptions**. class. Static properties of the **WebViewerFx** can be divided into following groups: Connection, Zooming and Toolbar.

10.4.1 Connection

The static properties described below belong to the **StiWebViewerFxOptions.Connection** group and responsible for connecting the client and server sides:

Name	Description
ClientRequestTimeout	property sets time (in seconds) that the client part will wait the response from the server side. The default value is 10 seconds.
ClientRepeatCount	property sets the number of repeats of requests of the server side to the client side, when getting errors of obtaining data. The default value is 2 repeats.
RelativeUrls	property allows using the relative Url . If the RelativeUrls is set to false , then the absolute Url is used. If the RelativeUrls is set to true , then the relative Url is used. By default, the value is set to false . A sample of the absolute and relative Urls is shown below: http://localhost:4444/WebViewerDemo/WebViewer.aspx is an absolute Url , the RelativeUrls property is set to false ; /WebViewerDemo/WebViewer.aspx is a relative Url , the RelativeUrls property is set to true .

10.4.2 Zooming

The group of static properties **Zoom** has one **StiZoomMode** static property. Depending on the values of this property it is possible to set report template zoom. This property has the following values: **PageWidth**, **PageHeight**, **OnePage**, **Zoom25**, **Zoom50**, **Zoom75**, **Zoom100**, **Zoom150**, **Zoom200**.

Name	Description
Default	Value sets previously saved report zoom in WebViewerFx . So, if a report was saved with 37% zoom then, when opening it the next time, the report is shown in 37% zoom.
PageWidth	Value sets zoom by Page Width . So the width of the report template matches the width of the web viewer window.
OnePage	Value sets zoom by One Page . So the entire page of the report template fits in the window of the web viewer.
TwoPages	Value sets zoom by Two Pages . So two report pages will be shown by width and height in the WebViewerFx .
Zoom25 , Zoom50 , Zoom75 , Zoom100 , Zoom150 , Zoom200	Value sets zoom level of the report template which is 25% , 50% , 75% , 100% , 150% , 200% .

10.4.3 Viewer Static Properties

A group of **StiWebViewerFxOptions.Toolbar** static properties of the **WebViewerFx** is described below:

Name	Description
ShowZoom	Property is used to show/hide the zoom panel. If the ShowZoom property is set to true , then the zoom panel will be shown. If the ShowZoom property is set to false , then the zoom panel will be hidden. By default this property is set to true .
ShowPrintButton	Property is used to show/hide the Print button. If the ShowPrintButton property is set to true , then the Print button is shown. If the ShowPrintButton property is set to true , then the Print button is hidden. By default this property is set to true .
ShowOpenButton	Property is used to show/hide the Open button. If the ShowOpenButton property is set to true , then the Open button is shown. If the ShowOpenButton property is set to true , then the Open button is hidden. By default this property is set to true .
ShowSaveButton	Property is used to show/hide the Save button. If the ShowSaveButton

	property is set to true , then the Save button is shown. If the ShowSaveButton property is set to true , then the Save button is hidden. By default this property is set to true .
ShowSendEMailButton	Property is used to show/hide the SendEMail button. If the ShowSendEMailButton property is set to true , then the SendEMail button is shown. If the ShowSendEMailButton property is set to true , then the SendEMail button is hidden. By default this property is set to true .
ShowPageNewButton	Property is used to show/hide the Page New button. If the ShowPageNewButton property is set to true , then the Page New button is shown. If the ShowPageNewButton property is set to true , then the Page New button is hidden. By default this property is set to true .
ShowPageDeleteButton	Property is used to show/hide the Page Delete button. If the ShowPageDeleteButton property is set to true , then the Page Delete button is shown. If the ShowPageDeleteButton property is set to true , then the Page Delete button is hidden. By default this property is set to true .
ShowPageSizeButton	Property is used to show/hide the Page Size button. If the ShowPageSizeButton property is set to true , then the Page Size button is shown. If the ShowPageSizeButton property is set to true , then the Page Size button is hidden. By default this property is set to true .
ShowBookmarksButton	Property is used to show/hide the Bookmarks button. If the ShowBookmarksButton property is set to true , then the Bookmarks button is shown. If the ShowBookmarksButton property is set to true , then the Bookmarks button is hidden. By default this property is set to true .
ShowThumbnailsButton	Property is used to show/hide the Thumbnails button. If the ShowThumbnailsButton property is set to true , then the Thumbnails button is shown. If the ShowThumbnailsButton property is set to true , then the Thumbnails button is hidden. By default this property is set to true .
ShowFindButton	Property is used to show/hide the Find button. If the ShowFindButton property is set to true , then the Find button is shown. If the ShowFindButton property is set to true , then the Find button is hidden. By default this property is set to true .
ShowEditButton	Property is used to show/hide the Edit button. If the ShowEditButton property is set to true , then the Edit button is shown. If the ShowEditButton property is set to true , then the Edit button is hidden. By default this property is set to true .
ShowFirstPageButton	Property is used to show/hide the First Page button. If the ShowFirstPageButton property is set to true , then the First Page button is shown. If the ShowFirstPageButton property is set to true , then the First Page button is hidden. By default this property is set to true .
ShowPreviousPageBut	Property is used to show/hide the Previous Page button. If the

ton	ShowPreviousPageButton property is set to true , then the Previous Page button is shown. If the ShowPreviousPageButton property is set to true , then the Previous Page button is hidden. By default this property is set to true .
ShowGoToPageButton	Property is used to show/hide the Go to Page button. If the ShowGoToPageButton property is set to true , then the Go to Page button is shown. If the ShowGoToPageButton property is set to true , then the Go to Page button is hidden. By default this property is set to true .
ShowNextPageButton	Property is used to show/hide the Next Page button. If the ShowNextPageButton property is set to true , then the Next Page button is shown. If the ShowNextPageButton property is set to true , then the Next Page button is hidden. By default this property is set to true .
ShowLastPageButton	Property is used to show/hide the Last Page button. If the ShowLastPageButton property is set to true , then the Last Page button is shown. If the ShowLastPageButton property is set to true , then the Last Page button is hidden. By default this property is set to true .
ShowPageViewModeSingleButton	Property is used to show/hide the Single Page button. If the ShowPageViewModeSingleButton property is set to true , then the Single Page button is shown. If the ShowPageViewModeSingleButton property is set to true , then the Single Page button is hidden. By default this property is set to true .
ShowPageViewModeContinuousButton	Property is used to show/hide the Continuous button. If the ShowPageViewModeContinuousButton property is set to true , then the Continuous button is shown. If the ShowPageViewModeContinuousButton property is set to true , then the Continuous button is hidden. By default this property is set to true .
ShowPageViewModeMultipleButton	Property is used to show/hide the Multiple Pages button. If the ShowPageViewModeMultipleButton property is set to true , then the Multiple Pages button is shown. If the ShowPageViewModeMultipleButton property is set to true , then the Multiple Pages button is hidden. By default this property is set to true .

10.5 Properties

The properties of **WebViewerFx** are described below:

- ▶ The **ServerTimeout** property is used to define time of storing a report in the server cache. By default, this property is set to "**00:10:00**", this means that the report is stored **10** minutes in the server cache and then it is removed.
- ▶ The **Background** property is used to change the background color. By default, this property is set to **White**, this means that the background color is white. It is also possible to set any color in the **#rrggbb** format and transparent color.
- ▶ The **DataEncryption** property is used to enable/disable data encryption. If the **DataEncryption** property is set to **false**, then data are not encrypted. If the **DataEncryption** property is set to **true**, then

data are encrypted. By default, this property is set to **false**.

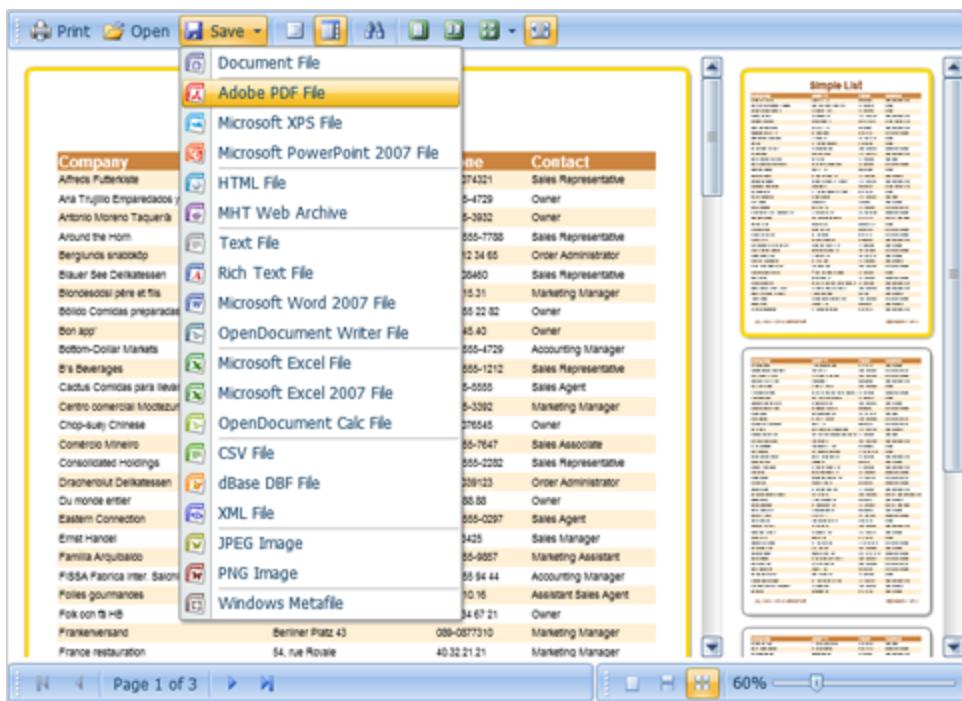
- ▷ The **DataCompression** property is used to enable/disable data compression. If the **DataCompression** property is set to **false**, then data are not compressed. If the **DataCompression** property is set to **true**, then data are compressed. By default, this property is set to **true**.
- ▷ The **AppCacheDirectory** property is used to indicate the path to the directory on the server, to what file caching of the **Flash**-application will occur. For this you need to set full access of the **ASP.NET** application to this folder.
- ▷ The **LocalizationDirectory** property is used to specify the path to the folder where localization **.xml** files are stored. The folder should be placed in the root directory of the project. A code sample for specifying the path to the folder with localization files is shown below (for example, the **Languages** folder):

```
<ccl:StiWebDesigner ID="StiWebViewerFx1" runat="server"
DirectoryLocalization="Files/Languages" />
```

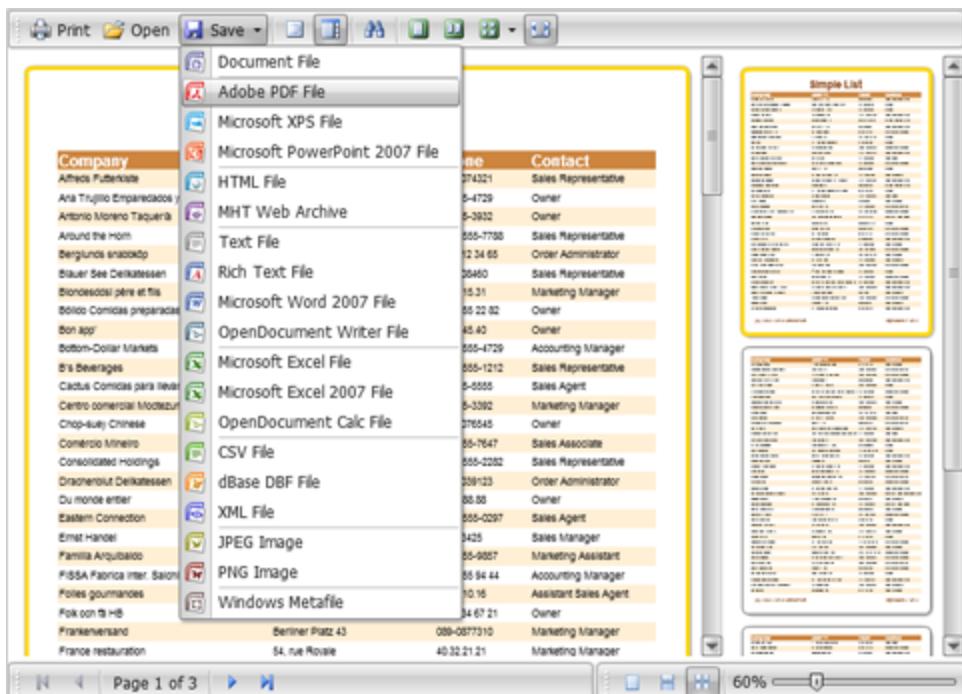
- ▷ The **Localization** property is used to specify the ability to localize the **WebViewerFx** UI in any of 26 languages available. The **Localization** property should be set to the value. The value is the **.xml** file in the **Localization** folder of the root directory in the project). By default, this property is set to "**en**", this means that the UI is localized in English. A code sample for setting the **Localization** property of the **WebViewerFx** UI to **English** language ("**en**") is shown below:

```
<ccl:StiWebDesigner ID="StiWebViewerFx1" runat="server"
Localization="ru" />
```

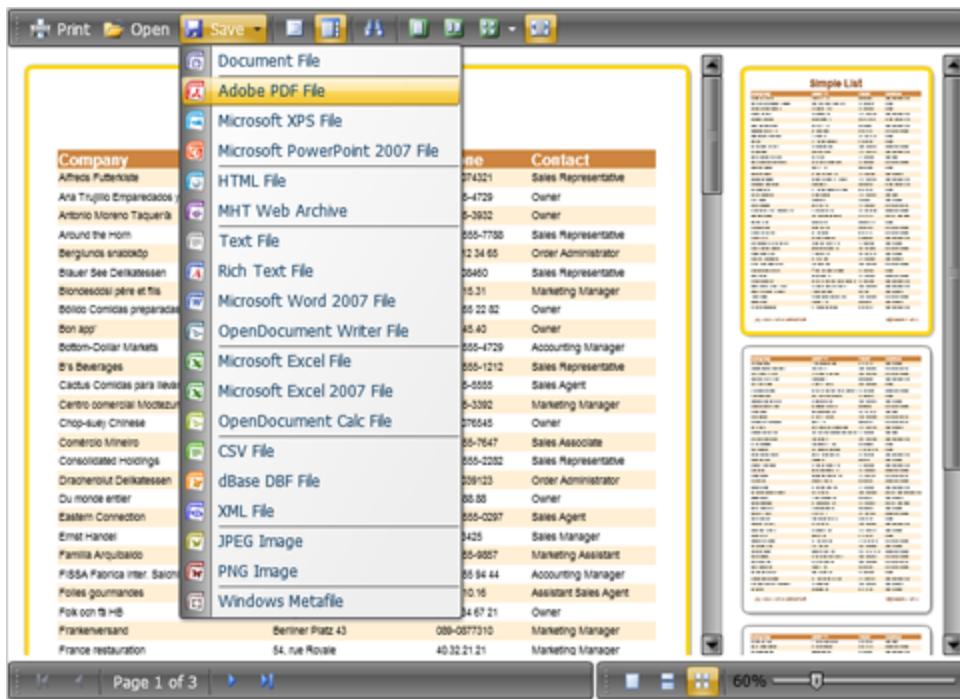
- ▷ The **ImageQuality** property is used to change the quality of images in the report. depending on the value of this property it is possible to change the image file size and image quality. If the **ImageQuality** property is set to **Low**, then the file size and quality will be low. If the **ImageQuality** property is set to **Normal**, then the file size and quality will have optimal ratio between size and quality. If the **ImageQuality** property is set to **High**, then the file size and quality will be the highest.
- ▷ The **ThemeName** property is used to change the style the theme of the **WebViewerFx**. If the **ThemeName** property is set to **Blue**, then the style of the viewer will be as shown on the picture below:



If the **ThemeName** property is set to **Silver**, then the style of the viewer will look as shown on the picture below:



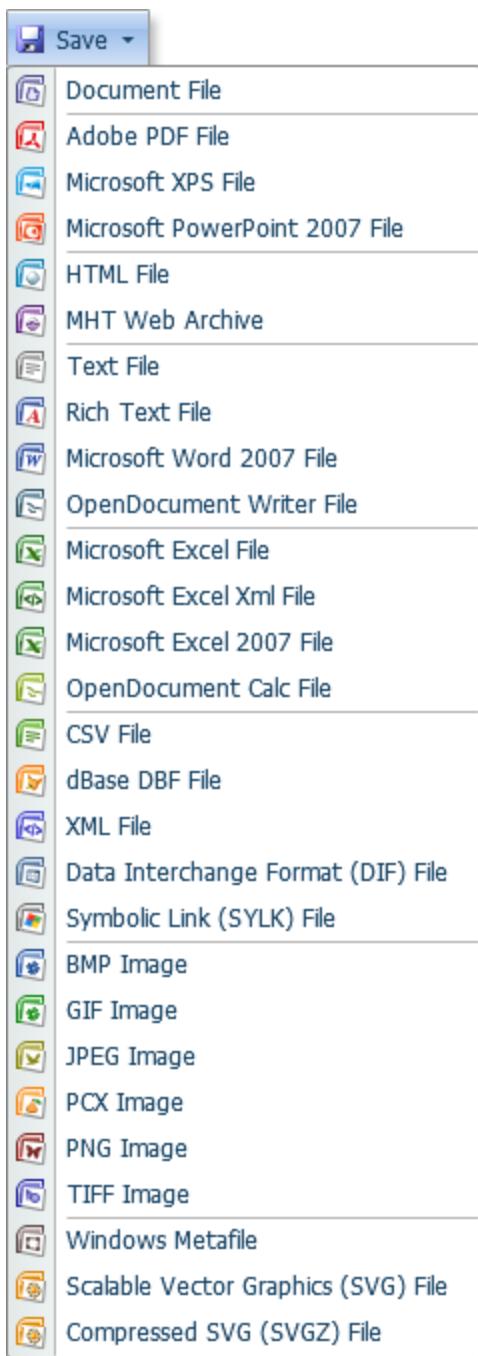
If the **ThemeName** property is set to **Black**, then the style of the viewer will look as shown on the picture below:



10.6 Export Settings

It is possible to customize a list of export formats. In other words, it is possible to hide unused export formats. Customization of the export formats list can be done by using the WebViewerFx properties.

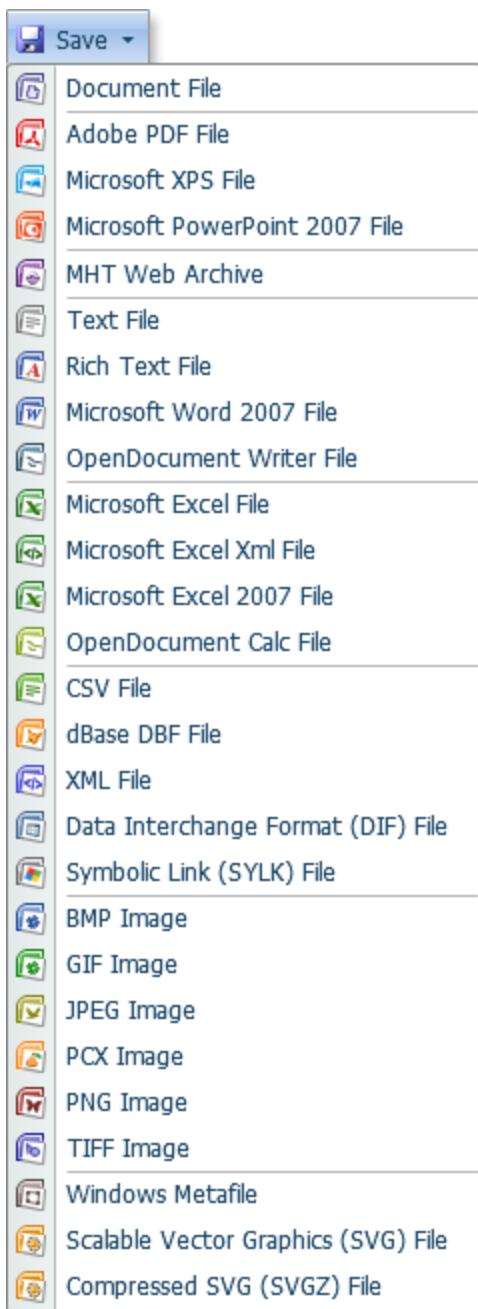
For example - export to **HTML**. Availability of this format in the list of formats for export depends on the value of the **ShowExportToHtml** property. The picture below shows the full list of formats:



As seen in the picture above the **Html** format is shown in the list of formats, and this means that the **ShowExportToHtml** property is set to **true**. If we set this property to **false** (code below):

```
<ccl:StiWebViewerFx ID="StiWebViewerFx1" runat="server"  
ShowExportToHtml="False" />
```

then the **HTML** format will not be shown in the list of export formats. The picture below shows a list of formats without **HTML**:



By default, all export formats are shown.

10.7 Data

To preview the report data are required. By default, data are taken from the **Dictionary** of the edited report. If necessary, they can be overridden. To do this you need to add the handler to the **GetDataSet** event. There is a sample code below using what data can be overridden:

C#:

```
protected void StiWebDesigner1_GetDataSet(object sender,
StiWebDesigner.StiPreviewDataSetEventArgs e)
{
    DataSet data = new DataSet();
    data.ReadXml("D:\\Demo.xml");
    data.ReadXmlSchema("D:\\Demo.xsd");
    e.DataSet = data;
}
```

VB.NET:

```
Protected Sub StiWebDesigner1_GetDataSet(ByVal sender As Object, ByVal e As StiWebDesigner.StiPreviewDataSetEventArgs)
    Dim data As DataSet = New DataSet()
    data.ReadXml("D:\\Demo.xml")
    data.ReadXmlSchema("D:\\Demo.xsd")
    e.DataSet = data
End Sub
```

As seen from code, data are taken from **XML** and **XSD** files. The same way exists for other data sources.

11 Flex Viewer

The **Viewer.Fx** component is delivered with **Stimulsoft Reports.Fx**. This component is used to show reports in Flex applications.

11.1 How to Show Report?

Put the **Viewer.Fx** on the scene of a **Flex** application:

```
<viewer:StiViewerFx id="viewerFx" left="0" right="0" bottom="0" />
```

Create, load and assign a report:

```
var report: StiReport = new StiReport();
report.loadDocumentFromString(documentString);
viewerFx.report = report;
```

Important: documentString: String - .mdc file loaded as a string.

Also, there is another way to show a report instead of placing **Viewer.Fx** on the scene of a Flex application:

```
var report: StiReport = new StiReport();
report.loadDocumentFromString(documentString);
report.showDialog();
```

Important: report.show() - showing ViewerFx on the working space of the application.

11.2 Dialog Options

If a report is shown in the dialog window, i.e. ability to set parameters of dialog window.

```
var report: StiReport = new StiReport();
report.loadDocumentFromString(documentString);
report.showDialog(rectangle: Rectangle, title: String, allowResize: Boolean, allowDrag: Boolean);
```

In example 4 parameters are described:

- ▷ **Rectangle** (position and dialog window size), is set as **Rectangle (x, y, width, height)**, where **x,y** are indents from the top left corner of the application. By default **Rectangle (10, 10, application.width - 20, application.height - 20)**;
- ▷ **title** - is the window title. If the title is not set, then the window will have the "**Viewer**" title.
- ▷ **allowResize** - this parameter allows changing window size. It may have two values: **true** and **false**. By default, the value is set to **false**, i.e. **it is impossible** to change window size.
- ▷ **allowDrag** - this parameter allows dragging dialog window. It may have two values: **true** and **false**. The default value is **false**, i.e. **it is impossible** to drag the dialog viewer window.

A sample of setting parameters:

```
var rect: Rectangle = new Rectangle(100, 100, 900, 600);
report.showDialog(rect, "Customized ViewerFx", true, true);
```

As seen from the sample:

- ▷ Indent from the top left corner is **100** by **x, y** axes. Width **900**, height **600**.
- ▷ The name of the dialog box is "**Customized ViewerFx**".
- ▷ Changing size of the dialog window of viewer - **possible**.
- ▷ Dragging the dialog window - **possible**.

11.3 User Interface Settings

It is possible to setup user interface, i.e. it is possible to hide some buttons or panels. On the picture above only 4 buttons of 9 are shown. The code below shows how to get this result:

```
StiOptions.viewer.toolbar.showOpenButton = false;
```

```
StiOptions.viewer.toolbar.showSaveButton = false;
StiOptions.viewer.toolbar.showThumbnailsButton = false;
StiOptions.viewer.toolbar.showBookmarksButton = false;
StiOptions.viewer.toolbar.showFindButton = false;
```

In other words each button has the show function. This function has two values: **true** or **false**. The default value of this function is **true**.

The list of available buttons

On the toolbar:

- ✓ **StiOptions.viewer.toolbar.showPrintButton** - Print button;
- ✓ **StiOptions.viewer.toolbar.showOpenButton** - Open button;
- ✓ **StiOptions.viewer.toolbar.showSaveButton** - Save button
- ✓ **StiOptions.viewer.toolbar.showBookmarksButton** - Bookmark button;
- ✓ **StiOptions.viewer.toolbar.showThumbnailsButton** - Thumbnails button;
- ✓ **StiOptions.viewer.toolbar.showFindButton** - Find button;
- ✓ **StiOptions.viewer.toolbar.showCloseButton** - Close button.

On the Navigation toolbar:

- ✓ **StiOptions.viewer.toolbar.showFirstPageButton** - First Page button;
- ✓ **StiOptions.viewer.toolbar.showPreviousPageButton** - Previous Page button;
- ✓ **StiOptions.viewer.toolbar.showGoToPageButton** - GoToPage button;
- ✓ **StiOptions.viewer.toolbar.showNextPageButton** - Next Page button;
- ✓ **StiOptions.viewer.toolbar.showLastPageButton** - Last Page button.

On the View Page toolbar:

- ✓ **StiOptions.viewer.toolbar.showPageViewModeSingleButton** - Single Page button;
- ✓ **StiOptions.viewer.toolbar.showPageViewModeContinuousButton** - Continuous Page button;
- ✓ **StiOptions.viewer.toolbar.showPageViewModeMultipleButton** - Multiple Page button.

Also it is possible to disable the Zoom panel, see the following:

```
StiOptions.viewer.toolbar.showZoom = false
```

12 Web Designer in Silverlight

The **StiWebDesignerSL** component is designed to edit reports in a window of a browser. You do not need to install **.NET Framework**, **ActiveX** components or other special plug-ins on the client. All you need is a Web browser and Silverlight Runtime. With **StiWebDesignerSL** it is possible to create, edit, save, view and print reports on any computer with any operating system, where there is Internet access and a Web browser with installed Silverlight Runtime version 4. **StiWebDesignerSL** is an ASP.NET component. It can be divided into two parts: client and server. Client side is a graphical part of the designer, realized under the Silverlight technology. Server side is a report engine and a module that performs the functions of receiving requests and passing data to the client side of the designer. These two parts are assembled into a single dll library and presented as a component.

12.1 How It Works?

To run the web report designer, it is required to put the **StiWebDesignerSL** component on the ASP.NET page. Then, in the **PageLoad** event of a page, you need to assign a report to the **Report** property of a component. An ASP.NET component will read the Silverlight-client application into memory from resources and run it. Being loaded, the client side will request all necessary settings and a report file from the server side. The server part will pass this. When you save or preview the report, the client side sends the report as an XML file, and server will perform preliminary processing of the report, and transfer this file for saving, or compiles it and displays in a browser window.

12.2 How to Run Web Report Designer?

For running the Web report designer it is necessary to put non visual **StiWebDesignerSL** component on the form and, in the event handler of a control, to call the **Design()** method:

ASP.NET:

```
<cc1:StiWebDesignerSL ID="StiWebDesignerSL1" runat="server" />
```

C#:

```
protected void Page_Load(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    StiWebDesignerSL1.Report = myReport;
}
```

For loading a report in the Web designer, the method of calling can be slightly modified:

C#:

```
protected void Page_Load(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\\\SimpleList.mrt");
    StiWebDesignerSL1.Design(report);
}
```

12.3 Loading Reports to Web Designer

One of the following methods can be used to load a report to the **Web designer**:

- ✓ Loading a report before loading the designer;
- ✓ Loading a report after loading the designer;
- ✓ Loading a report from the main menu of the designer.

► **Loading a report before loading the designer.** In this way the report (for example, from a file) is loaded first and then the designer is loaded. The previously loaded report is specified as a parameter of a method of calling the designer. A code below is a sample for loading a report before loading the designer:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    StiWebDesignerSL1.Design(report);
}
```

or, as a way, the previously loaded report is assigned to the designer. In this case designer loading is done with this report. See the code below:

```
protected void Button1_Click(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    StiWebDesignerSL1.Report = report;
    StiWebDesignerSL1.Design();
}
```

► **Loading a report after loading the designer** is done using the **GetReport** event. After adding the handler to this event, it will occur each time when a report is required for the designer. In other words, after loading the Web designer requests a report from the server and, if the handler is added to the **GetReport** event, then in this event a report can be assigned to the designer. See the code below how to use the **GetReport** event:

```
protected void StiWebDesignerSL1_GetReport(object sender,
    StiWebDesignerSL.StiGetReportEventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    e.Report = report;
}
```

► **Loading a report from the main menu of the designer.** A report can be loaded by selecting the **Open Report** menu item. After selecting this menu item the dialog box for specifying a report for loading will appear. Also the designer supports loading reports and other report items (for example, images) using **Drag&Drop**.

12.4 Report Preview

You can preview the report in the window of the Web-designer by selecting the **Preview** tab in the designer. The picture below shows tabs of the Web-designer:



Data are required to preview a rendered report. By default, data specified in the **Dictionary** of the edited report are taken. If it is necessary, they can be overridden. Below is a sample code with which overrides the data:

```
protected void StiWebDesignerSL1_GetPreviewDataSet(object sender,
StiWebDesignerSL.StiPreviewDataSetEventArgs e)
{
    DataSet data = new DataSet();
    data.ReadXml("D:\\Demo.xml");
    data.ReadXmlSchema("D:\\Demo.xsd");
    e.PreviewDataSet = data;
}
```

As can be seen from the code, the data is taken from XML and XSD files. In the same way you can substitute the data from other data sources.

12.5 Web Designer Settings

Setting the **Web** designer can be done using the static properties, which are described in the **Stimulsoft.Report.Web.StiWebDesignerSLOptions** class. Static properties of the Web designer can be divided into following groups: Main menu, Zooming, Viewer.

12.5.1 Main Menu

The main menu of the **Web** designer can be setup according to user's requirements. This group of static properties **StiWebDesignerSLOptions.Menu** allows enabling/disabling the main menu or submenu items

Name	Description
SaveReportAsPageEnabled	property enables/disables the Save Report As... menu item. If the SaveReportAsPageEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true ;
OpenReportEnabled	property enables/disables the Open Report menu item. If the OpenReportEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true ;
CloseEnabled	property enables/disables the Close menu item. If the CloseEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true ;

SavePageAsEnabled	property enables/disables the Save Page As... menu item. If the SavePageAsEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true ;
OpenPageEnabled	property enables/disables the Open Page... menu item. If the OpenPageEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true ;
DeletePageEnabled	property enables/disables the Delete Page menu item. If the DeletePageEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true ;
NewEnabled	property enables/disables the New menu item. If the NewEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true . The New menu item contains the submenu, where the submenu item are present. They are: New Report , New Report With Wizard , New Page . The picture below is shows the submenu of the New item:
ReportEnabled	property enables/disables the Report menu item. If the ReportEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true . This menu item contains the submenu shown on the picture below:
DesignerEnabled	property enables/disables the Designer menu item. If the DesignerEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true . This menu item contains the submenu shown on the picture below:
OptionsEnabled	property enables/disables the Options menu item. If the OptionsEnabled property is set to true , then the menu item is enabled and can be available for a user. If false then it is disabled and cannot be available for a user. By default the property is set to true .

12.5.2 Zooming

The properties of the **StiWebDesignerSL** Zoom panel are described below:

Name	Description
ShowPageViewContinuousModeButton	Property is used to show/hide the Continuous button. If the ShowPageViewContinuousModeButton property is set to true , then the Continuous button will be shown. If the ShowPageViewContinuousModeButton property is set to false , then the Continuous button will be hidden. By default, this property is set to true .

ShowPageViewMultipleModeButton	Property is used to show/hide the Multiple Pages button. If the ShowPageViewMultipleModeButton property is set to true , then the Multiple Pages button will be shown. If the ShowPageViewMultipleModeButton property is set to false , then the Multiple Pages button will be hidden. By default, this property is set to true .
ShowSliderZoomControl	Property is used to show/hide the Zoom slider. If the ShowZoom property is set to true , then the Zoom slider will be shown. If the ShowZoom property is set to false , then the Zoom slider will be hidden. By default, this property is set to true .
ZoomMode	<p>Is used to change report zoom. This property has the following values: Default, OnePage, TwoPages, PageWidth:</p> <ul style="list-style-type: none"> ▷ The Default value sets previously saved zoom of a report in WebDesignerSL. So, if a report was saved with 37% zoom then, when opening it next time, 37% zoom of a report showing remains; ▷ The PageWidth value sets zoom by Page Width. So the width of the report template matches the width of the window of the web designer; ▷ The PageHeight value sets zoom by Page Height. So the height of the report template matches the height of the window of the web designer; ▷ The OnePage value sets zoom by One Page. So the entire page of the report template fits in the window of the web designer.
Zoom	Provides an opportunity to zoom in the designer. This property can take any value from 0 to 100 , where the value of the Zoom is the zoom percentage. For example, if the Zoom property is set to 70 , the zoom in the designer will be equal to 70 percent.

12.5.3 Viewer

The group of static **StiWebDesignerSLOptions.Viewer**. properties allows setting the viewer. The list of properties is represented below.

Name	Description
ShowMainToolBar	Property is used to show/hide the Tool Bar . If the ShowMainToolBar property is set to true , then the Tool Bar panel will be shown. If the ShowMainToolBar property is set to false , then the Tool Bar panel will be hidden. By default the property is set to true .
ShowPrintButton	Property is used to show/hide the Print button. If the ShowPrintButton property is set to true , then the Print button will be shown. If the ShowPrintButton property is set to false , then the Print button will be hidden. By default the property is set to true .
ShowReportOpenButton	Property is used to show/hide the Open button. If the ShowReportOpenButton property is set to true , then the Open button will

	be shown. If the ShowReportOpenButton property is set to false , then the Open button will be hidden. By default the property is set to true .
ShowReportSaveButton	Property is used to show/hide the Save button. If the ShowReportSaveButton property is set to true , then the Save button will be shown. If the ShowReportSaveButton property is set to false , then the Save button will be hidden. By default the property is set to true .
ShowPageNewButton	Property is used to show/hide the Page New button. If the ShowPageNewButton property is set to true , then the Page New button will be shown. If the ShowPageNewButton property is set to false , then the Page New button will be hidden. By default the property is set to true .
ShowPageDeleteButton	Property is used to show/hide the Page Delete button. If the ShowPageDeleteButton property is set to true , then the Page Delete button will be shown. If the ShowPageDeleteButton property is set to false , then the Page Delete button will be hidden. By default the property is set to true .
ShowPageDesignButton	Property is used to show/hide the Edit button. If the ShowPageDesignButton property is set to true , then the Edit button will be shown. If the ShowPageDesignButton property is set to false , then the Edit button will be hidden. By default the property is set to true .
ShowPageSizeButton	Property is used to show/hide the Page Size button. If the ShowPageSizeButton property is set to true , then the Page Size button will be shown. If the ShowPageSizeButton property is set to false , then the Page Size button will be hidden. By default the property is set to true .
ShowBookmarksPanel	Property is used to show/hide the Bookmarks panel. If the ShowBookmarksButton property is set to true , then the Bookmarks panel will be shown. If the ShowBookmarksButton property is set to false , then the Bookmarks panel will be hidden. By default the property is set to true .
ShowToolFindButton	Property is used to show/hide the Find button. If the ShowToolFindButton property is set to true , then the Find button will be shown. If the ShowToolFindButton property is set to false , then the Find button will be hidden. By default the property is set to true .
ShowFullScreenButton	Property is used to show/hide the Full Screen button. If the ShowFullScreenButton property is set to true , then the Full Screen button will be shown. If the ShowFullScreenButton property is set to false , then the Full Screen button will be hidden. By default the property is set to true .
ShowZoomOnePageButton	Property is used to show/hide the One Page button. If the ShowZoomOnePageButton property is set to true , then the One Page button will be shown. If the ShowZoomOnePageButton property is set to false , then the One Page button will be hidden. By default the property is set to true .
ShowZoomTwoPagesButton	Property is used to show/hide the Two Pages button. If the ShowZoomTwoPagesButton property is set to true , then the Two Pages button will be shown. If the ShowZoomTwoPagesButton property is set

	to false , then the Two Pages button will be hidden. By default the property is set to true .
ShowZoomPageWidthButton	Property is used to show/hide the Page Width button. If the ShowZoomPageWidthButton property is set to true , then the Page Width button will be shown. If the ShowZoomPageWidthButton property is set to false , then the Page Width button will be hidden. By default the property is set to true .
ShowToolEditorButton	Property is used to show/hide the Tool Editor button. If the ShowToolEditorButton property is set to true , then the Tool Editor button will be shown. If the ShowToolEditorButton property is set to false , then the Tool Editor button will be hidden. By default the property is set to true .
ShowDocumentButton	Property is used to show/hide the Document button. If the ShowDocumentButton property is set to true , then the Document button will be shown. If the ShowDocumentButton property is set to false , then the Document button will be hidden. By default the property is set to true .
ShowPdfButton	Property is used to show/hide the Export to PDF button . If the ShowPdfButton property is set to true , then the Tool Editor button will be shown. If the ShowPdfButton property is set to false , then the Tool Editor button will be hidden. By default the property is set to true .
ShowFirstPageButton	Property is used to show/hide the First Page button. If the ShowFirstPageButton property is set to true , then the First Page button will be shown. If the ShowFirstPageButton property is set to false , then the First Page button will be hidden. By default the property is set to true .
ShowPageLastButton	Property is used to show/hide the Last Page button. If the ShowPageLastButton property is set to true , then the Last Page button will be shown. If the ShowPageLastButton property is set to false , then the Last Page button will be hidden. By default the property is set to true .
ShowPageGoToButton	Property is used to show/hide the Go to Page button. If the ShowPageGoToButton property is set to true , then the Go to Page button will be shown. If the ShowPageGoToButton property is set to false , then the Go to Page button will be hidden. By default the property is set to true .
ShowPageNextButton	Property is used to show/hide the Next Page button. If the ShowPageNextButton property is set to true , then the Next Page property is set to false , then the ShowPageNextButton property is set to false , then the Next Page button will be hidden. By default the property is set to true .
ShowPreviousPageButton	Property is used to show/hide the Previous Page button. If the ShowPreviousPageButton property is set to true , then the Previous Page button will be shown. If the ShowPreviousPageButton property is set to false , then the Previous Page button will be hidden. By default the property is set to true .

ShowPageViewSingleModeButton	Property is used to show/hide the Single Page button. If the ShowPageViewSingleModeButton property is set to true , then the Single Page button will be shown. If the ShowPageViewSingleModeButton property is set to false , then the Single Page button will be hidden. By default the property is set to true .
ShowPageViewContinuousModeButton	Property is used to show/hide the Continuous button. If the ShowPageViewContinuousModeButton property is set to true , then the Continuous button will be shown. If the ShowPageViewContinuousModeButton property is set to false , then the Continuous button will be hidden. By default the property is set to true .
ShowPageViewMultipleModeButton	Property is used to show/hide the Multiple Pages button. If the ShowPageViewMultipleModeButton property is set to true , then the Multiple Pages button will be shown. If the ShowPageViewMultipleModeButton property is set to false , then the Multiple Pages button will be hidden. By default the property is set to true .
ShowSliderZoomControl	Property is used to show/hide the Zoom slider. If the ShowZoom property is set to true , then the Zoom slider will be shown. If the ShowZoom property is set to false , then the Zoom slider will be hidden. By default the property is set to true .
Zoom	Property is used to change the report zoom in the viewer. This property can have any value from 0 to 100 , where the Zoom value is zoom in percent.

12.6 Changing Web Designer Properties from Code

The **PreInit** event is used to change **Web designer** properties. This event occurs before initialization of the designer, i.e. before passing Web designer properties to the client part of an application. In other words, to change the Web designer properties from code it is necessary to add the handler to the **PreInit** event. The code below shows how to add the handler to the PreInit event and set the Localization property to en:

```
protected void StiWebDesignerSL1_PreInit(object sender,
StiWebDesignerSL.StiPreInitEventArgs e)
{
    e.WebDesignerSL.Localization = "en";
}
```

Now, when running the **Web designer**, it will be localized in English. For example, we need to change the browser title. By default, the browser title is the value of the **Report Alias** property. If the value is not set then the **Report Name** is taken. To change the browser title it is necessary to add the code below to the event:

```
protected void StiWebDesignerSL1_PreInit(object sender,
```

```
StiWebDesignerSL.StiPreInitEventArgs e)
{
    e.WebDesignerSL.BrowserTitle = "Stimulsoft";
}
```

Now, when running the **Web designer, Stimulsoft** word will be shown as a report title.

12.7 Web Report Designer Localization

The Web designer can be localized in 26 languages. It is necessary to change the value of the Localization property. The localization file will be applied right after loading the Web designer. See the code that can be used to change the localization:

```
<cc1:StiWebDesignerSL ID="StiWebDesignerSL1" runat="server"
Localization="ru" />
```

12.8 WCF Server

When designing a report, a user may process events via the **WCF** server. For this you need to set the **UseWCFSERVICE** property to **true**:

```
Stimulsoft.Report.StiOptions.Silverlight.WCFSERVICE.UseWCFSERVICE =
true;
```

When the **UseWCFSERVICE** property is set to **true**, a user may use the following events:

```
Stimulsoft.Report.StiOptions.Silverlight.WCFSERVICE.WCFRenderReport
```

The **WCFRenderReport** event occurs when rendering a report;

```
Stimulsoft.Report.StiOptions.Silverlight.WCFSERVICE.WCFTestConnection
```

The **WCFTestConnection** event occurs when clicking the **Test Connection** button;

```
Stimulsoft.Report.StiOptions.Silverlight.WCFSERVICE.WCFBuildObjects
```

The **WCFBuildObjects** event occurs when returning the list tables from the created data source;

```
Stimulsoft.Report.StiOptions.Silverlight.WCFSERVICE.WCFRetrieveColumns
```

The **WCFRetrieveColumns** event occurs when returning the list of data columns for the table;

```
Stimulsoft.Report.StiOptions
    .Silverlight.WCFService.WCFOpeningReportInDesigner
```

The **WCFOpeningReportInDesigner** event occurs when clicking the **Open Report** button in the main menu;

```
Stimulsoft.Report.StiOptions.Engine.GlobalEvents.SavingReportInDesigner
```

The **SavingReportInDesigner** event occurs when saving a report;

```
Stimulsoft.Report.StiOptions.Silverlight.WCFService.WCFExportDocument
```

The **WCFExportDocument** event occurs when exporting a report by means of the server. In order to make available a menu with exports in the viewer by means of the server, you must set the **ShowReportSaveToServerButton** property to **true**:

```
Stimulsoft.Report.StiOptions
    .Viewer.Elements.ShowReportSaveToServerButton = true;
```

13 Silverlight Web Viewer

The **StiWebViewerSL** component is delivered as a part of **Stimulsoft Reports.WebSL**. The component is used for showing reports in the web browser.

13.1 How to Show Report?

Put the **StiWebViewerSL** component on a web page. Then you need to use the following code to show a report:

```
Stimulsoft.Report.StiReport report = new Stimulsoft.Report.StiReport();
report.Load("Simple_List.mrt");
WebViewerSL1.Report = report;
```

If the report was not rendered before showing, then the **WebViewerSL** component renders it automatically. Also the viewer supports loading reports using the **Drag&Drop**.

13.2 WebViewerSL Settings

Setting the **WebViewerSL** can be done using the properties, which are described in the **StiWebViewerSL** class.

13.2.1 Control Panel

A list of properties for customizing the **WebViewerSL** toolbar:

Name	Description
ShowMainToolBar	Property is used to show/hide the Tool Bar . If the ShowMainToolBar property is set to true , then the Tool Bar panel will be shown. If the ShowMainToolBar property is set to false , then the Tool Bar panel will be hidden. By default the property is set to true .
ShowPrintButton	Property is used to show/hide the Print button. If the ShowPrintButton property is set to true , then the Print button will be shown. If the ShowPrintButton property is set to false , then the Print button will be hidden. By default the property is set to true .
ShowReportOpenButton	Property is used to show/hide the Open button. If the ShowReportOpenButton property is set to true , then the Open button will be shown. If the ShowReportOpenButton property is set to false , then the Open button will be hidden. By default the property is set to true .
ShowReportSaveButton	Property is used to show/hide the Save button. If the ShowReportSaveButton property is set to true , then the Save button will be shown. If the ShowReportSaveButton property is set to false , then the Save button will be hidden. By default the property is set to true .
ShowPageNewButton	Property is used to show/hide the Page New button. If the ShowPageNewButton property is set to true , then the Page New button will be shown. If the ShowPageNewButton property is set to false , then the Page New button will be hidden. By default the property is set to true .
ShowPageDeleteButton	Property is used to show/hide the Page Delete button. If the ShowPageDeleteButton property is set to true , then the Page Delete button will be shown. If the ShowPageDeleteButton property is set to false , then the Page Delete button will be hidden. By default the property is set to true .
ShowPageDesignButton	Property is used to show/hide the Edit button. If the ShowPageDesignButton property is set to true , then the Edit button will be shown. If the ShowPageDesignButton property is set to false , then the Edit button will be hidden. By default the property is set to true .
ShowPageSizeButton	Property is used to show/hide the Page Size button. If the ShowPageSizeButton property is set to true , then the Page Size button will be shown. If the ShowPageSizeButton property is set to false , then the Page Size button will be hidden. By default the property is set to true .
ShowBookmarksPanel	Property is used to show/hide the Bookmarks panel. If the ShowBookmarksButton property is set to true , then the Bookmarks panel will be shown. If the ShowBookmarksButton property is set to false , then the Bookmarks panel will be hidden. By default the property is set to true .

ShowToolFindButton	Property is used to show/hide the Find button. If the ShowToolFindButton property is set to true , then the Find button will be shown. If the ShowToolFindButton property is set to false , then the Find button will be hidden. By default the property is set to true .
ShowFullScreenButton	Property is used to show/hide the Full Screen button. If the ShowFullScreenButton property is set to true , then the Full Screen button will be shown. If the ShowFullScreenButton property is set to false , then the Full Screen button will be hidden. By default the property is set to true .
ShowZoomOnePageButton	Property is used to show/hide the One Page button. If the ShowZoomOnePageButton property is set to true , then the One Page button will be shown. If the ShowZoomOnePageButton property is set to false , then the One Page button will be hidden. By default the property is set to true .
ShowZoomTwoPagesButton	Property is used to show/hide the Two Pages button. If the ShowZoomTwoPagesButton property is set to true , then the Two Pages button will be shown. If the ShowZoomTwoPagesButton property is set to false , then the Two Pages button will be hidden. By default the property is set to true .
ShowZoomPageWidthButton	Property is used to show/hide the Page Width button. If the ShowZoomPageWidthButton property is set to true , then the Page Width button will be shown. If the ShowZoomPageWidthButton property is set to false , then the Page Width button will be hidden. By default the property is set to true .
ShowToolEditorButton	Property is used to show/hide the Tool Editor button. If the ShowToolEditorButton property is set to true , then the Tool Editor button will be shown. If the ShowToolEditorButton property is set to false , then the Tool Editor button will be hidden. By default the property is set to true .

13.2.2 Navigation Panel

The properties of the **WebViewerSL** navigation panel are described below.

Name	Description
ShowFirstPageButton	Property is used to show/hide the First Page button. If the ShowFirstPageButton property is set to true , then the First Page button will be shown. If the ShowFirstPageButton property is set to false , then the First Page button will be hidden. By default the property is set to true .
ShowPageLastButton	Property is used to show/hide the Last Page button. If the ShowPageLastButton property is set to true , then the Last Page button will be shown. If the ShowPageLastButton property is set to false , then the Last Page button will be hidden. By default the property is set to true .

ShowPageGoToButton	Property is used to show/hide the Go to Page button. If the ShowPageGoToButton property is set to true , then the Go to Page button will be shown. If the ShowPageGoToButton property is set to false , then the Go to Page button will be hidden. By default the property is set to true .
ShowPageNextButton	Property is used to show/hide the Next Page button. If the ShowPageNextButton property is set to true , then the Next Page property is set to false , then the ShowPageNextButton property is set to false , then the Next Page button will be hidden. By default the property is set to true .
ShowPreviousPageButton	Property is used to show/hide the Previous Page button. If the ShowPreviousPageButton property is set to true , then the Previous Page button will be shown. If the ShowPreviousPageButton property is set to false , then the Previous Page button will be hidden. By default the property is set to true .

13.2.3 Zooming

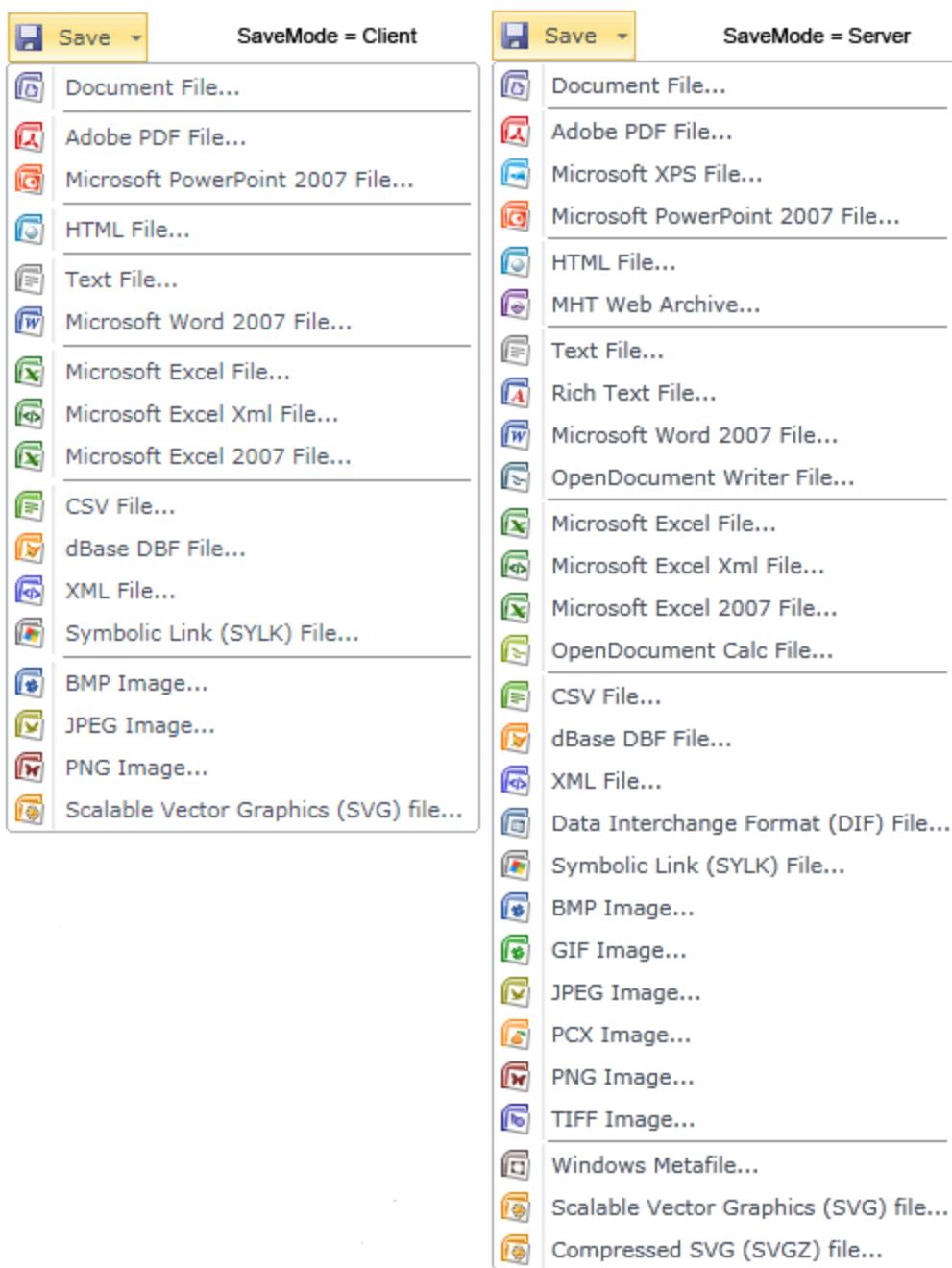
The properties of the **WebViewerSL** Zoom panel are described below:

Name	Description
ShowPageViewSingleModeButton	Property is used to show/hide the Single Page button. If the ShowPageViewSingleModeButton property is set to true , then the Single Page button will be shown. If the ShowPageViewSingleModeButton property is set to false , then the Single Page button will be hidden. By default the property is set to true .
ShowPageViewContinuousModeButton	Property is used to show/hide the Continuous button. If the ShowPageViewContinuousModeButton property is set to true , then the Continuous button will be shown. If the ShowPageViewContinuousModeButton property is set to false , then the Continuous button will be hidden. By default the property is set to true .
ShowPageViewMultipleModeButton	Property is used to show/hide the Multiple Pages button. If the ShowPageViewMultipleModeButton property is set to true , then the Multiple Pages button will be shown. If the ShowPageViewMultipleModeButton property is set to false , then the Multiple Pages button will be hidden. By default the property is set to true .
ShowSliderZoomControl	Property is used to show/hide the Zoom slider. If the ShowZoom property is set to true , then the Zoom slider will be shown. If the ShowZoom property is set to false , then the Zoom slider will be hidden. By default the property is set to true .
ZoomMode	Is used to change report zoom. This property has the following values: Default , OnePage , TwoPages , PageWidth .

	<ul style="list-style-type: none">» The Default value sets previously saved zoom of a report in WebViewerSL. So, if a report was saved with 37% zoom then, when opening it next time, 37% zoom of a report showing remains;» The PageWidth value sets zoom by Page Width. So the width of the report template matches the width of the window of the web designer;» The OnePage value sets zoom by One Page. So the entire page of the report template fits in the window of the web viewer;» The TwoPages value sets zoom by Two Pages. So two pages of a report fit by height and width the window of the WebViewerSL.
Zoom	Property is used to change the report zoom in the viewer. This property can have any value from 0 to 100 , where the Zoom value is zoom in percent. For example, if the Zoom property is set to 70 , the zoom in the viewer will be equal to 70 percent.

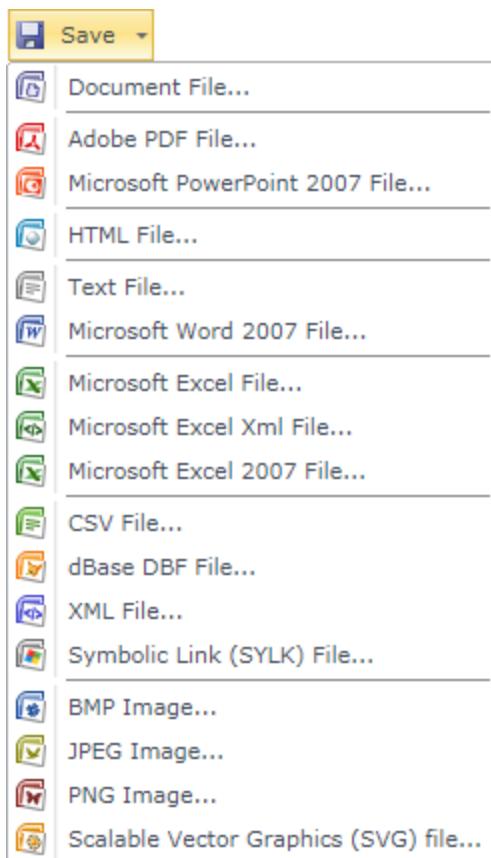
13.3 Saving Mode

When you export a report to any format, saving the report will take place in one of the following saving modes: **Client** or **Server**. Using the **SaveMode** property it is possible to change the mode of saving. If the **SaveMode** property is set to **Client**, then the report will be saved on the client side of the **WebViewerSL** application by means of **Silverlight** without a server. If the **SaveMode** property is set to **Server**, then saving the report will take place directly on the server, and after saving the report will be transferred to the client side. Depending on the value of the **SaveMode** property user will see different the lists of export formats. The picture below shows lists of exports in various saving modes:



13.3.1 Export Settings

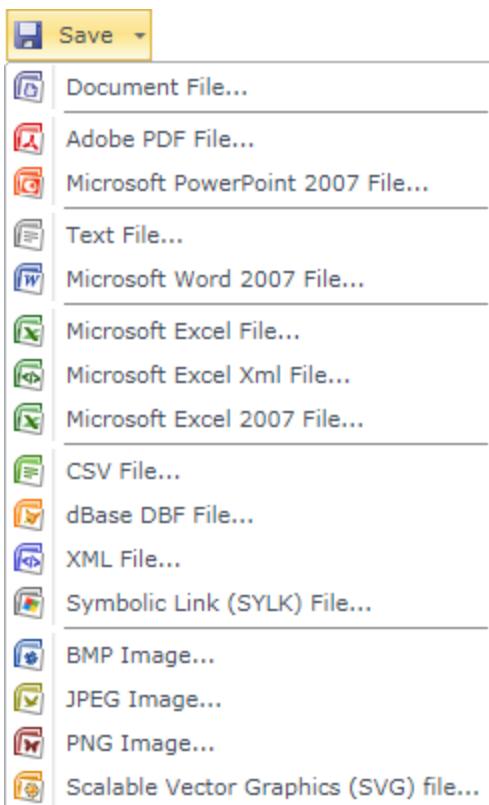
A report opened in **WebViewerSL** can be exported to many different formats. The list of formats for export can be customized. In other words, you can hide unused export formats. Customization of the list of formats of exports can be made by means of WebViewerSL properties. **For example** the **HTML** format, in the **Client** saving mode. Showing of this format in the list of formats for export depends on the value of the **ShowHtmlButton** property. The picture below shows the complete list of formats in the **Client** save mode:



As can be seen from the picture above, the **HTML** format is displayed in the list of formats that corresponds to the **ShowHtmlButton** property set to **true**. If you set this property to false:

```
<cc1:StiWebViewerSL          ID="StiWebViewerSL1"          runat="server"
  ShowHtmlButton="False" />
```

then the **HTML** will not be displayed in the list of formats for exporting. The picture below shows a list of formats for exporting without the **HTML** format:



By default, all available formats are listed for exporting.

14 HTML5 Designer

The **HTML5 Report Designer** is a Web-service. This designer is used to create and edit reports on mobile devices using a browser. Designer's interface provides the user with easy operation, a huge set of tools, components, and facilities to develop reports, their visual design and preview.

14.1 Designer Properties

Name	Description
InterfaceType	Defines the control way in the designer. If it is set to Touch, a defined method is touching. If this property is set to Mouse, then the control is carried out using the mouse device. If the property is set to UserSelection, then the user himself defines which control method to choose.
UseRelativeUrls	Defines a way of using an absolute or relative URL-address for getting images.

GlobalizationFile	Defines a path to the globalization file.
ImageFormat	Defines the format of used images.
CacheMode	Defines for the report generator which report cache to use to store images and service information. You can choose one of the following options: ➤ Page Uses the page cache. ➤ Session Uses the session cache.
ServerTimeout	Determines the storage time of the report in the server cache. By default, this property is set to "00:20:00", equivalent to 20 minutes of storing the report in the cache, i.e. after 20 minutes, the report will automatically be removed from the server cache.
ImagesPath	Defines the path to the viewer and designer images.

14.2 Working With Report Code

In this topic, we will consider events of the **HTML5** report designer.

Events	Description
CreateReport	The event occurs when you create a report.
PreviewReport	The event occurs before the report rendering.
GetReport	Loading a report after loading the report designer is carried out with help of the GetReport event. After you subscribed to this event, it will occur each time the designer will need the report. In other words, after loading, the web designer requests the report from the server and, if subscribed to the GetReport event, then you can assign the report to the designer.
SaveReport	The event occurs when saving a report.
Exit	The event occurs when closing the report designer.



Notice: You can assign a report to the designer with help of the **GetReport** event, but this also can be done using the **Page_Load** event.

To load a report in the designer, you need to assign the **Report** property of the designer. For example, from a file. If nothing is assigned to the **Report** property, the designer will be loaded blank and ready to create a report. Here is the code to load the report.

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

```

{
    if (Page != null && !Page.IsPostBack)
    {
        StiReport report = new StiReport();
        report.Load("D:\\Result.mrt");
        StiMobileDesigner1.Report = report;
    }
}

```

The main events of the report designer

Usually loadable report template does not contain the actual data. Therefore, before you render a report, connecting the data source for the report. You can do this using **OnCreateReport** and **OnLoadReport** events of the report designer.

```
<StiMobileDesigner ID="StiMobileDesigner1"
    OnSaveReport="StiMobileDesigner1_SaveReport"
    OnLoadReport="StiMobileDesigner1_GetDataSetOnLoad"
    OnCreateReport="StiMobileDesigner1_GetDataSetOnCreate" />
```

Below is a sample code of the **CreateReport** event.

```

protected void StiMobileDesigner1_CreateReport(object sender,
    StiMobileDesigner.StiCreateReportEventArgs e)
{
    DataSet data = new DataSet();
    data.ReadXmlSchema(Server.MapPath(string.Empty) + "\\Data\\
    \Demo.xsd");
    data.ReadXml(Server.MapPath(string.Empty) + "\\Data\\Demo.xml");

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

```

Below is a sample code of the **PreviewReport** event.

```

protected void StiMobileDesigner1_Preview(object sender,
    StiMobileDesigner.StiPreviewReportEventArgs e)
{
    DataSet data = new DataSet();
    data.ReadXmlSchema(Server.MapPath(string.Empty) + "\\Data\\
    \Demo.xsd");
    data.ReadXml(Server.MapPath(string.Empty) + "\\Data\\Demo.xml");
    e.Report.RegData(data);
}

```

Below is a sample code of the **GetReport** event.

```
protected void StiMobileDesigner1_GetReport(object sender,
```

```
StiMobileDesigner.StiGetReportEventArgs e)
{
    e.Report.Load(@"D:\TwoSimpleLists.mrt");

    DataSet data = new DataSet();
    data.ReadXml(Server.MapPath(string.Empty) + @"\Data\Demo.xml");

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

After creating or editing a report template, you can save changes. This can be done using the **SaveReport** event, assigning a saving method. The following example describes a method to store the report code.

```
protected void StiMobileDesigner1_SaveReport(object sender,
    StiMobileDesigner.StiSaveReportEventArgs e)
{
    StiReport report = e.Report;
    //report.Save(@"D:\" + e.Report.ReportName + ".mrt");
    report.SaveToJson(@"D:\" + e.Report.ReportName + ".mrtj");
}
```

Below is a sample code of the **Exit** event.

```
protected void StiMobileDesigner1_Exit(object sender,
    StiMobileDesigner.StiExitEventArgs e)
{}
```

[Go back the table of events](#)

15 HTML5 Viewer

The **HTML5 Viewer** is used to display reports on mobile devices. The component is able to display reports, zoom them, save to various formats and also print.

15.1 Showing Reports

Add the **StiMobileViewer** component on the **HTML** page:

```
<cc1:StiMobileViewer ID="StiMobileViewer1" runat="server" />
```

Create, load and assign a report:

```
protected void Page_Load(object sender, EventArgs e)
{
    StiReport report = new StiReport();
    report.Load("D:\\SimpleList.mrt");
    StiWebViewer1.Report = report;
}
```

15.2 Caching

The **StiMobileViewer** component can output reports in two modes: 1. Using the caching and 2. Without caching. If the caching is not used then it is necessary, when every page refreshing, to get data from a report and render a report again. When using caching the rendered report is saved in cache on the server. The next time when the page is refreshed, the previously rendered report is loaded from cache and its re-rendering is not required. It is important to remember that every report saved in cache takes the server memory and, if there are a lot of queries to reports, it can be a critical factor. Therefore, one should choose either low requirements to the memory but high requirements to performance or high requirements to the memory but low requirements to speed. Caching should not be used if the end user needs a report with actual data when every refreshing. The caching process can be controlled using the **RenderMode**, **CacheMode**, and **ServerTimeOut** properties. If the caching of a report is not used then the report that was rendered using the last data when page refreshing will be printed but not the report that is shown on the current moment. If it is necessary to get the exact copy of a report from the browser, then it is necessary to use caching.

15.2.1 RenderMode Property

The RenderMode property indicates how and when a report should be rendered. All modes of the **StiMobileViewer** component can be divided in two categories: using the caching of a rendered report, and without using caching of the rendered report.

The modes without caching

► **RenderOnlyCurrentPage**

Very interesting mode of the report output. In this mode, the report is rendered only to the page that is currently displayed in the **StiMobileViewer** component. For example, if the report consists of 100 pages (this is a big report to be output in the web), and the current page is the page number 5, the report will be rendered only up to the page number 5. The sever memory is saved in this mode.

► **Ajax**

This mode uses Ajax to output a report and to update the content of the **StiMobileViewer** component. For example, if a user goes to the next page of a report then not the whole page of the browser on what the **StiMobileViewer** component is placed will be refreshed but only the next page of a report will be sent to the browser using the post-back query. This increases the convenience of working with the **StiMobileViewer** component.

Modes with caching are UseCache and AjaxWithCache.

► **UseCache**

With each refresh in the **StiMobileViewer** component page reloading from the server occurs, but the report is not re-rendered but each time is loaded from the cache.

▷ AjaxWithCache

This mode as well as the Ajax mode uses the Ajax technology to output a report and also is used for refreshing operations of the **StiMobileViewer** component. However, unlike the Ajax mode, this **AjaxWithCache** mode does not re-render the report after each information request on the server. The report that was earlier saved in cache is used.

15.2.2 CacheMode Property

The **CacheMode** property indicates what cache should be used to store reports, images, and service information. There are two ways:

▷ Page

A page cache will be used.

▷ Session

A session cache will be used.

15.2.3 ServerTimeout Property

The **ServerTimeout** property indicates the amount of time on what it is necessary to save a report, pictures of a report, or other service information in the cache. Do not use too much time or too little time. If the time is too large, then the used cache will be overflowed and will be automatically cleared by the server. As a result, the report (or images of a report) will not be cached, and incorrect result of the report output in the **StiMobileViewer** component will occur. If the time is too small, then by the time of request to the cache, there some necessary data may not be found. It is recommended to set the time equal to 10 minutes. But the exact time can be found experimentally, considering the parameters of the server, users activity, etc.

15.3 StiReportResponse Class

A report can be shown without using the **StiMobileViewer** component. The special **Stimulsoft.Report.Mobile.StiReportResponse** class is used in this case. This class is a set of methods for saving a rendered report in different formats to the stream of a page. It is possible to assign a lot of input parameters which allows controlling the saving format of a report. For example, the following code can be used to save a report to the PDF format to the stream of a page:

```
StiReport report = new StiReport();
report.Load("MyReport.mrt");
report.Render(false);
Stimulsoft.Report.Mobile.StiReportResponse.ResponseAsPdf(this, report);
```

In this code the report will be loaded first. Then this report will be rendered. Then the result of the report rendering will be saved to the stream of a page. The following code saves a report to the **Excel 2007** format.

```
StiReport report = new StiReport();
```

```

report.Load("MyReport.mrt");
report.Render(false);
Stimulsoft.Report.Mobile.StiReportResponse.ResponseAsExcel2007(this,
report);

```

15.4 Localization of StiMobileViewer Component

To make the **StiMobileViewer** component "speak" another language it is necessary to copy a localization file of the standard delivery to the **Localization** folder. For example, select the "**de.xml**" file. Then define the default localization name.

```

<ccl:StiMobileViewer ID="StiMobileViewer1" runat="server"
GlobalizationFile="en.xml" />

```

15.5 Settings

Setting the **HTML5 Viewer** can be done using properties. Properties of **HTML5 Viewer** can be divided into the following groups: Zooming and Toolbar.

15.5.1 Zooming

The **Zooming** group of properties is represented by one property **ZoomPercent**. The **ZoomPercent** property value will be any number from **2** to **403**. This number corresponds to the zooming percentage of the report in the viewer.

15.5.2 Toolbar

The group of static properties **StiMobileViewerOptions.Toolbar** allows setting the toolbox of the **HTML5 Viewer**.

Name	Description
ShowZoom	Property is used to show/hide the zoom panel. If the ShowZoom property is set to true, then the zoom panel will be shown. If the ShowZoom property is set to false, then the zoom panel will be hidden. By default this property is set to true.
ShowPrintButton	Property is used to show/hide the Print button. If the ShowPrintButton property is set to true, then the Print button is shown. If the ShowPrintButton property is set to true, then the Print button is hidden. By default this property is set to true.

ShowSave	Property is used to show/hide the Save button. If the ShowSave property is set to true, then the Save button is shown. If the ShowSave property is set to false, then the Save button is hidden. By default this property is set to true.
ShowBookmarksButton	Property is used to show/hide the Bookmarks button. If the ShowBookmarksButton property is set to true, then the Bookmarks button is shown. If the ShowBookmarksButton property is set to false, then the Bookmarks button is hidden. By default this property is set to true.
ShowFirstButton	Property is used to show/hide the First Page button. If the ShowFirstButton property is set to true, then the First Page button is shown. If the ShowFirstButton property is set to false, then the First Page button is hidden. By default this property is set to true.
ShowPreviousButton	Property is used to show/hide the Previous Page button. If the ShowPreviousButton property is set to true, then the Previous Page button is shown. If the ShowPreviousButton property is set to false, then the Previous Page button is hidden. By default this property is set to true.
ShowNextButton	Property is used to show/hide the Next Page button. By default this property is set to true.
ShowLastButton	Property is used to show/hide the Last Page button. By default this property is set to true.
ShowViewMode	Property is used to show/hide the Single Page button. By default this property is set to true.
ShowAboutButton	Property is used to show/hide the About button. By default this property is set to true.
ShowInterfaceTypeButton	Property is used to show/hide the Control button. By default this property is set to true.

15.6 Properties

The properties of **HTML5 Viewer** are described below:

- ▷ The **ServerTimeout** property is used to define time of storing a report in the server cache. By default, this property is set to "**00:10:00**", this means that the report is stored **10** minutes in the server cache and then it is removed.
- ▷ The **BackColor** property is used to change the background color. By default, this property is set to **White**, this means that the background color is white. It is also possible to set any color in the **#rrggbb** format and transparent color.
- ▷ The **GlobalizationFile** property is used to localize the **HTML5 Viewer**. To do this, you must specify the value of the **GlobalizationFile** property, where the value is .xml file in the folder \Localization of

the root user project. By default, this property is set to "**en**", which means the English localization. Below is a sample code where the property is set to "**ru**", which corresponds to Russian localizations of **HTML5 Viewer**:

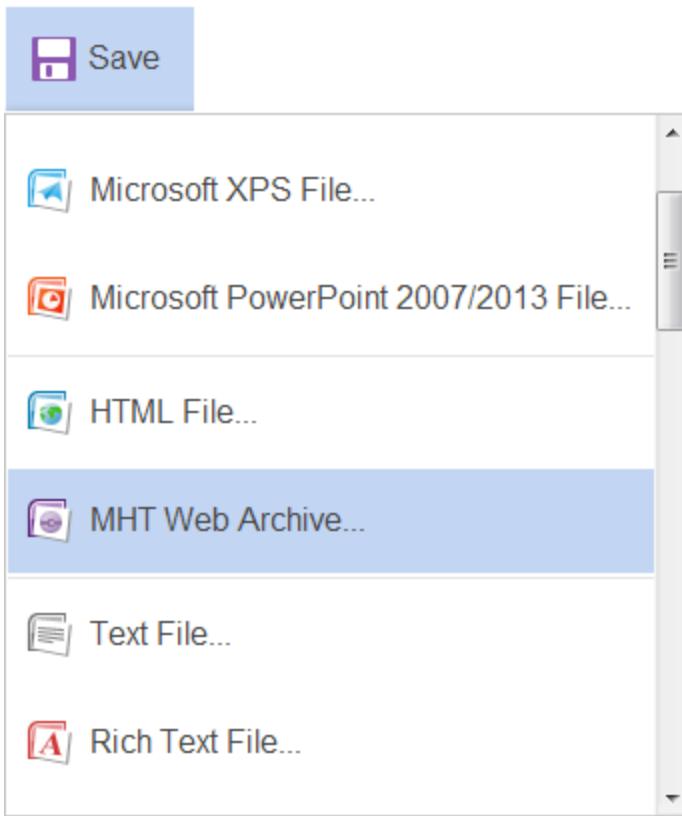
```
<ccl:StiWebDesigner ID="StiWebViewerFx1" runat="server"  
Localization="ru.xml" />
```

- » The **ImageQuality** property is used to change the quality of images in the report. depending on the value of this property it is possible to change the image file size and image quality. If the **ImageQuality** property is set to **Low**, then the file size and quality will be low. If the **ImageQuality** property is set to **Normal**, then the file size and quality will have optimal ratio between size and quality. If the **ImageQuality** property is set to **High**, then the file size and quality will be the highest.

15.7 Setting Export

It is possible to customize a list of export formats. In other words, it is possible to hide unused export formats. Customization of the export formats list can be done by using the Mobile Viewer properties.

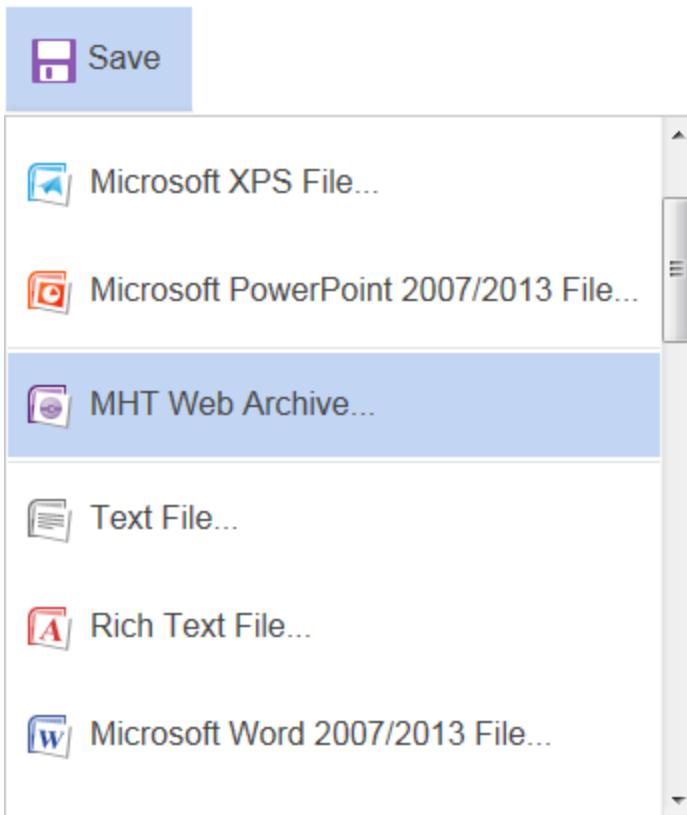
For example, the export to **HTML**. Availability of this format in the list of formats for export depends on the value of the **ShowExportToHtml** property. The picture below shows the full list of formats:



As seen in the picture above the **HTML** format is shown in the list of formats, and this means that the **ShowExportToHtml** property is set to **true**. If the set this property to **false** (code below):

```
<cc1:StiMobileViewer ID="StiMobileViewer" runat="server"
ShowExportToHtml="False" />
```

then the **HTML** format will not be shown in the list of export formats. The picture below shows a list of formats without the **HTML** option:



By default, all export formats are shown.

15.8 Defining Data

Data are needed for rendering a report. By default, the data are taken which the specified in the **Dictionary** of the the report. If you want to override the data, you should subscribe to the **GetReportData** event. Here is an example of a code used to override data:

```
protected void StiMobileViewer1_GetReportData(object sender,
StiReportEventArgs e)
{
    DataSet data = new DataSet();
    data.ReadXml(appDirectory + "\\Data\\Demo.xml");

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

As can be seen from the code, the data are taken from the **XML** and **XSD** files. The same way, you can put the data from other data sources.

16 HTML5 MVC Designer

The **HTML5 MVC Report Designer** is a Web-service. It is designed to create and edit reports on mobile devices using the browser. Designer interface provides the user with easy control, a huge set of tools, components and tools to develop reports, visual design and preview.

16.1 How It Works?

In order to run the **HTML5 MVC** report designer, it is required to put the **StiMvcMobileDesinger** component on the **ASP.NET** page, set necessary properties and define required actions in the view controller. When running the Web report designer the following actions occur:



When it is loaded, the client side requests a report that should be returned by action set and defined in the controller settings.



When exporting a report, the client side calls an action, which also must return the specific response.

16.2 How to Run Designer?

In order to run the report designer on the main master-page of the application you need to place a code of adding java-scripts, which are required for the component work properly. This code is placed in the block **<head>**:

ASPX:

```
<head runat="server">
    <%= Html.Stimulsoft().RenderMvcMobileDesignerScripts() %>
</head>
```

Razor:

```
<head runat="server">
    @Html.Stimulsoft().RenderMvcMobileDesignerScripts()
</head>
```

To view the report you should add the **StiMvcMobileDesinger** component to an **ASP.NET** page, and set it the necessary properties, and in the view controller, to determine the necessary steps

ASPX:

```
<%= Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new
StiMvcMobileDesignerOptions() {
    ActionGetReportTemplate = "GetReportTemplate",
    Width = Unit.Percentage(100),
    Height = Unit.Pixel(700)
})%>
```

Razor:

```
@Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new
StiMvcMobileDesignerOptions() {
    ActionGetReportTemplate = "GetReportTemplate",
    Width = Unit.Percentage(100),
    Height = Unit.Pixel(700)
})
```

Controller:

```
public ActionResult GetReportTemplate()
{
    StiReport report = new StiReport();
    report.Load(@"d:\Reports\SimpleList.mrt");

    DataSet data = new DataSet();
    data.ReadXml(@"D:\Data\Demo.xml");
    report.RegData(data);

    return
StiMvcMobileDesigner.GetReportTemplateResult(HttpContext, report);
}
```

16.3 Loading Reports

For loading report templates a special action is used. It has the recommended name **GetReportTemplate**:

Controller:

```
public ActionResult GetReportTemplate()
{
    StiReport report = new StiReport();
    report.Load(@"d:\Reports\SimpleList.mrt");

    DataSet data = new DataSet();
    data.ReadXml(@"D:\Data\Demo.xml");
    report.RegData(data);
```

```

        return
StiMvcMobileDesigner.GetReportTemplateResult(HttpContext, report);
    }
}

```

Controller:

```

public ActionResult GetReportTemplate()
{
    StiReport report = new StiReportCompiledClass();
    return StiMvcMobileDesigner.GetReportTemplateResult(HttpContext,
report);
}

```

16.4 Previewing Reports

The preview function of the edited report, in the **HTML5 Designer**, has two modes: **HTML**, **PDF**. To preview the rendered report it is necessary to determine a specific action which has a recommended name **GetReportSnapshot**:

ASPX:

```

<%= Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new
StiMvcMobileDesignerOptions() {
    ActionGetReportSnapshot = "GetReportSnapshot"
}) %>

```

Razor:

```

Razor:

@Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new
StiMvcMobileDesignerOptions() {
    ActionGetReportSnapshot = "GetReportSnapshot"
})

```

As in the previous form of a preview, the report will be in the data transmitted to the controller. This action should return a report prepared in a special format to the client side. To do this, use the following code:

Controller:

```

public ActionResult GetReportSnapshot()
{
    StiReport report = StiMvcMobileDesigner.GetReportObject(HttpContext);
    return StiMvcMobileDesigner.GetReportSnapshotResult(HttpContext,
}

```

```
report);
}
```

If the report requires data other than those specified in the report template, you can use the same solution as in the previous form of the preview:

Controller:

```
public ActionResult GetReportSnapshot()
{
    StiReport report =
StimvcMobileDesigner.GetReportObject(HttpContext);

    DataSet data = new DataSet();
    data.ReadXmlSchema(@"D:\Data\Demo.xsd");
    data.ReadXml(@"D:\Data\Demo.xml");

    report.RegData(data);
    report.Dictionary.Synchronize();

    return
StimvcMobileDesigner.GetReportSnapshotResult(HttpContext, report);
}
```

16.5 Saving Reports

A special action that has the name **SaveReportTemplate** for processing the saved report in the web designer:

Controller:

```
public ActionResult SaveReportTemplate()
{
    StiReport report = StimvcMobileDesigner.GetReportObject(HttpContext);

    string packedReport = report.SavePackedReportToString();
    // ...
    // Here the save report code
    // ...
    return StimvcMobileDesigner.SaveReportTemplateResult(HttpContext);
}
```

Saving the report can be performed in the background, i.e. visually it will not be displayed. At the same time, the report will be saved again, i.e. the report file will be overwritten. Also, before you save the report, you can display a dialog box in which you should specify the report name with which it will be saved. To do this, you must set the **ShowSaveDialog** property to true:

ASPX:

```
<%= Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new  
StiMvcMobileDesignerOptions() {  
    ActionGetReportTemplate = "GetReportTemplate",  
    ShowSaveDialog = true  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new  
StiMvcMobileDesignerOptions() {  
    ActionGetReportTemplate = "GetReportTemplate",  
    ShowSaveDialog = true  
})
```

16.6 Localizing HTML5 Designer

In order to localize the interface designer in the appropriate language, use special actions, that has the name **Localization**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new  
StiMvcMobileDesignerOptions() {  
    Localization = "Localizations/en.xml",  
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new  
StiMvcMobileDesignerOptions() {  
    Localization = "Localizations/en.xml",  
})
```

16.7 Other Actions of Designer

There are some other actions of the Web designer. The actions are **OpenReport**, **DesignerEvent** and **ExitDesigner**.

ASPX:

```
<%= Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new  
StiMvcMobileDesignerOptions() {
```

```
ActionOpenReportTemplate = "OpenReportTemplate",
ActionExitDesigner = "ExitDesigner",
ActionDesignerEvent = "DesignerEvent"
}) %>
```

Razor:

```
@Html.Stimulsoft().StiMvcMobileDesigner("MvcMobileDesigner1", new
StiMvcMobileDesignerOptions() {
    ActionOpenReportTemplate = "OpenReportTemplate",
    ActionExitDesigner = "ExitDesigner",
    ActionDesignerEvent = "DesignerEvent"
})
```

OpenReport:

The **OpenReport** action is used to open the report designer. This action is determined in the view controller:

Controller:

```
public ActionResult OpenReportTemplate()
{
    return StiMvcMobileDesigner.OpenReportTemplateResult(HttpContext);
}
```

ExitDesigner:

The **ExitDesigner** action is called when you click the Exit button in the main menu of the designer:

```
public ActionResult ExitDesigner()
{
    return View(" MainPage");
}
```

DesignerEvent:

All the other actions which occur in the designer register in **DesignerEvent**. **DesignerEvent** is determined in the view controller:

```
public ActionResult DesignerEvent()
{
    return StiMvcMobileDesigner.DesignerEventResult(HttpContext);
}
```

16.8 HTML5 Designer Settings

Setting the web designer can be done using the properties, which are described in the **StiMvcMobileDesinger** class.

16.8.1 Actions

This group includes properties that define the name of actions in the view controller, which will call the client side report designer using the appropriate functions. The group includes the following properties:

Name	Description
Controller	Specifies the action method name of processing queries of the report designer. If the property is not specified, then the current controller will be used for processing queries.
ActionGetReportTemplate	Specifies the action method name of loading a report template.
ActionSaveReportTemplate	Specify the action method name of saving a report template.
ActionGetReportSnapshot	Specifies the action method name of preparing a rendered report for preview.
ActionOpenReportTemplate	Specifies the action method name of opening a report template.
ActionGetNewReportData	Specifies the action method name of retrieving data to the new report template.
ActionDesignerEvent	Specifies the action method name of the designer event.
ActionExitDesigner	Specifies the action method name for transition to the desired representation when closing the web designer.

]

16.8.2 Server

This group includes properties that define parameters of the client connection and the server part.

Name	Description
ServerTimeout	Defines the time that the report will be stored in the server or session cache.
ServerRelativeUrls	Defines the time that shows that the viewer will use both relative and absolute URLs.
ServerCacheMode	Enables the mode of caching.
ServerCacheItemPriority	Sets the relative priority of the report that is stored in the system cache.

16.8.3 File Menu

This group of properties allows setting the **File** menu.

Name	Description
ShowFileMenu	Shows/hides the File menu item. Setting it to true makes the menu item visible.
ShowFileMenuNew	Shows/hides the New menu item. Setting it to true makes the menu item visible.
ShowFileMenuOpen	Shows/hides the Open menu item. Setting it to true makes the menu item visible.
ShowFileMenuSave	Shows/hides the Save menu item. Setting it to true makes the menu item visible.
ShowFileMenuClose	Shows/hides the Close menu item. Setting it to true makes the menu item visible.
ShowFileMenuExit	Shows/hides the Exit menu item. Setting it to true makes the menu item visible.

16.8.4 Interface

This group includes properties using which you can setup the interface of the designer.

Name	Description
Width	Sets the width of the component with units set in the Unit class. Values available - Unit.Pixel() , Unit.Point() and Unit.Percentage() .
Height	Sets the height of the component with units set in the Unit class Values available - Unit.Pixel() , Unit.Point() and Unit.Percentage() .
DefaultUnit	Sets units in the report by default - pixels, centimeters, inches, hundredths of inch.
ShowAnimation	Enables/disables animation. Setting it to true enables animation.
ShowPropertiesGrid	Enables/disables the Property panel. Setting it to true enables this panel.
PropertiesGridWidth	Sets the width of the property panel with units set in the Unit class. Values available - Unit.Pixel() , Unit.Point() and Unit.Percentage() .
PropertiesGridLabelWidth	Sets the width of the column header on the properties panel using the property. Values available - Unit.Pixel() , Unit.Point() and Unit.Percentage() .
ShowDictionary	Shows/hides the Data Dictionary . If this property is set to true the Dictionary will be shown.
ShowInsertButton	Shows/hides the Insert tab. If this property is set to true the tab will be shown.
ShowPageButton	Shows/hides the Page tab. If this property is set to true the tab will be shown.
ShowPreviewButton	Shows/hides the Preview tab. If this property is set to true the tab will be shown.
ShowToolips	Shows/hides the Toolips . If this property is set to true tooltips will be shown.
ShowSaveButton	Shows/hides the Save button. If this property is set to true the button will be shown.
ShowSaveDialog	Shows/hides the dialog window when saving. If this property is set to true the dialog will be shown and you should set the name of the saved report. If the property is set to false the dialog will hidden and the report will be saved with the previous name, the report will be rewritten.
Theme	Changes the UI theme using this property.
Localization	Sets the directory (relative and absolute), in which the localization files are placed. This property is used to build the list of available localizations.

16.8.5 Bands

Using this group of properties you can show/hide bands on the tab **Insert**.

Name	Description
ShowDataBand	Shows/hides the band Data . If it is set to true , the band is shown in the list of bands.
ShowHierarchicalBand	Shows/hides the band Hierarchical . If it is set to true , the band is shown in the list of bands.
ShowHeaderBand	Shows/hides the band Header . If it is set to true , the band is shown in the list of bands.
ShowFooterBand	Shows/hides the band Footer . If it is set to true , the band is shown in the list of bands.
ShowPageHeaderBand	Shows/hides the band PageHeader . If it is set to true , the band is shown in the list of bands.
ShowPageFooterBand	Shows/hides the band PageFooter . If it is set to true , the band is shown in the list of bands.
ShowReportTitleBand	Shows/hides the band ReportTitle . If it is set to true , the band is shown in the list of bands.
ShowReportSummaryBand	Shows/hides the band ReportSummary . If it is set to true , the band is shown in the list of bands.
ShowGroupHeaderBand	Shows/hides the band GroupHeader . If it is set to true , the band is shown in the list of bands.
ShowGroupFooterBand	Shows/hides the band GroupFooter . If it is set to true , the band is shown in the list of bands.
ShowColumnHeaderBand	Shows/hides the band ColumnHeader . If it is set to true , the band is shown in the list of bands.
ShowColumnFooterBand	Shows/hides the band ColumnFooter . If it is set to true , the band is shown in the list of bands.
ShowChildBand	Shows/hides the band Child . If it is set to true , the band is shown in the list of bands.
ShowOverlayBand	Shows/hides the band Overlay . If it is set to true , the band is shown in the list of bands.
ShowEmptyBand	Shows/hides the band Empty . If it is set to true , the band is shown in the list of bands.

Cross bands are grouped into another list.

Name	Description
ShowCrossDataBand	Shows/hides the cross band CrossData . If it is set to true , the cross band is shown in the list of bands.
ShowCrossHeaderBand	Shows/hides the cross band CrossHeader . If it is set to true , the cross band is shown in the list of bands.
ShowCrossFooterBand	Shows/hides the cross band CrossFooter . If it is set to true , the cross band is shown in the list of bands.
ShowCrossGroupHeaderBand	Shows/hides the cross band CrossGroupHeader . If it is set to true , the cross band is shown in the list of bands.
ShowCrossGroupFooterBand	Shows/hides the cross band CrossGroupFooter . If it is set to true , the cross band is shown in the list of bands.
ShowCrossTab	Shows/hides the cross band CrossTab . If it is set to true , the cross band is shown in the list of bands.

16.8.6 Components

Using this group of properties you can show/hide components on the tab **Insert**.

Name	Description
ShowBarCode	Shows/hides the component Bar-code . If it is set to true , the component is shown in the list of components.
ShowChart	Shows/hides the component Chart . If it is set to true , the component is shown in the list of components.
ShowText	Shows/hides the component Text . If it is set to true , the component is shown in the list of components.
ShowRichText	Shows/hides the component RichText . If it is set to true , the component is shown in the list of components.
ShowTextInCells	Shows/hides the component TextInCells . If it is set to true , the component is shown in the list of components.
ShowCheckBox	Shows/hides the component CheckBox . If it is set to true , the component is shown in the list of components.
ShowClone	Shows/hides the component Clone . If it is set to true , the component is shown in the list of components.
ShowImage	Shows/hides the component Image . If it is set to true , the component is shown in the list of components.
ShowPanel	Shows/hides the component Panel . If it is set to true , the component is shown in the list of components.

ShowShape	Shows/hides the component Shape . If it is set to true , the component is shown in the list of components.
ShowSubReport	Shows/hides the component Sub-Report . If it is set to true , the component is shown in the list of components.
ShowTable	Shows/hides the component Table . If it is set to true , the component is shown in the list of components.
ShowZipCode	Shows/hides the component ZipCode . If it is set to true , the component is shown in the list of components.

17 Stimulsoft Reports.JS

We made a library for working with Stimulsoft Reports on **JavaScript**, but we used the ideology of **.NET**, so all **JavaScript** code presented below uses .NET C# notation.

General:	Viewer:	Designer:
 Connecting Library	 Running Viewer	 Running Designer
 Loading and Saving Report	 Viewer Events	 Designer Events
 Getting Access to Pages	 Viewer Options	 Designer Options
 Rendering Report	 Viewer Appearance	 Designer Appearance
 Binding Data	 Viewer Toolbar	 Designer Toolbar
 Saving Rendered Report	 Viewer Exports	 Designer Bands, Cross-Bands, Components
 Report Printing	 Exporting Rendered Report	 Designer Dictionary

17.1 Connecting Library

The library **StimulsoftReports.JS** provides to the developer the ability to quickly and easily integrate a powerful and flexible reporting system from **Stimulsoft** in yours web application written in any language. To do this, connect the **JavaScript** library to the client application:

```
<script src="stimulsoft.reports.js"></script>
```

Additionally, to connect Stimulsoft Report **Viewer** you need to add **CSS** and **JS**:

```
<link href="stimulsoft.viewer.office2013.css" rel="stylesheet">
<script src="stimulsoft.viewer.js" type="text/javascript"></script>
```

Additionally, to connect Stimulsoft Report **Designer** you need to add **CSS** and **JS**:

```
<link href="stimulsoft.designer.office2013.white.blue.css"
rel="stylesheet">
<script src="stimulsoft.designer.js" type="text/javascript"></script>
```

Viewer

The appearance of the viewer and the designer can be modified using any of the available themes by simply connecting the desired CSS. The list of **CSS** themes:

- stimulsoft.viewer.default.min.css
- stimulsoft.viewer.windows.xp.css
- stimulsoft.viewer.windows7.css
- stimulsoft.viewer.office2003.css
- stimulsoft.viewer.office2007.blue.css
- stimulsoft.viewer.office2007.silver.css
- stimulsoft.viewer.office2007.black.css
- stimulsoft.viewer.office2010.black.css
- stimulsoft.viewer.office2010.css
- stimulsoft.viewer.office2010.silver.css
- stimulsoft.viewer.office2013.css
- stimulsoft.viewer.office2013.white.carmine.css
- stimulsoft.viewer.office2013.white.green.css
- stimulsoft.viewer.office2013.white.orange.css
- stimulsoft.viewer.office2013.white.purple.css
- stimulsoft.viewer.office2013.white.teal.css
- stimulsoft.viewer.office2013.white.violet.css

Designer

The list of CSS themes:

- stimulsoft.designer.office2013.white.blue.css
- stimulsoft.designer.office2013.white.carmine.css
- stimulsoft.designer.office2013.white.green.css
- stimulsoft.designer.office2013.white.orange.css
- stimulsoft.designer.office2013.white.purple.css
- stimulsoft.designer.office2013.white.teal.css
- stimulsoft.designer.office2013.white.violet.css
- stimulsoft.designer.office2013.lightgray.blue.css
- stimulsoft.designer.office2013.lightgray.carmine.css
- stimulsoft.designer.office2013.lightgray.green.css
- stimulsoft.designer.office2013.lightgray.orange.css
- stimulsoft.designer.office2013.lightgray.purple.css
- stimulsoft.designer.office2013.lightgray.teal.css
- stimulsoft.designer.office2013.lightgray.violet.css
- stimulsoft.designer.office2013.darkgray.blue.css
- stimulsoft.designer.office2013.darkgray.carmine.css
- stimulsoft.designer.office2013.darkgray.green.css
- stimulsoft.designer.office2013.darkgray.orange.css
- stimulsoft.designer.office2013.darkgray.purple.css
- stimulsoft.designer.office2013.darkgray.teal.css
- stimulsoft.designer.office2013.darkgray.violet.css
- stimulsoft.designer.office2013.verydarkgray.blue.css
- stimulsoft.designer.office2013.verydarkgray.carmine.css

- stimulsoft.designer.office2013.verydarkgray.green.css
- stimulsoft.designer.office2013.verydarkgray.orange.css
- stimulsoft.designer.office2013.verydarkgray.purple.css
- stimulsoft.designer.office2013.verydarkgray.teal.css
- stimulsoft.designer.office2013.verydarkgray.violet.css

The uncompressed version for customized settings for each **CSS** file is available.

Note: The HTML page must begin with a proper **DOCTYPE** definition to set the type of the current document in HTML5:

```
<!DOCTYPE html>
```

Due to limitations of the JavaScript language to work with external data, you must run a minimum server instance. This will provide the ability to open reports from files, use the external data in the report and to work with rendered reports.

17.2 Loading and Saving Report

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:

```
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:

```
var report = new Stimulsoft.Report.StiReport();
```

```
var jsonString = report.saveToJsonString();
```

Use the code below to load a report from this string:

```
var report = new Stimulsoft.Report.StiReport();
report.load(jsonString);
```

17.3 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:

```
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);
}
```

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:

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.render();

for (var index = 0; index < report.renderedPages.count; index++) {
    alert(report.renderedPages.getByIndex(index).name);
}
```

17.4 Rendering Report

The report includes presentation and data. To merge these entities and build a report using the “**render()**” method. The following code renders a report:

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.render();
```

17.5 Binding Data to Report

To connect to the data you need to perform the “**regData()**” method of the report, passing it as a data

source parameter. The data source can load data from XML files (with XSD scheme) and from JSON files. For example, loading data from **XML**:

```
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);
```

You could just download **JSON** data source:

```
var dataSet = new Stimulsoft.System.Data.DataSet("SimpleDataSet");
dataSet.readJsonFile("Demo.json");

var report = new Stimulsoft.Report.StiReport();
report.regData(dataSet.dataSetName, "", dataSet);
```

The **StiDataSet** object has other methods for loading data:

- `readJsonFile(fileName)` – takes a path to JSON file as a parameter;
- `readJson(string)` – takes a string with JSON data as a parameter;
- `readJson(data)` – takes a byte array with JSON data as a parameter;
- `readJson(obj)` – takes a JavaScript object as a parameter;

- `readXmlFile(fileName)` – takes a path to XML file as a parameter;
- `readXml(string)` – takes a string with XML data as a parameter;
- `readXml(data)` – takes an byte array with XML data as a parameter;
- `readXml(obj)` – takes a JavaScript object as a parameter;

- `readXmlSchemaFile(fileName)` – takes a path to XML schema (XSD) file as a parameter;
- `readXmlSchema(string)` – takes a string with XML schema as a parameter;
- `readXmlSchema(data)` – takes an byte array with XML schema as a parameter;
- `readXmlSchema(obj)` – takes a JavaScript object as a parameter;

Note: Loading data schema is not required. If you want to use the scheme, you must load it

before loading XML data.

To convert an array of bytes to string and back you can use the static functions –

Stimulsoft.System.Text.Encoding.UTF8.getString(buffer) (returns a string derived from the byte array passed in the parameter) and **Stimulsoft.System.Text.Encoding.UTF8.getBytes(str)** (returns a byte array derived from the string passed in the parameter). There is also another way of data management, which allows you to connect to different data sources and modify them in real time:

```
var report = new Stimulsoft.Report.StiReport();
var xmlDataBase = new
Stimulsoft.Report.Dictionary.StiXmlDatabase("Demo", "demo.xsd",
"demo.xml");

report.loadFile("SimpleList.mrt");
report.dictionary.databases.clear();
report.dictionary.databases.add(xmlDataBase);
```

This way you can easily add the required amount of data sources of different types.

17.6 Connect to MySQL and MS SQL Database

Since pure JavaScript does not allow access to a remote database, this functionality is implemented through the server-side code. For this, Stimulsoft Reports.JS supports two server-side technologies – PHP and Node.JS.

The database adapter is a layer between DBMS and the client-side script. It connects to the DBMS and receives the necessary data, converting them to the JSON format. The script running on the server (with help of adapter) provides exchange of JSON data between the client-side JavaScript and the server-side using a POST-request.

In order to use this mechanism in the client-side JavaScript must specify the URL of the host that handles POST-requests to the required adapter. In case, if it is a PHP script it indicates a web page:

```
StiOptions.WebServer.url = "http://mywebsite.com/script.php";
```

If you have adapters for Node.JS, you need to specify the host and port, running the script:

```
StiOptions.WebServer.url = "http://mywebsite.com:9615";
```

If you use the report designer, then this line should be placed before the code to display the designer. Call the function to render a report template:

```
report.renderAsync();
```

PHP and Node.JS scripts to handle POST-requests to the server and data transfer between the adapter

and the client can be downloaded together with the Stimulsoft.Reports.JS assembly. On the client-side, the connection string in any of the supported DBMS (MySQL, MS SQL or Firebird) must specify the required parameters, including the host, which is not a host that is running the DBMS, but the host on which the POST-requests are processed and the adapter is running:

```
Server=mysqlserver; Database=sakila; uid=root; password=mypass;
```

As an example the following architecture, built on multiple servers:



- a MySQL server **mysqlserver** that is running the database;
- a server supporting Node.JS **adapterserver** that is running the script to handle POST-requests (app.js) and adapter's script (in this case, a script MySQLAdapter.js);
- a HTTP server **httpserver** that is running the http server for distribution to your pages and the JavaScript client.

On the client-side, the next option runs a script that starts the designer:

```
StiOptions.WebServer.url = "http://adapterserver:9615";
```

In the form of creating a new connection the required parameters and the address of the MySQL server are specified:

```
Server=mysqlserver; Database=sakila; uid=root; password=mypass;
```

When all is done, then by pressing the Test button on the form should display "Connection was successful".

17.7 Synchronize Data between DataStore and Dictionary

In the process of adding data to the Dictionary, there is a need to synchronize the data structure of the Dictionary with the added data source. This problem is easily solved by calling:

```
Stimulsoft.Report.Dictionary.syncronise()
```

The method adds a data structure (Tables and Columns, Relations and whole DataSources) in the dictionary, and solves the problem of the empty data source added to the dictionary using the method:

```
Stimulsoft.Report.regData()
```

If DataSources, Columns or Relations from the DataStore does not exist in Dictionary, then new elements will be added to the Dictionary.

17.8 Saving Rendered Report

Rendered report can be saved into a **JSON** document for later viewing (data stored within a document):

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.render();

var jsonString = report.saveDocumentToJsonString();
```

17.9 Report Printing

Report printing is implemented by means of the browser, so the view of the dialog box may be different on various operating systems and browsers. Printing is possible after rendering a report by the "**print()**" method:

```
var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.render();

report.print();
```

The optional parameter of **Stimulsoft.Report.StiPagesRange** allows you to specify the print range. The parameters of the constructor of the Stimulsoft.Report.StiPagesRange object is:

- The range type (can have values Stimulsoft.Report.StiRangeType.All, Stimulsoft.Report.StiRangeType.Pages, Stimulsoft.Report.StiRangeType.CurrentPage);
- The range in the string representation (page numbers separated by commas may specify the range with a hyphen);
- The current page number.

Now you can flexibly configure report printing:

```
var pageRange = new
Stimulsoft.Report.StiPagesRange(Stimulsoft.Report.StiRangeType.CurrentPa
ge, "1,3-8", 5);
report.print(pageRange);
```

17.10 Running Report Viewer

Stimulsoft Report Viewer is a tool to view reports that can be easily integrated into your **Web**

application with a few lines of code. Minimal server support required to start Stimulsoft Reports.JS is any **Web** server that can provide access to the files of any type. To get started, you must connect **CSS** and **scripts**:

```
<link href="stimulsoft.viewer.office2013.css" rel="stylesheet">
<script src="stimulsoft.reports.js" type="text/javascript"></script>
<script src="stimulsoft.viewer.js" type="text/javascript"></script>
```

Then load the required report and insert it into the HTML page code to deploy Stimulsoft Report Viewer in the current DOM element:

```
<script type="text/javascript">
    var report = new Stimulsoft.Report.StiReport();
    report.loadFile("SimpleList.mrt");

    var viewer = new Stimulsoft.Viewer.StiViewer();
    viewer.report = report;
</script>
```

Stimulsoft Report Viewer object is created by the **Stimulsoft.Viewer.StiViewer()** constructor, which has three optional parameters. The first parameter is a set of options

Stimulsoft.Viewer.StiViewerOptions, divided into categories, appearance, toolbar and exports. All of them are listed in the table below. The second parameter is the identifier of the viewer as a DOM object. The third parameter is the flag of rendering the viewer in the moment of creation. If this flag is set to **true**, the viewer will be drawn in the same place of the DOM tree, which houses the code. If the flag is set to **false**, the viewer will be drawn where there is the "**renderHtml()**" method of the viewer. For example, Initialization of the viewer in the head of a page:

```
<script type="text/javascript">
    var viewer = new Stimulsoft.Viewer.StiViewer(null, "StiViewer",
        false);
</script>
```

And rendering of the viewer control in div:

```
<div>Page content</div>
<div>
    <script type="text/javascript">
        // Render the report viewer in this place
        viewer.renderHtml();
    </script>
</div>
```

Another way to display the viewer in a particular control is to use the optional method parameter – **renderHtml(elementId)**, which indicates the ID of the HTML element - Initialization of the viewer in the head of a page:

```
<script type="text/javascript" >
```

```
var viewer = new Stimulsoft.Viewer.StiViewer(null, "StiViewer",
false);
viewer.renderHtml("viewerContent");
</script>
```

This will display the viewer in the element:

```
<div id="viewerContent"></div>
```

Note: a viewer can be hidden by specifying the values of the property "visible":
`viewer.visible = false;`

17.11 Viewer Events

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:

onBeginProcessData	onPrintReport	onEndExportReport	onDesignReport
onEndProcessData	onBeginExportReport	onEmailReport	Designer Events



onBeginProcessData

is called before requesting the data for the report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current even., By default it is "BeginProcessData".
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.
connectionString	a connection string to the data source for a report.
queryString	a string with the SQL query to the database for data acquisition. Used only with the command = "ExecuteQuery".
database	a string name of the current database.
preventDefault	a flag to prevent further processing of the event. The default value is "false".

```
viewer.onBeginProcessData = function (event) {
    event.connectionString = "server=real.server; uid=root;
password=actual_password; database=production_db;";
}
```



onEndProcessData

Is called after retrieving data for the report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "EndProcessData".
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.
preventDefault	a flag to prevent further processing of the event. The default value is "false".

This event can be used to add its own data sources generated "on the fly".



onPrintReport

Is called before printing the report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "PrintReport".
fileName	a printing file name.
report	a report for saving.
preventDefault	a flag to prevent further processing of the event. The default value is "true".

```
viewer.onPrintReport = function (event) {
    console.log("printing");
}
```



onBeginExportReport

Is called before export but after applying the export options. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "BeginExportReport".
fileName	an exporting file name.

settings	export report settings.
format	export report format. Can have values <ul style="list-style-type: none">▪ Stimulsoft.Report.StiExportFormat.Pdf,▪ Stimulsoft.Report.StiExportFormat.Excel2007,▪ Stimulsoft.Report.StiExportFormat.Word2007,▪ Stimulsoft.Report.StiExportFormat.Html,▪ Stimulsoft.Report.StiExportFormat.Html5,▪ Stimulsoft.Report.StiExportFormat.Document.
preventDefault	a flag to prevent further processing of the event. The default value is "false".

```
viewer.onBeginExportReport = function (event) {
    switch (event.format) {
        case Stimulsoft.Report.StiExportFormat.Html:
            event.settings.zoom = 2; // Set zoom to 200%
            break;
    }
    console.log("exporting");
}
```



onEndExportReport

Is called after export report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "EndExportReport".
fileName	an exporting file name.
format	export report format. Can have values <ul style="list-style-type: none">▪ Stimulsoft.Report.StiExportFormat.Pdf,▪ Stimulsoft.Report.StiExportFormat.Excel2007,▪ Stimulsoft.Report.StiExportFormat.Word2007,▪ Stimulsoft.Report.StiExportFormat.Html,▪ Stimulsoft.Report.StiExportFormat.Html5,▪ Stimulsoft.Report.StiExportFormat.Document
data	export data to the string or byte array representation.
preventDefault	a flag to prevent further processing of the event. The default value is "false".



onEmailReport

Is called before sending the report by email. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "EmailReport".

fileName	an exporting file name.
settings	export report settings.
format	export report format. Can have values <ul style="list-style-type: none"> ▪ Stimulsoft.Report.StiExportFormat.Pdf, ▪ Stimulsoft.Report.StiExportFormat.Excel2007, ▪ Stimulsoft.Report.StiExportFormat.Word2007, ▪ Stimulsoft.Report.StiExportFormat.Html, ▪ Stimulsoft.Report.StiExportFormat.Html5, ▪ Stimulsoft.Report.StiExportFormat.Document.
data	export data to the string or byte array representation.
preventDefault	a flag to prevent further processing of the event. The default value is "false".



onDesignReport

Is called by pressing the Design button. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event, by default it is "DesignReport".
fileName	an exporting file name.
report	a report for designing.
preventDefault	a flag to prevent further processing of the event. The default value is "false".

```
viewer.onDesignReport = function (event) {
    console.log("designing");
}
```

17.12 Viewer Options

The following options are used to customize the appearance and behavior of the Stimulsoft Report Viewer.

Stimulsoft.Viewer.StiViewerOptions:

Option	Description	Default value
width	Specifies the width of the component in "px" or "%"	"100%"
height	Specifies the height of the component in "px" or "%". Works only with options.appearance.scrollbarsMode = true;	String.empty

17.12.1 Appearance

Stimulsoft.Viewer.StiViewerOptions.appearance:

Option	Description	Default value
appearance.backgrounColor	Changes the background color of the viewer	Stimulsoft.System.Drawing.Color.white
appearance.pageBorderColor	Sets the border color of report pages	Stimulsoft.System.Drawing.Color.gray
appearance.rightToLeft	Sets the Right to Left mode for viewer controls	false
appearance.fullScreenMode	Sets the full screen mode of the report viewer. If set to true, then the width and height are ignored	false
appearance.scrollbarsMode	Sets the mode of displaying the report viewer without scroll bars or with them	false
appearance.openLinksTarget	Target window to open the link contained in the report	"_self"
appearance.openExportedReportTarget	Target window to open the export file from the viewer	"_blank"
appearance.showTooltips	Displays tips for the viewer controls when hovering the mouse	True
appearance.pageAlignment	Sets the position of the report page in the viewer window. <ul style="list-style-type: none"> • Stimulsoft.Viewer.StiContentAlignment.Left – the page is aligned by the left side of the viewer; • Stimulsoft.Viewer.StiContentAlignment.Center – the page is aligned by the center; • Stimulsoft.Viewer.StiContentAlignment.Right – the page will be aligned by the right side of the viewer. 	Stimulsoft.Viewer.StiContentAlignment.Center
appearance.showPageShadow	Hides the report page shadow	true
appearance.bookmarksPrint	Prints report bookmarks on the printer (in addition to the report)	false
appearance.bookmarksTreeWidth	Sets the width of the bookmark panel in pixels	180
appearance.parametersPanelMaxHeight	Sets the maximum height of the parameter panel in pixels	300

ht		
appearance.parametersPanelColumnsCount	Sets the number of columns for showing report parameters	2
appearance.parametersPanelDateFormat	Sets the date format and time for variables of the corresponding type on the parameters panel	String.empty
appearance.interfaceType	<p>Sets the type of the interface viewer. Can take the following values:</p> <ul style="list-style-type: none"> • Stimulsoft.Viewer.StiInterfaceType.Auto – the interface type of the report viewer is selected automatically depending on the device used; • Stimulsoft.Viewer.StiInterfaceType.Mouse – forced use the interface to manage viewer using the mouse device; • Stimulsoft.Viewer.StiInterfaceType.Touch – force use of the touch interface to manage viewer using the touch screen (mobile devices). 	Stimulsoft.Viewer.StiInterfaceType.Auto
appearance.chartRenderType	<p>Sets the type of drawing charts on the report page. It has the following values:</p> <ul style="list-style-type: none"> • Stimulsoft.Viewer.StiChartRenderType.AnimatedVector – charts are drawn in the vector mode with animation; • Stimulsoft.Viewer.StiChartRenderType.Vector – charts are drawn as vector images; • Stimulsoft.Viewer.StiChartRenderType.Image – charts are drawn as images. 	Stimulsoft.Viewer.StiChartRenderType.AnimatedVector

17.12.2 Toolbar

Stimulsoft.Viewer.StiViewerOptions.toolbar:

Option	Description	Default value
toolbar.visible	Shows/hides the toolbar of the report viewer	true
toolbar.backgroundColor	Changes the toolbar color	Stimulsoft.System.Drawing.Color.empty
toolbar.borderColor	Changes the color of the toolbar borders	Stimulsoft.System.Drawing.Color.empty
toolbar.fontSize	Changes the font size for the toolbar and menu	Stimulsoft.System.Drawing.Color.empty
toolbar.fontFamily	Changes the font family for the toolbar and menu	"Arial"

toolbar.alignment	Sets the position of the toolbar controls. <ul style="list-style-type: none">• Stimulsoft.Viewer.StiContentAlignment.Default – the items are placed depending on the value of the RightToLeft option;• Stimulsoft.Viewer.StiContentAlignment.Left – the items are aligned by the left side of the toolbar;• Stimulsoft.Viewer.StiContentAlignment.Center – the items are aligned by the center of the toolbar;• Stimulsoft.Viewer.StiContentAlignment.Right – the items are aligned by the right side of the toolbar.	Stimulsoft.Viewer.StiContentAlignment.Default
toolbar.showButtonsCaptions	Shows/hides the buttons of the viewer toolbar	true
toolbar.showPrintButton	Shows/hides the Print button on the toolbar of the report viewer	true
toolbar.showSaveButton	Shows/hides the Save button on the toolbar of the report viewer	true
toolbar.showSendEmailButton	Shows/hides the Send Email button on the toolbar of the report viewer	false
toolbar.showBookmarksButton	Shows/hides the Bookmarks button on the toolbar of the report viewer. If this button is hidden then the toolbar panel will not be shown even if bookmarks are present in the report.	true
toolbar.showParametersButton	Shows/hides the Parameters button on the toolbar of the report viewer. If this button is hidden then the toolbar panel will not be shown even if bookmarks are present in the report.	true
toolbar.showEditorButton	Shows/hides the Editor button on the report viewer toolbar. If the report does not have editable components, the button will be hidden regardless of the property value.	true
toolbar.showFullScreenButton	Shows/hides the Full Screen button on the toolbar of the report viewer	true
toolbar.showFirstPageButton	Shows/hides the First Page button on the toolbar of the report viewer	true
toolbar.showPreviousPageButton	Shows/hides the Previous Page button on the toolbar of the report viewer	true
toolbar.showCurrentPageControl	Shows/hides the indicator of the current report page	true
toolbar.showNextPageButton	Shows/hides the Next Page button on the toolbar of the report viewer	true
toolbar.showLastPageButton	Shows/hides the Last Page button on the toolbar of the report viewer	true

toolbar.showZoomButton	Shows/hides the button for selecting the report zoom	true
toolbar.showViewModeButton	Shows/hides the button for selecting viewing modes of report pages	true
toolbar.showDesignButton	Shows/hides the Design button on the toolbar of the report viewer	true
toolbar.showAboutButton	Shows/hides the About button on the toolbar of the report viewer	true
toolbar.viewMode	Sets the mode of showing report pages. <ul style="list-style-type: none">• Stimulsoft.Viewer.StiWebViewMode.OnePage – shows one page of the report selected in the toolbar of the viewer;• Stimulsoft.Viewer.StiWebViewMode.WholeReport – displays all report pages at once.	Stimulsoft.Viewer.StiWebViewMode.OnePage
toolbar.zoom()	This method sets zoom of report pages. By default, it is 100 percent. Values vary from 10 to 500 percent. You can use the predefined values: <ul style="list-style-type: none">• Stimulsoft.Viewer.StiZoomMode.PageWidth – when you run the viewer, the zoom will be set to display the report by the page width.• Stimulsoft.Viewer.StiZoomMode.PageHeight – when you run the viewer, the zoom will be set to display the report by the page height.	100
toolbar.menuAnimation	Disables animation when showing/hiding the menu of the report viewer	true
toolbar.showMenuMode	Sets the mode of showing the menu	Stimulsoft.Viewer.StiShowMenuMode.Click

17.12.3 Exports

Stimulsoft.Viewer.StiViewerOptions.exports:

Option	Description	Default value
exports.showExportDialog	Shows/hides the export parameters dialog. If the dialog window is disabled then exporting will be processed with the standard settings.	true
exports.showExportToDocument	Shows/hides the menu item Document File	true
exports.showExportToPdf	Shows/hides the menu item Adobe PDF File	false

exports.showExportToHtml	Shows/hides the menu item HMTL File	true
exports.showExportToWord2007	Shows/hides the menu item Microsoft Word 2007/2010 File.	false
exports.showExportToExcel2007	Shows/hides the menu item Microsoft Excel 2007/2010 File.	false

For example, run customized viewer with some options:

```
<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.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>
```

17.13 Running Report Designer

Stimulsoft Report Designer is an application for working with reports. It can be easily integrated into any web page (requires any web server). To get started, you must connect **CSS** and **scripts**:

```
<link href="stimulsoft.designer.office2013.css" rel="stylesheet">
<script src="stimulsoft.reports.js" type="text/javascript"></script>
<script src="stimulsoft.designer.js" type="text/javascript"></script>
```

Then load the required report and insert it into the HTML page code to deploy Stimulsoft Report Designer in the current **DOM** element:

```

var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");

var designer = new Stimulsoft.Designer.StiDesigner();
designer.report = report;

```

Stimulsoft Report Designer is an object created by the **Stimulsoft.Viewer.StiDesigner()** constructor, which has three optional parameters. The first parameter is a set of **Stimulsoft.Designer.StiDesignerOptions** divided into categories toolbar, bands, cross bands, components and dictionary. Some of them are listed in the table below. The second parameter is an identifier of the designer as a **DOM** object. The third parameter is a flag of rendering the designer in the moment of creation. If this flag is set to true, the designer will be drawn in the same place of the **DOM** tree, which houses the code. If the flag is set to false, the designer will be drawn where there is the "renderHtml()" method of the designer. For example, Initialization of the designer in the head of a page:

```

<script type="text/javascript">
    var designer = new Stimulsoft.Designer.StiDesigner(null,
        "StiDesigner", false);
</script>

```

Rendering of the designer control in div:

```

<div>Page content</div>
<div>
    <script type="text/javascript">
        // Render the report designer in this place
        designer.renderHtml();
    </script>
</div>

```

Another way to display the designer in a particular control is using the optional method parameter **renderHtml(elementId)**, which indicates the ID of the HTML element. Initialization of the designer in the head of a page:

```

<script type="text/javascript" >
    var designer = new Stimulsoft.Designer.StiDesigner(null,
        "StiDesigner", false);
    designer.renderHtml("designerContent");
</script>

```

This will display the designer in the element:

```
<div id="designerContent"></div>
```

Note: a designer can be hidden by specifying the values of the property "visible":

```
designer.visible = false;
```

17.14 Designer Events

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:

 onBeginPr ocessData	 onSaveRe port	 onCreateR eport	 onExit
 onEndProc cessData	 onOpenRe port	 onPreview Report	 Viewer Events



onBeginProcessData

Is called before requesting the data for the report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event, by default is "BeginProcessData".
command	a string ID for the current command. Can have values "TestConnection" and "ExecuteQuery" to call the command for test connection and data acquisition respectively.
connectionString	a connection string to the data source for the report.
queryString	a string with the SQL query to the database for data acquisition, used only with the command = "ExecuteQuery".
database	a string name of the current database.
preventDefault	a flag to prevent further processing of the event. The default value is "false".

```
designer.onBeginProcessData = function (event) {
    event.connectionString = "server=real.server; uid=root;
password=actual_password; database=production_db;";
}
```



onEndProcessData

Is called after retrieving the data for the report. The event handler argument "event" is an object with the next fields:

Name	Description
------	-------------

event	a string ID for the current event. By default it is "EndProcessData".
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.
preventDefault	a flag to prevent the further processing of the event. The default value is "false".

This event can be used to add its own data sources generated "on the fly" to the report.



onSaveReport

Is called before saving the report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "SaveReport".
fileName	a saving file name.
report	a report for saving.
preventDefault	a flag to prevent further processing of the event. The default value is "true".

```
designer.onSaveReport = function (event) {
    var jsonStr = event.report.saveToJsonString();
    console.log("saving a report");
}
```



onOpenReport

Is called before user click button for opening a report. The event handler argument "event" is an object with the next fields:

Name	Description
event	a string ID for current event. By default it is "OpenReport".
preventDefault	a flag to prevent further processing of the event. The default value is "true".



onCreateReport

Is called after a new report is created. The event handler argument "event" it is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is "CreateReport".
report	report for saving.

```
Designer.onCreateReport = function (event) {
    var dataSet = new Stimulsoft.System.Data.DataSet("Demo");
    dataSet.readJsonFile("/reports/Demo.json");
    event.report.regData("Demo", "Demo", dataSet);
    console.log("creating a new report");
}
```



onPreviewReport

is called before the launch of the preview when requesting data for the report. If the data source is set in the report template, then calling “report.regData()” with the same name as the data source has a lower priority, and data will be taken from the source specified in the report template. It is recommended to clean the collection of data sources by “report.dictionary.databases.clear()”. The event handler argument “event” is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is “PreviewReport”.
fileName	a report file name.
report	report for preview.
preventDefault	a flag to prevent further processing of the event. The default value is “false”.

```
designer.onPreviewReport = function (event) {
    switch (event.format) {
        case Stimulsoft.Report.StiExportFormat.Html:
            event.settings.zoom = 2; // Set zoom to 200%
            break;
    }
    console.log("running the preview");
}
```



onExit

Is called before closing the designer. The event handler argument “event” is an object with the next fields:

Name	Description
event	a string ID for the current event. By default it is “Exit”.

```
designer.onExit = function (event) {
    console.log("exiting");
}
```

17.15 Designer Options

The following options are used to customize the appearance and behavior of Stimulsoft Report Designer.

Stimulsoft.Designer.StiDesignerOptions:

Option	Description	Default value
Width	Specifies the width of the component in "px" or "%"	"100%"
Height	Specifies the height of the component in "px" or "%"	"800px"

17.15.1 Appearance

Stimulsoft.Designer.StiDesignerOptions.appearance:

Option	Description	Default value
defaultUnit	A default value of unit in the designer. It has the following values: <ul style="list-style-type: none">• Stimulsoft.Report.StiReportUnitType.Centimeters,• Stimulsoft.Report.StiReportUnitType.HundredthsOfInch,• Stimulsoft.Report.StiReportUnitType.Inches,• Stimulsoft.Report.StiReportUnitType.Millimeters	Stimulsoft.Report.StiReportUnitType.Centimeters
interfaceType	The type of the designer interface. It has the following values: <ul style="list-style-type: none">• Stimulsoft.Designer.StiInterfaceType.Auto,• Stimulsoft.Designer.StiInterfaceType.Mouse,• Stimulsoft.Designer.StiInterfaceType.Touch	Stimulsoft.Designer.StiInterfaceType.Auto
showAnimation	Show menu animation for designer	true
showSaveDialog	Displays the dialog to enter the name of the report when it is saved	true
showTooltips	Displays tips for the designer controls when hovering the mouse	true
showTooltipsHelp	Displays a link to the online documentation on the tips	true
appearance.fullScreenMode	Sets the full screen mode of the report designer. If set to true then width and height are ignored	false

17.15.2 Toolbar

Stimulsoft.Designer.StiDesignerOptions.toolbar:

Option	Description	Default value
showPageButton	Shows/hides the Page button in the toolbar of the designer	true
showPreviewButton	Shows/hides the Preview button in the toolbar of the designer	true
showSaveButton	Shows/hides the Save button in the toolbar of the designer	true
showAboutButton	Shows/hides the About button in the toolbar of the designer	true
showFileMenu	Shows/hides the file menu of the designer	true
showFileMenuNew	Shows/hides the New item in the file menu of the designer	true
showFileMenuOpen	Shows/hides the Open item in the file menu of the designer	true
showFileMenuSave	Shows/hides the Save item in the file menu of the designer	true
showFileMenuClose	Shows/hides the Close item in the file menu of the designer	true
showFileMenuExit	Shows/hides the Exit item in the file menu of the designer	true
showFileMenuReportSetup	Shows/hides the Report Setup item in the file menu of the designer	true
showFileMenuOptions	Shows/hides the Options item in the file menu of the designer	true

17.15.3 Bands

Stimulsoft.Designer.StiDesignerOptions.bands:

Option	Description	Default value
showReportTitleBand	Shows/hides the ReportTitleBand item in the bands menu of the designer	true
showReportSummaryBand	Shows/hides the ReportSummaryBand item in the bands menu of the designer	true
showPageHeaderBand	Shows/hides the PageHeaderBand item in the bands menu of the designer	true
showPageFooterBand	Shows/hides the PageFooterBand item in the bands menu of the designer	true
showGroupHeaderBand	Shows/hides the GroupHeaderBand item in the bands menu of the designer	true

showHeaderBand	Shows/hides the HeaderBand item in the bands menu of the designer	true
showFooterBand	Shows/hides the FooterBand item in the bands menu of the designer	true
showColumnHeaderBand	Shows/hides the ColumnHeaderBand item in the bands menu of the designer	true
showColumnFooterBand	Shows/hides the ColumnFooterBand item in the bands menu of the designer	true
showDataBand	Shows/hides the DataBand item in the bands menu of the designer	true
showHierarchicalBand	Shows/hides the HierarchicalBand item in the bands menu of the designer	true
showChildBand	Shows/hides the ChildBand item in the bands menu of the designer	true
showEmptyBand	Shows/hides the EmptyBand item in the bands menu of the designer	true
showOverlayBand	Shows/hides the OverlayBand item in the bands menu of the designer	true
showTable	Shows/hides the Table item in the bands menu of the designer	true

17.15.4 Cross-Bands

Stimulsoft.Designer.StiDesignerOptions.crossBands:

Option	Description	Default value
showCrossTab	Shows/hides the CrossTab item in the cross bands menu of the designer	false
showCrossGroupHeaderBand	Shows/hides the CrossGroupHeaderBand item in the cross bands menu of the designer	false
showCrossGroupFooterBand	Shows/hides the CrossGroupFooterBand item in the cross bands menu of the designer	false
showCrossHeaderBand	Shows/hides the CrossHeaderBand item in the cross bands menu of the designer	false
showCrossFooterBand	Shows/hides the CrossFooterBand item in the cross bands menu of the designer	false
showCrossDataBand	Shows/hides the CrossDataBand item in the cross bands menu of the designer	false

17.15.5 Components

Stimulsoft.Designer.StiDesignerOptions.components:

Option	Description	Default value
showText	Shows/hides the Text item in the components menu of the designer	true
showTextInCells	Shows/hides the TextInCells item in the components menu of the designer	false
showRichText	Shows/hides the RichText item in the components menu of the designer	false
showImage	Shows/hides the Image item in the components menu of the designer	true
showBarCode	Shows/hides the BarCode item in the components menu of the designer	false
showShape	Shows/hides the Shape item in the components menu of the designer	true
showPanel	Shows/hides the Panel item in the components menu of the designer	true
showClone	Shows/hides the Clone item in the components menu of the designer	false
showCheckBox	Shows/hides the CheckBox item in the components menu of the designer	true
showSubReport	Shows/hides the SubReport item in the components menu of the designer	true
showZipCode	Shows/hides the ZipCode item in the components menu of the designer	false
showChart	Shows/hides the Chart item in the components menu of the designer	true

17.15.6 Dictionary

Stimulsoft.Designer.StiDesignerOptions.dictionary

Almost all options described below can take the following values:



Stimulsoft.Designer.StiDesignerPermissions.None - deny all;

- Stimulsoft.Designer.StiDesignerPermissions.Create – allows creating an item;
- Stimulsoft.Designer.StiDesignerPermissions.Delete – allows deleting an item;
- Stimulsoft.Designer.StiDesignerPermissions.Modify – allows modifying an item;
- Stimulsoft.Designer.StiDesignerPermissions.View – allows viewing an item;
- Stimulsoft.Designer.StiDesignerPermissions.ModifyView – allows modifying and viewing an item (Modify + View);
- Stimulsoft.Designer.StiDesignerPermissions.All – allows any action with the item (Create + Delete + Modify + View).

Option	Description	Default value
showDictionary	Shows/hides the dictionary in the designer	true
dataSourcesPermissions	A value of permissions for data sources in the designer	Stimulsoft.Designer.StiDesignerPermissions.All
dataConnectionsPermissions	A value of connections for data sources in the designer	Stimulsoft.Designer.StiDesignerPermissions.All
dataColumnsPermissions	A value of connections for columns in the designer	Stimulsoft.Designer.StiDesignerPermissions.All
dataRelationsPermissions	A value of connections for relations in the designer	Stimulsoft.Designer.StiDesignerPermissions.All
businessObjectsPermissions	A value of connections for business objects in the designer	Stimulsoft.Designer.StiDesignerPermissions.All
variablesPermissions	A value of connections for variables in the designer	Stimulsoft.Designer.StiDesignerPermissions.All

For example, run the designer on the full screen of a browser:

```
<script type="text/javascript">
    var report = new Stimulsoft.Report.StiReport();
    report.loadFile("SimpleList.mrt");

    var options = new Stimulsoft.Designer.StiDesignerOptions;
```

```

options.appearance.fullScreenMode = true;

var designer = new Stimulsoft.Designer.StiDesigner(options);
designer.report = report;
</script>
```

17.16 Exporting Rendered Report

A rendered report can be exported to several formats. This example demonstrates the export to HTML:

```

var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.render();

var settings = new Stimulsoft.Report.Export.StiHtmlExportSettings();
var service = new Stimulsoft.Report.Export.StiHtmlExportService();

var textWriter = new Stimulsoft.System.IO.TextWriter();
var htmlTextWriter = new
Stimulsoft.Report.Export.StiHtmlTextWriter(textWriter);

service.exportTo(report, htmlTextWriter, settings);

var resultHtml = textWriter.getStringBuilder().toString();
```

It uses one of the export services, in this case it is **Stimulsoft.Report.Export.StiHtmlExportService**. The “**exportTo()**” method produces output in one of the parameters (htmlTextWriter) the flow of the report in HTML. After executing the code, “resultHtml” contains a string with the contents of the HTML report.

Options for exporting to HTML are defined in the **Stimulsoft.Report.Export.StiHtmlExportSettings** object and passed as a parameter to the “**exportTo()**” method of the **Stimulsoft.Report.Export.StiHtmlExportService** object. These options allow you to customize the look and the structure of the HTML document.

Table of properties of **Stimulsoft.Report.Export.StiHtmlExportSettings**:

Option	Description	Default value
htmlType	A type of the report file to be converted into. Can have the following values: <ul style="list-style-type: none"> • Stimulsoft.Report.StiHtmlType.Html, • Stimulsoft.Report.StiHtmlType.Html5, • Stimulsoft.Report.StiHtmlType.Mht 	Stimulsoft.Report.StiHtmlType.Html
imageQuality	Quality of images which will be exported to the result file	0.75

imageResolution	Resolution of images in DPI which will be exported to the result file	100
imageFormat	Sets an image export format. Can have the following values: <ul style="list-style-type: none"> • Stimulsoft.Report.ImageFormat.Bmp, • Stimulsoft.Report.ImageFormat.Emf, • Stimulsoft.Report.ImageFormat.Exif, • Stimulsoft.Report.ImageFormat.Gif, • Stimulsoft.Report.ImageFormat.Guid, • Stimulsoft.Report.ImageFormat.Icon, • Stimulsoft.Report.ImageFormat.Jpeg, • Stimulsoft.Report.ImageFormat.MemoryBmp, • Stimulsoft.Report.ImageFormat.Png, • Stimulsoft.Report.ImageFormat.Tiff, • Stimulsoft.Report.ImageFormat.Wmf 	null
encoding	File encoding. Can have the following values: <ul style="list-style-type: none"> • Stimulsoft.System.Text.Encoding.ASCII, • Stimulsoft.System.Text.Encoding.BigEndianUnicode, • Stimulsoft.System.Text.Encoding.Default (set to Stimulsoft.System.Text.Encoding.Unicode), • Stimulsoft.System.Text.Encoding.Unicode, • Stimulsoft.System.Text.Encoding.UTF32, • Stimulsoft.System.Text.Encoding.UTF7, • Stimulsoft.System.Text.Encoding.UTF8 	Stimulsoft.System.Text.Encoding.Default
zoom	Zoom factor. By default the value is set to 1.0 that is equal 100% in the export settings dialog	1
exportMode	Sets the mode of the document export using the div, span or table element. Can have the following values: <ul style="list-style-type: none"> • Stimulsoft.Report.Export.StiHtmlExportMode.Span, • Stimulsoft.Report.Export.StiHtmlExportMode.Div, • Stimulsoft.Report.Export.StiHtmlExportMode.Table 	Stimulsoft.Report.Export.StiHtmlExportMode.Table
exportQuality	Export quality of the component size. Can have the following values: <ul style="list-style-type: none"> • Stimulsoft.Report.Export.StiHtmlExportQuality.High, • Stimulsoft.Report.Export.StiHtmlExportQuality.Low 	Stimulsoft.Report.Export.StiHtmlExportQuality.High
addPageBreaks	Adds page breaks	true
bookmarksTreeWidth	Bookmark column width, in pixels	150
exportBookmarksMode	An exporting mode of a document with bookmarks. Can have the following values: <ul style="list-style-type: none"> • Stimulsoft.Report.Export.StiHtmlExportBookmarksMode.BookmarksOnly, • Stimulsoft.Report.Export.StiHtmlExportBookmarksMode.ReportOnly, • Stimulsoft.Report.Export.StiHtmlExportBookmarksMode.All 	Stimulsoft.Report.Export.StiHtmlExportBookmarksMode.All

useStylesTable	Uses the Styles Table; if false then the style table is empty and all properties of each component will be described directly in the style of this component	true
removeEmptySpaceAtBottom	Indicates that table styles will be used in the result HTML file. The HTML5 and MHT export mode is not supported.	true
pageHorAlignment	The horizontal alignment of pages. The HTML5 and MHT export mode is not supported. Can have the following values: <ul style="list-style-type: none">• Stimulsoft.Base.Drawing.StiHorAlignment.Left,• Stimulsoft.Base.Drawing.StiHorAlignment.Center,• Stimulsoft.Base.Drawing.StiHorAlignment.Right	Stimulsoft.Base.Drawing.StiHorAlignment.Left
compressToArchive	Provides the ability (when exporting to HTML) to get the zip file after conversion. If this flag set to is true, the report processing occurs first, and then all the files and folders will be packed in a zip archive	false
useEmbeddedImages	Provides the ability to embed images directly into the HTML file. In this case, it is necessary to consider that the correct displaying of this file depends on the browser being used. Not all browsers support the option to view the HTML file with embedded pictures	false
continuousPages	Indicates that all report pages will be shown as vertical ribbon. The HTML and MHT export mode is not supported	true
chartType	A type of the chart in the HTML exports. Can have the following values: <ul style="list-style-type: none">• Stimulsoft.Report.Export.StiHtmlChartType.Image,• Stimulsoft.Report.Export.StiHtmlChartType.Vector,• Stimulsoft.Report.Export.StiHtmlChartType.AnimatedVector	Stimulsoft.Report.Export.StiHtmlChartType.AnimatedVector
openLinksTarget	A target (string) to open links from the exported report	null

This example demonstrates the export to PDF:

```

var report = new Stimulsoft.Report.StiReport();
report.loadFile("SimpleList.mrt");
report.render();

var settings = new Stimulsoft.Report.Export.StiPdfExportSettings();
var service = new Stimulsoft.Report.Export.StiPdfExportService();

var stream = new Stimulsoft.System.IO.MemoryStream();

service.exportTo(report, stream, settings);

```

```
var data = stream.toArray();

Object.saveAs(data, "SimpleList.pdf", "application/pdf");
```

It uses the export service **Stimulsoft.Report.Export.StiPdfExportService**. The “**exportTo()**” method produces output in one of the parameters (stream) the flow of the report in PDF. After executing the code, “data” contains a byte array with the contents of the PDF report.

Options for exporting to PDF are defined in the **Stimulsoft.Report.Export.StiPdfExportService** object and passed as a parameter to the “**exportTo()**” method of the **Stimulsoft.Report.Export.StiPdfExportService** object. These options allow you to customize the look and the structure of the PDF document.

Table of properties of **Stimulsoft.Report.Export.StiPdfExportService**:

Option	Description	Default value
imageQuality	Quality of images which will be exported to the result file	0.75
imageResolution	Resolution of images in DPI which will be exported to the result file	100
imageResolutionMode	Image resolution mode. Can have the following values: • Stimulsoft.Report.Export.StiImageResolutionMode.Exactly, • Stimulsoft.Report.Export.StiImageResolutionMode.NoMoreThan, • Stimulsoft.Report.Export.StiImageResolutionMode.Auto	Stimulsoft.Report.Export.StiImageResolutionMode.Exactly
embeddedFonts	Indicates that fonts which used in report will be included in PDF file	false
standardPdfFonts	Indicates that only standard PDF fonts will be used in result PDF file	false
compressed	Indicates that result file will be used compression	false
useUnicode	Indicates that Unicode symbols must be used in result PDF file	false

18 Java Viewer

The **StiViewerFx** component is delivered as a part of StimulsoftReports.Fx for 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.

18.1 Showing Reports

Add the **StiViewerFx** component on the required component (for example on JFx):

```
StiViewerFx viewerfx = new StiViewerFx(parentFrame);
fx.add(viewerfx);
```

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:

```
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:

```
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.

18.2 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 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:

```

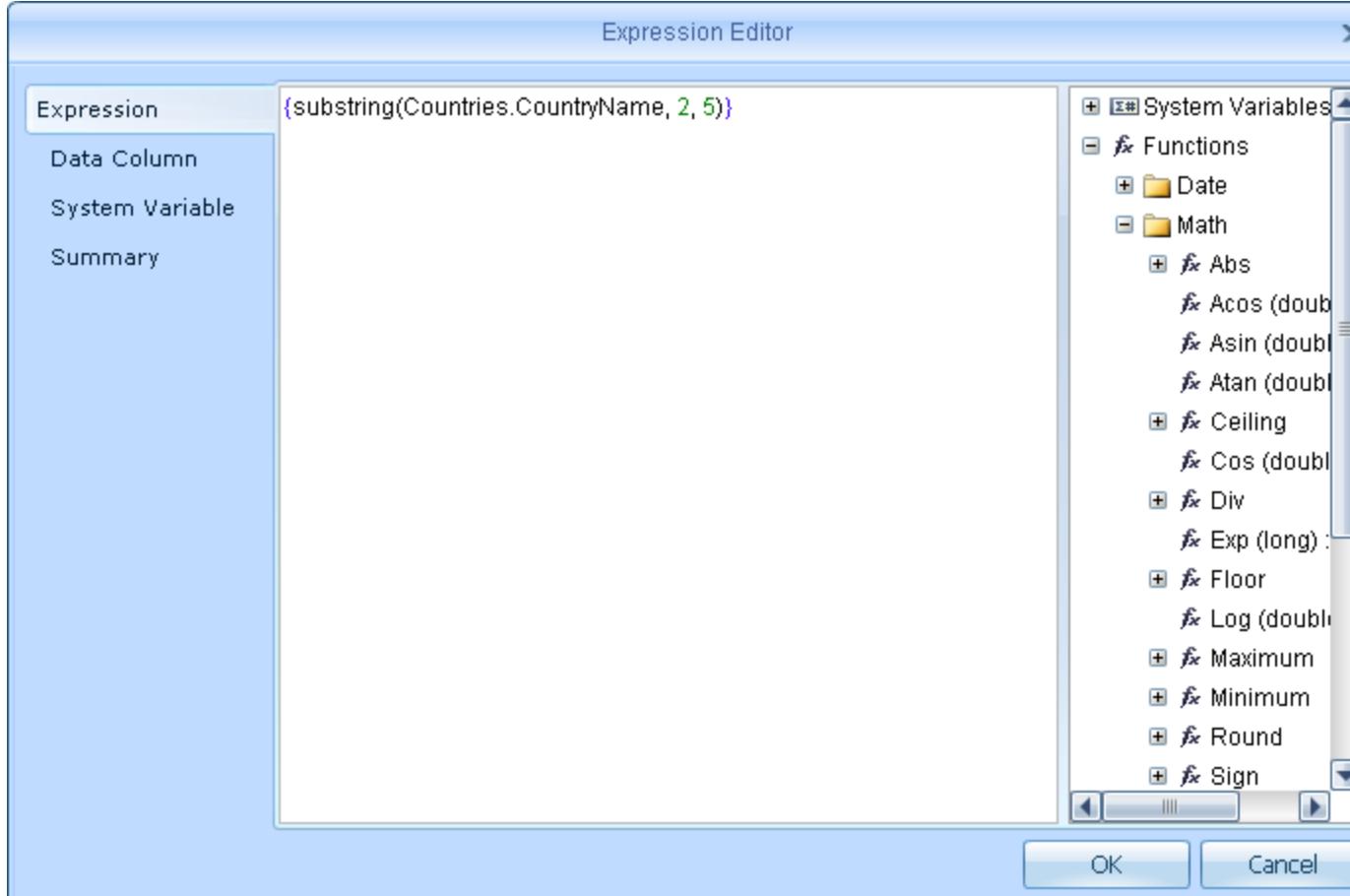
public StiReport render(StiReport report) throws IOException, StiException {
    report.getCustomFunctions().add(new StiCustomFunction() {
        public Object invoke(List<Object> args) {
            return ((String) args.get(0)).substring((Integer)
args.get(1), (Integer) args.get(2));
        }

        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:



19 Java with Flex Client

Using a report designer and viewer in the Web application. Flexible setup of an application using the configuration file. Loading and saving reports from the file system or a database. Ability to override mechanisms for loading and saving reports. Ability to use a set of custom JSP tags.

19.1 Installation

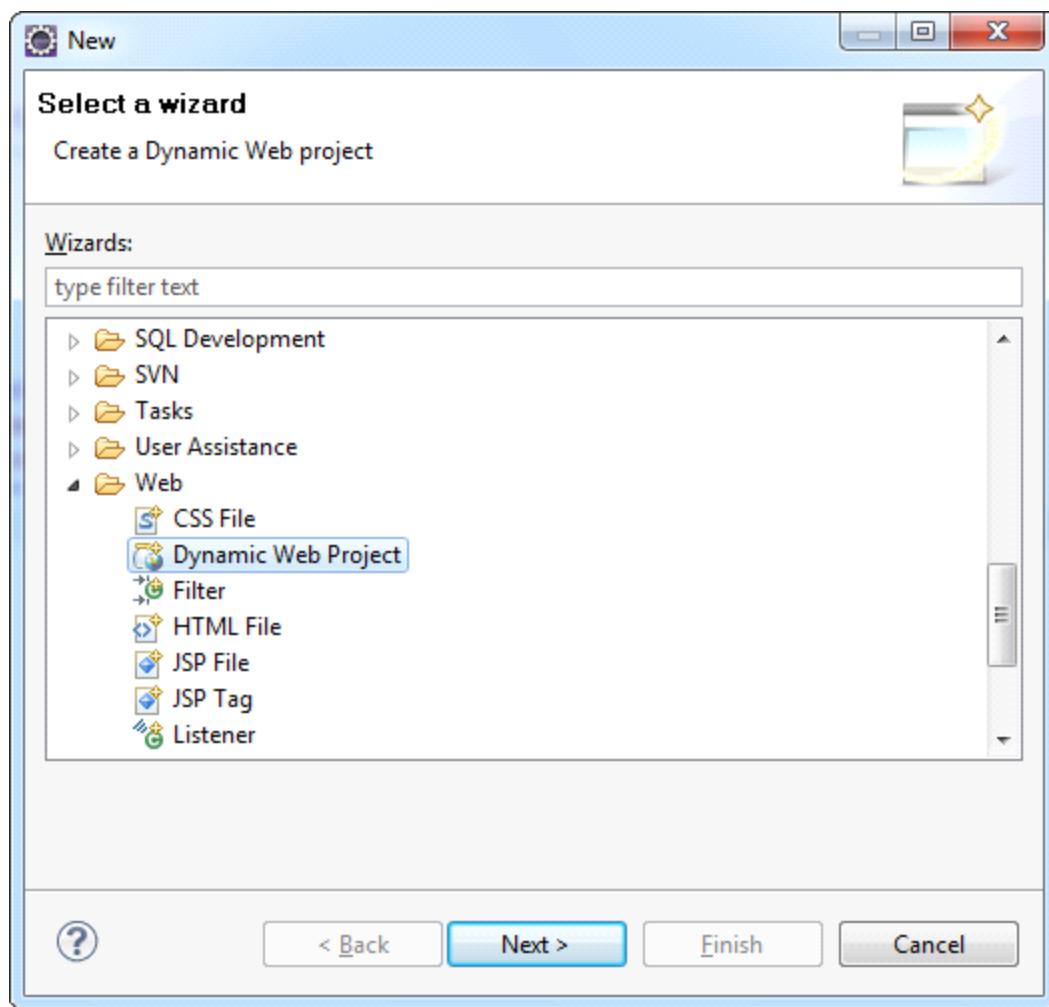
Installation:

- ▷ 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.com.

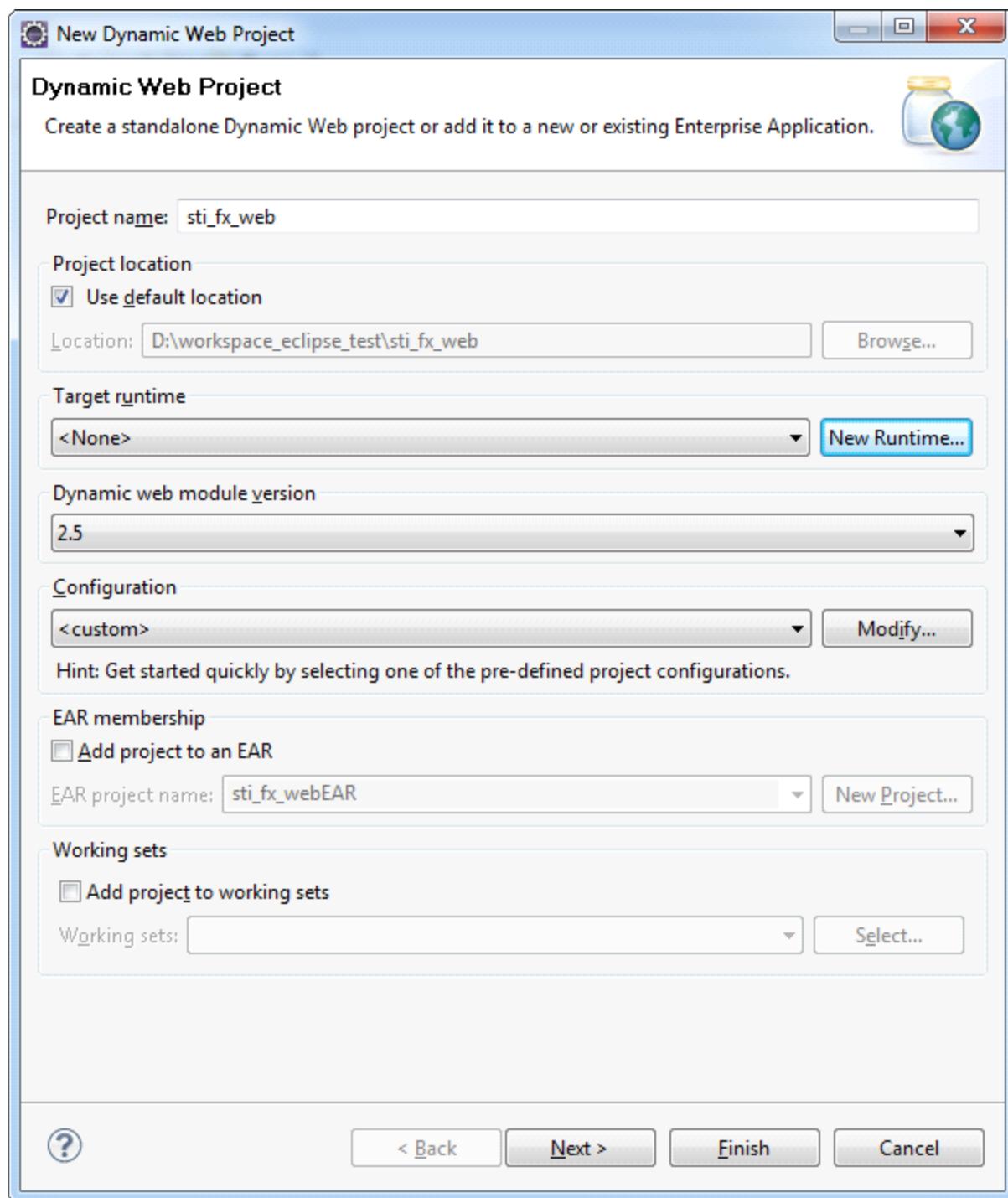
19.2 Creating Project

Creating a Web 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 **Next** (See the picture below):



In the window opened fill in the **Project name** (e.g. **sti_fx_web**, as shown in the picture below). Then configure the web server, on which the application will run.

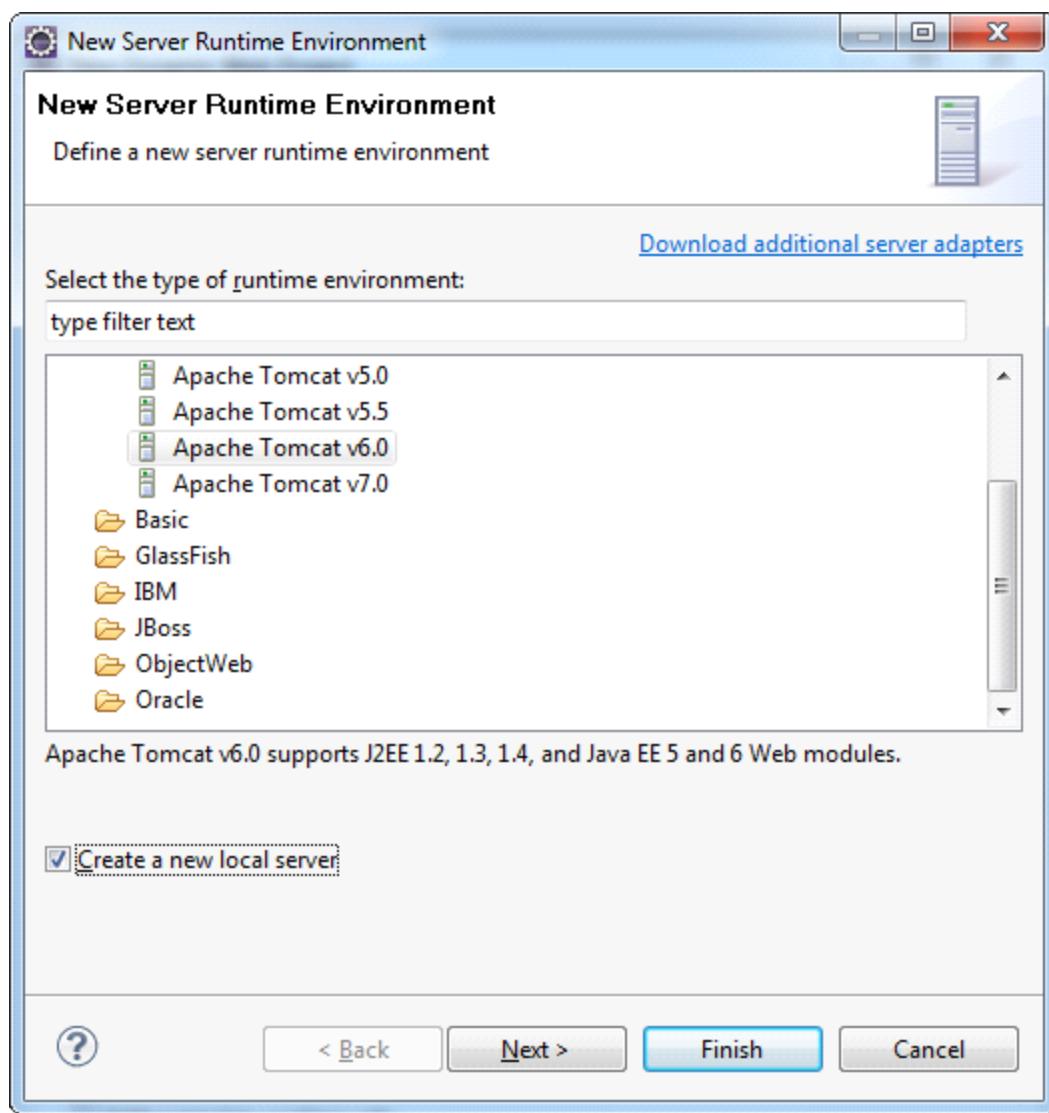


19.3 Creating Server

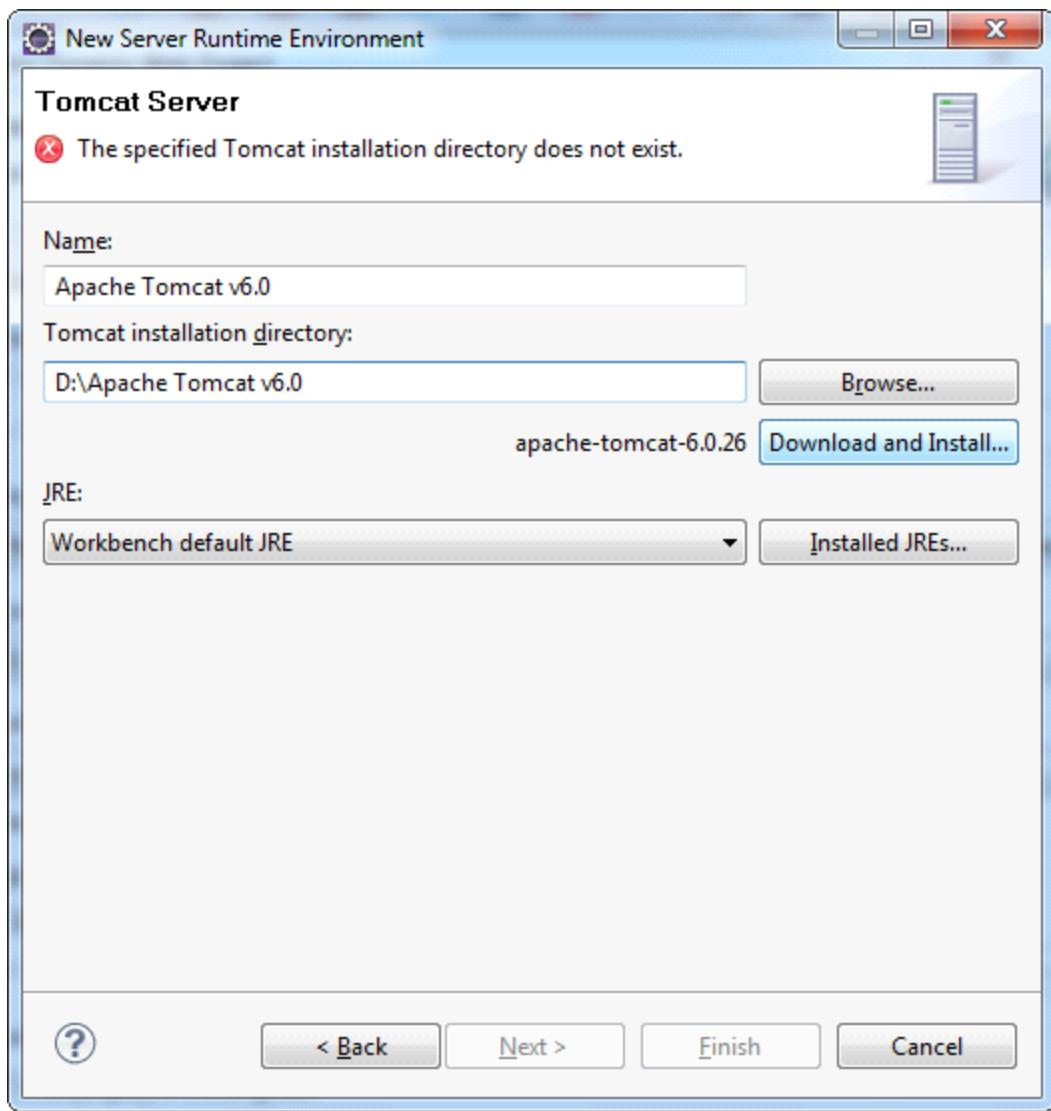
Target a runtime:

Under Target Runtime, you see <None>, as shown in the picture below, 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 list. Check Create a new local server as shown on Figure 3, then click Next.

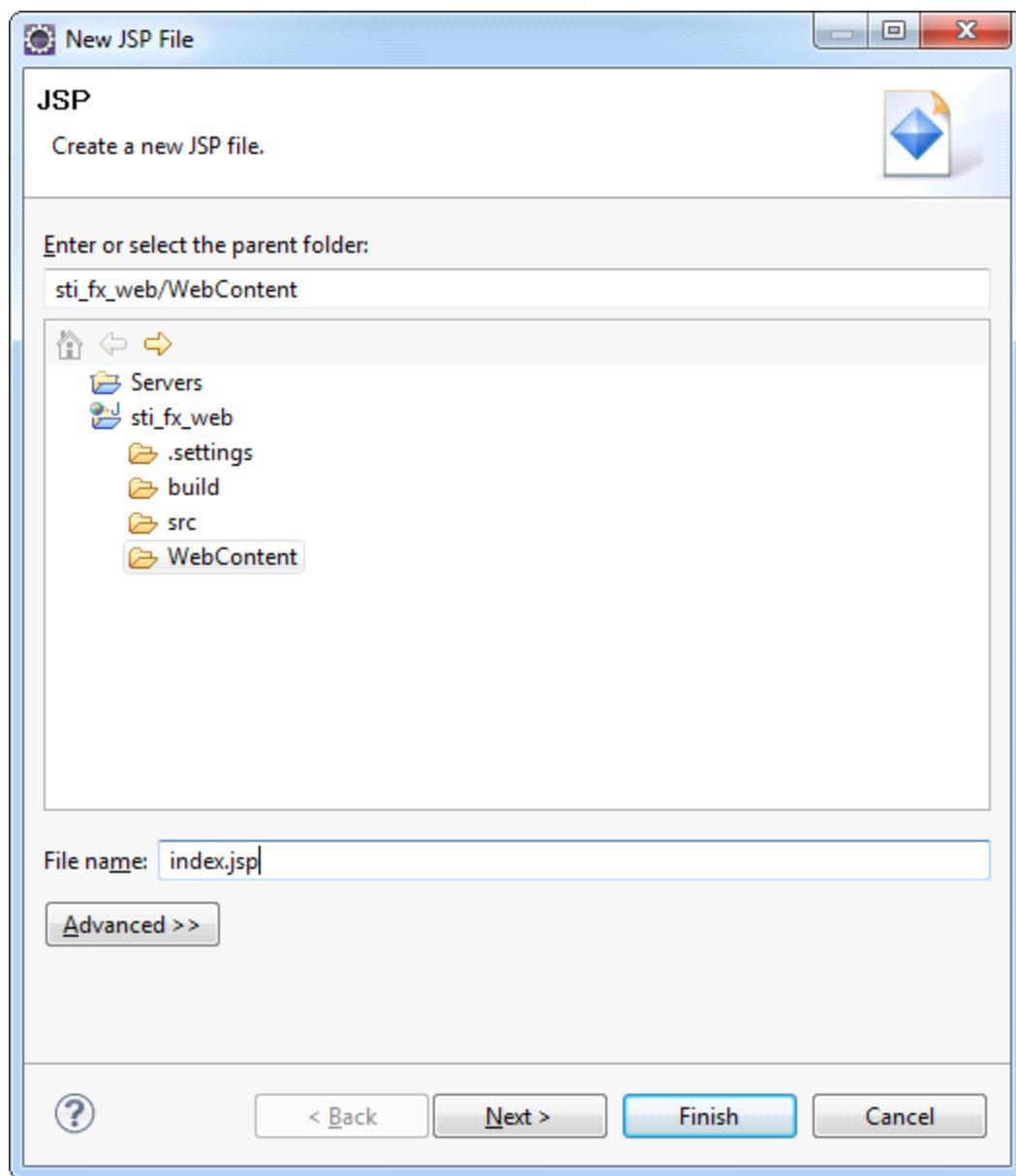


Then define the Tomcat installation directory, in which Apache Tomcat is installed, or in which one needs to install it, as shown in the picture. If it is not installed, then click Download and Install. After all fields are specified, click Finish.



19.4 Creating Sample

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 the picture) define the directory Web Content, and in the **File name** write **index.jsp**. Click **Finish** to create pages using the default template.

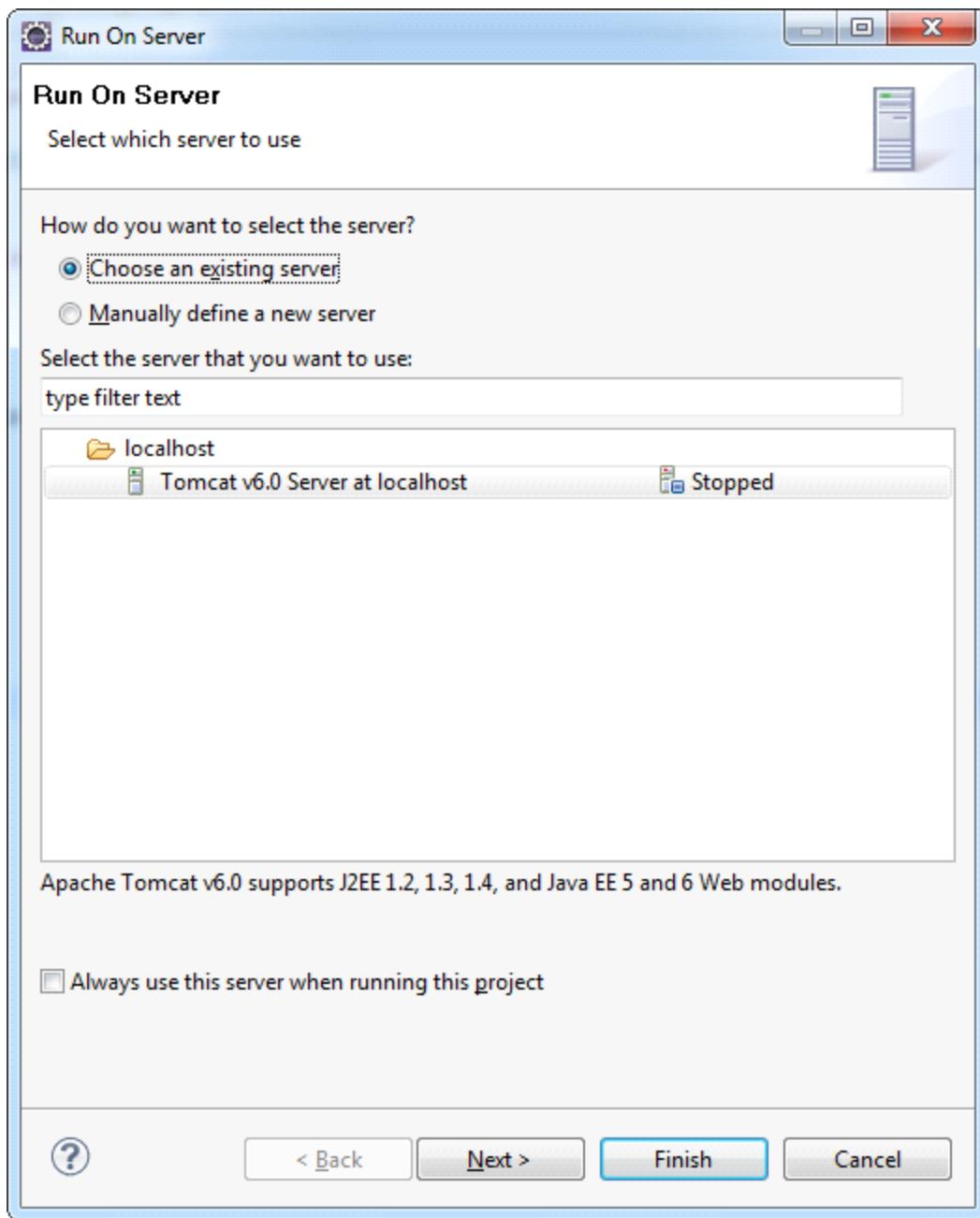


Now open the **index.jsp** and edit it so that it displays the current date. The code page is specified in the code:

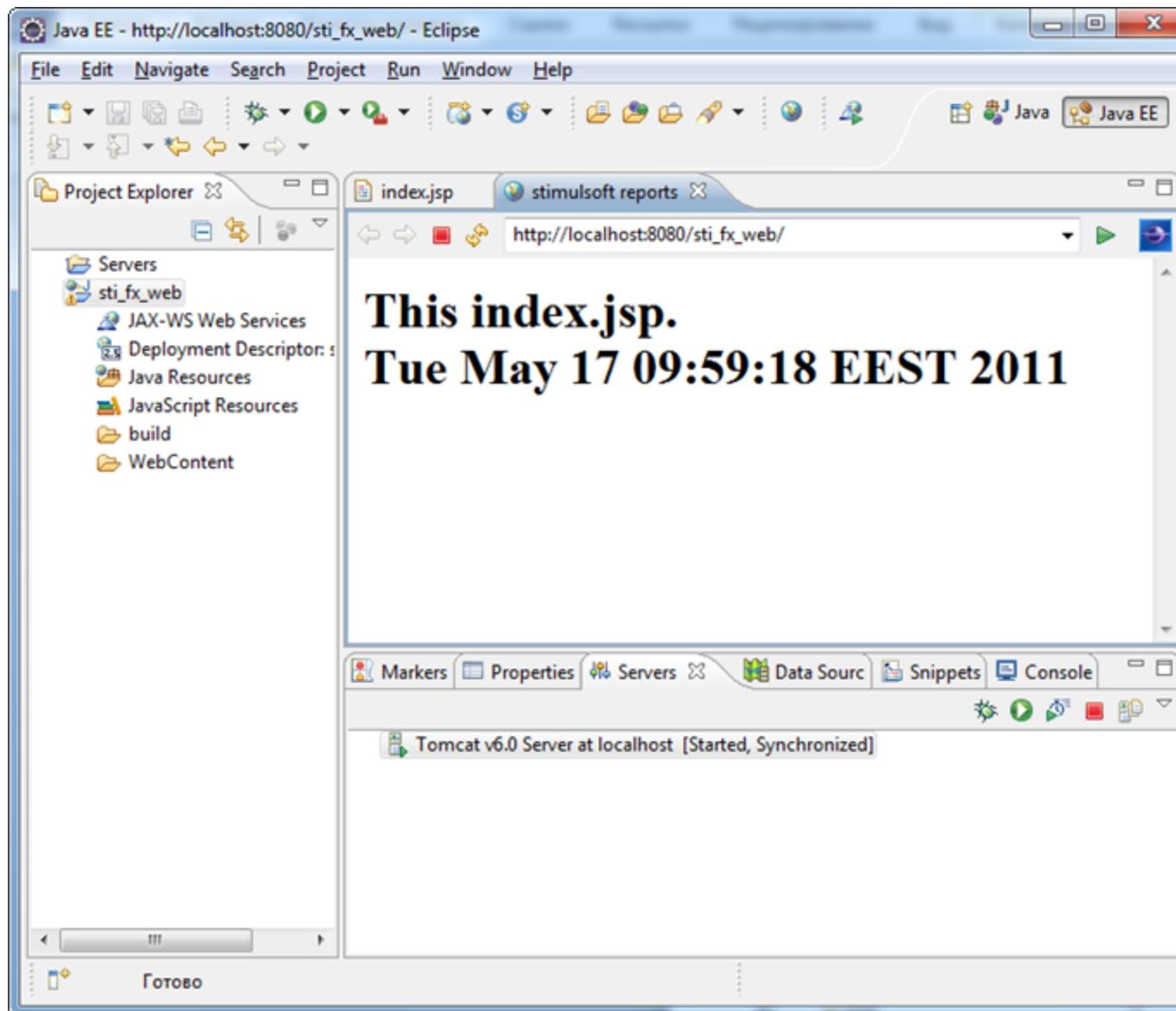
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>stimulsoft reports</title>
</head>
<body>
<% java.util.Date date = new java.util.Date(); %>
```

```
<h1>
    This index.jsp.<br>
    <%=date.toString()%>
</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**.

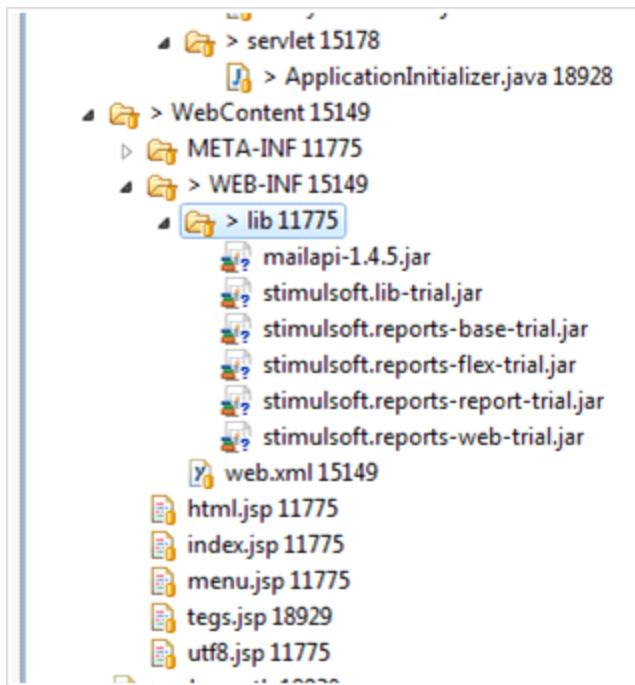


As a result, you receive the following (see the picture below). This page will be available from any browser at <http://localhost:8080/{ProjectName}> (where the **{ProjectName}** name of the created project, in our case **sti_fx_web**).



19.5 Creating Sample Page with Report Designer

Create a simple page with a report designer. 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. As a result, one can see the following (picture below):



Next, open the **web.xml** for editing, it should look like in Listing 2:

```

<?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.StiWebViewActionServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>StimulsoftAction</servlet-name>
    </servlet-mapping>

```

```

<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 (see the code below).

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<%@ taglib uri="http://stimulsoft.com/designer" prefix="stidesignerfx" %>
<%@ taglib uri="http://stimulsoft.com/viewer" prefix="stiviewerfx" %>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Stimulsoft Reports.Fx for Java</title>
</head>
<body>
<h1 align="center">My first report!</h1>
<stidesignerfx:iframe
    width="100%" height="90%" align="middle"
    styleClass="" frameborder="0" styleId=""
    marginheight="4" marginwidth="10" name="stiviewer"
    scrolling="no" style="" title="report"/>
</body>
</html>

```

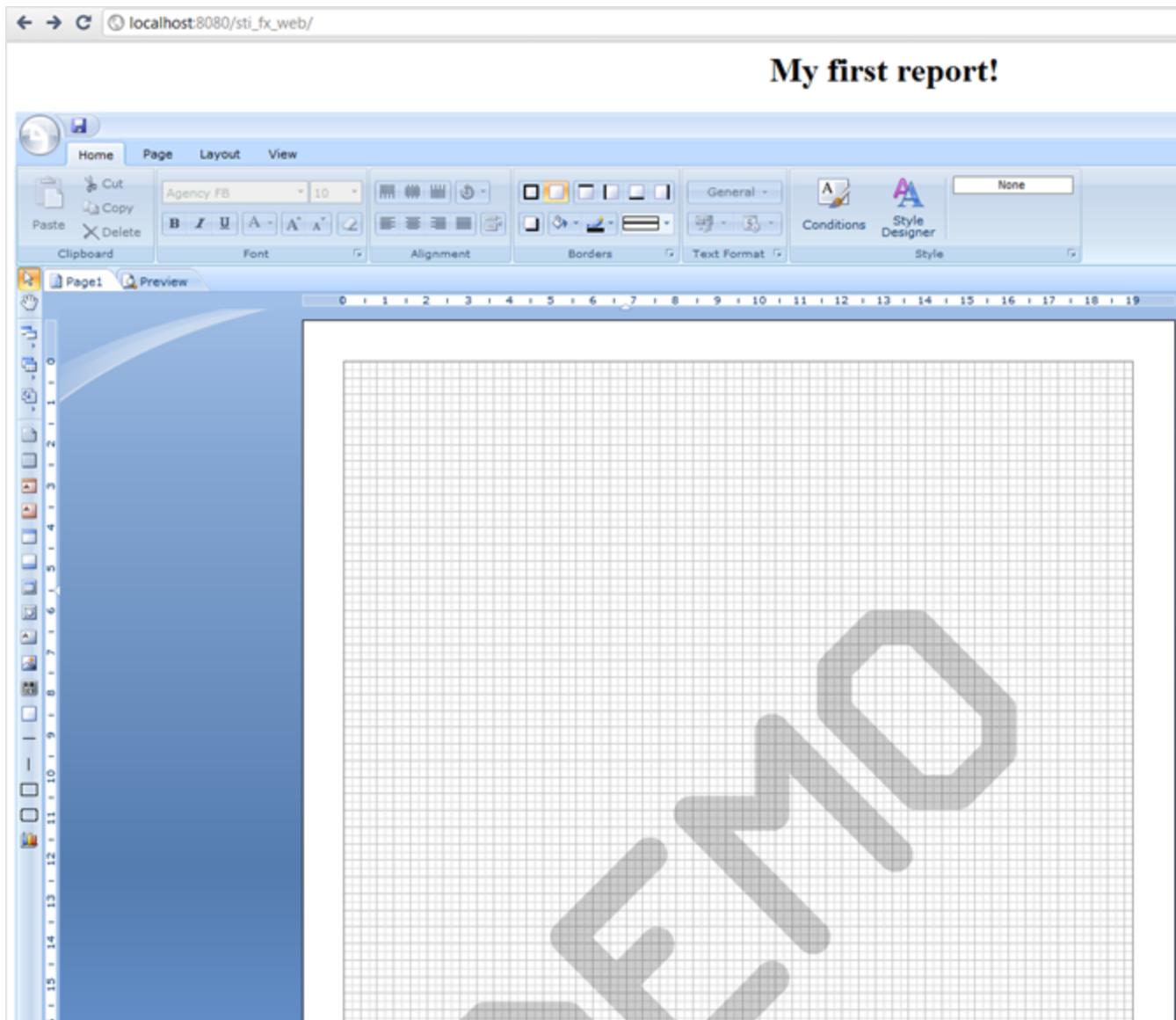
Add taglib directives in the JSP (Listing 5). They will work with custom tags on the page.

```

<%@ taglib uri="http://stimulsoft.com/designer" prefix="stidesignerfx" %>
<%@ taglib uri="http://stimulsoft.com/viewer" prefix="stiviewerfx" %>

```

Add a tag `<stidesignerfx:iframe/>`, an analog of an html tag `iframe` with the support of all its attributes. See the following as a result of the application deployment, (see the picture below):



Description of custom tags:

There is a division into two components: DesignerFx and ViewerFx, it can be seen from Listing 5. Consider a DesignerFx component. For a ViewerFx it works the same way. Tags:

```
<stidesignerfx:link text="a link for jumping to the Designer"/>
<stidesignerfx:button value="a button for jumping to the Designer"/>
<stidesignerfx:frame title="analog of the html tag frame which contains a Designer"/>
<stidesignerfx:iframe title="analog of the html tag iframe which contains a Designer"/>
```

All these are analogs of similar HTML tags, supporting all the attributes. A list of standard attributes is expanded for displaying the report and setting variables for the report. The report = "SimpleList.mrt" attribute opens the report with the name SimpleList.mrt. Variables in the Report can be passed in two

ways:

- ▶ Set the value of a variableStr attribute as a string in the following format: "Variable1 = value1 & Variable2 = value2". In this case, two variables Variable1 with a value1 and Variable2 with a value2 in the report will be passed. For example, you need to edit the index.jsp file to open a report with the name MyFirstReport.mrt by clicking the button and the MyVar report variable has the stidesignerfx value (Listing 5).

```
<!DOCTYPEhtmlPUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page contentType="text/html;charset=UTF-8" import="java.util.*" %>
<%@ taglib uri="http://stimulsoft.com/designer" prefix="stidesignerfx" %>
<%@ taglib uri="http://stimulsoft.com/viewer" prefix="stiviewerfx" %>

<html>
<head>
    <title>Report</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
    <stidesignerfx:button value="Run the report designer" report="MyFirstReport.mrt"
        variableStr="MyVar=stidesignerfx" />
</body>
</html>
```

- ▶ It is also possible to pass parameters to a report as a Map <String, String>. Redesign our webpage as follows (Listing 6). In this case, a report with the name MyFirstReport.mrt will be loaded in a report and two parameters will be passed into it:

```
<!DOCTYPEhtmlPUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page contentType="text/html;charset=UTF-8" import="java.util.*" %>
<%@ taglib uri="http://stimulsoft.com/designer" prefix="stidesignerfx" %>
<%@ taglib uri="http://stimulsoft.com/viewer" prefix="stiviewerfx" %>

<html>
<head>
    <title>Report</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<%
    Map<String, String>variableMap= new HashMap<String, String>();
    variableMap.put("Variable1", "var1");
    variableMap.put("Variable2", "var2");
    request.setAttribute("myMap", variableMap);
%>
    <stidesignerfx:iframe report="MyFirstReport.mrt" variableMap="myMap"
        width="100%" height="100%" align="right"
        styleClass="" frameborder="0" styleId=""
```

```

marginheight="1" marginwidth="1" name="stidesignerfx"
scrolling="no" style="" title="report" />

</body>
</html>

```

Data here are passed as a `HashMap`, this parameter should be set to the request or session, and the key under which it will be there should be passed to the tag as a `variableMap` attribute. Applying two attributes `variableMap` and `variableStr` is not allowed.

19.6 Loading, Saving and Loading Custom Data

Before running, the application should be configured. For configuration the `my.servlet.ApplicationInitializer` class that is specified in the `web.xml` is used. The code init reports:

```

package my.servlet;

import java.io.IOException;
import java.util.Properties;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import my.actions.MyLoadAction;
import my.actions.MyLoadDataAction;
import my.actions.MyLocalizationAction;
import my.actions.MyMailAction;
import my.actions.MyRenderReportAction;
import my.actions.MySaveAction;

import com.stimulsoft.base.exception.StiException;
import com.stimulsoft.flex.StiFlexConfig;

/**
 * Application initialization.
 */
public class ApplicationInitializer implements ServletContextListener {

    @Override
    public void contextInitialized(final ServletContextEvent event) {
        try {
            // configuration application
            StiFlexConfig stiConfig = initConfigWithoutDir();
            // -----
            // need to override the standard methods
            // another comment
            stiConfig.setLoadClass(MyLoadAction.class);
            stiConfig.setSaveClass(MySaveAction.class);
        }
    }
}

```

```

        stiConfig.setLoadDataClass(MyLoadDataAction.class);
        stiConfig.setMailAction(MyMailAction.class);
        stiConfig.setLocalizationAction(MyLocalizationAction.class);
        stiConfig.setRenderReportAction(MyRenderReportAction.class);
        // -----
        StiFlexConfig.init(stiConfig);

        // set variable in servlet context attribute
        // Map<String, String> myVariableMap = new HashMap<String,
        String>();
        // myVariableMap.put("Variable1", "myVariableMap");
        // event.getServletContext().setAttribute("myMap",
        myVariableMap);
    } catch (Exception e) {
        throw new RuntimeException(e);
    }
}

@Override
public void contextDestroyed(final ServletContextEvent event) {
    // empty
}

public StiFlexConfig initConfigWithoutDir() throws StiException,
IOException {
    Properties properties = new Properties();
    // load your own Properties;
    // InputStream inStream =
    getClass().getResourceAsStream("RESOURCE_PATH");
    // properties.load(inStream);
    return new StiFlexConfig(properties);
}
}

```

In which the main application directory with the file `stimulsoft.properties` will be defined. In order to make your own reports saving or loading, it is necessary to specify these classes in a configuration, just the same way as you can specify a class to load data from xml. Classes are as follow: Listing MyLoadAction.java

```
package my.actions;

import java.io.InputStream;

import com.stimulsoft.flex.StiLoadAction;
import com.stimulsoft.flex.utils.StiSaveLoadFileReport;

public class MyLoadAction extends StiLoadAction {
```

```

public InputStream load(String repotrName) {
    System.out.println("must override this method to specify your own
load repotr");
    return new StiSaveLoadFileReport().getReport(repotrName);
}
}

```

Listing MySaveAction.java:

```

package my.actions;

import com.stimulsoft.flex.StiSaveAction;
import com.stimulsoft.flex.utils.StiOperationResult;
import com.stimulsoft.flex.utils.StiSaveLoadFileReport;

public class MySaveAction extends StiSaveAction {

    @Override
    public StiOperationResult save(String report, String reportName,
        boolean newReportFlag) {
        System.out.println("must override this method to specify your own
save report");
        return new StiSaveLoadFileReport().save(report, reportName,
            newReportFlag);
    }

}

```

Listing MyLoadDataAction.java:

```

package my.actions;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.stimulsoft.flex.StiLoadDataAction;

public class MyLoadDataAction extends StiLoadDataAction {

    @Override
    protected String getConnectionString() {
        System.out.println("must override this method to specify your own
connection string");
        // return
        // "Data Source=localhost\\SQLEXPRESS;Initial Catalog=Mybase;User
        // ID=UserName; Password=Password;";
        return super.getConnectionString();
    }
}

```

```

    }

    @Override
    protected String getUserName() {
        System.out.println("must override this method to specify your own
        user name");
        // return "UserName";
        return super.getUserName();
    }

    @Override
    protected String getPassword() {
        System.out.println("must override this method to specify your own
        password");
        // return "Password";
        return super.getPassword();
    }

    @Override
    protected String getQuery() {
        System.out.println("my Query " + super.getQuery());
        return super.getQuery();
    }

    @Override
    public Connection getConnection() throws ClassNotFoundException,
    SQLException {
        System.out.println("must override this method to specify your own
        connection");
        boolean overrideByConnectionString = getConnectionString() != null
            && getConnectionString().equals("needOverride");
        boolean overrideByDataSource = getDataSourceName() != null
            && getDataSourceName().equals("DataSourceOverride");
        if (overrideByConnectionString || overrideByDataSource) {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            Properties info = new Properties();
            info.setProperty("user", "test");
            info.setProperty("password", "test");
            String connectionString = "jdbc:sqlserver://localhost\
                \\SQLEXPRESS1:1433;databaseName=mybase;";
            return DriverManager.getConnection(connectionString, info);
        } else {
            return super.getConnection();
        }
    }
}

```

Listing MyLocalizationAction.java:

```
package my.actions;
```

```
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.stimulsoft.base.exception.StiException;
import com.stimulsoft.base.utils.StiXmlMarshalUtil;
import com.stimulsoft.flex.StiLocalizationAction;
import com.stimulsoft.flex.StiLocalizationInfo;
import com.stimulsoft.lib.io.StiFileUtil;

public class MyLocalizationAction extends StiLocalizationAction {

    @Override
    public List<StiLocalizationInfo> getLocalizations() throws
    StiException, FileNotFoundException {
        System.out.println("must override this method to specify your own
Localizations");
        List<StiLocalizationInfo> list = new
        ArrayList<StiLocalizationInfo>();
        File localizationDir = getLocalizationDir();
        if (localizationDir.exists()) {
            Iterator<File> iterateLocalization =
            StiFileUtil.iterateFiles(localizationDir,
            new String[] { "xml" }, false);
            for (; iterateLocalization.hasNext();) {
                File fileLoc = iterateLocalization.next();
                InputStream is = new BufferedInputStream(new
                FileInputStream(fileLoc));
                StiLocalizationInfo localization =
                StiXmlMarshalUtil.unmarshal(is,
                StiLocalizationInfo.class);
                localization.setKey(fileLoc.getName());
                list.add(localization);
            }
        }
        return list;
    }

    @Override
    protected File getLocalizationDir() {
        System.out.println("must override this method to specify your own
LocalizationDir");
        return new File("Localization");
    }
}
```

```

@Override
public InputStream getLocalization(String key) throws StiException,
FileNotFoundException {
    System.out.println("must override this method to specify your own
load Localization");
    File file = new File(getLocalizationDir(), key);
    return new BufferedInputStream(new FileInputStream(file));
}
}

```

Listing MyMailAction.java:

```

package my.actions;

import java.util.Properties;

import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import javax.mail.internet.PreencodedMimeBodyPart;

import com.stimulsoft.base.mail.StiMailProperties;
import com.stimulsoft.flex.StiMailAction;
import com.stimulsoft.flex.interactionObject.StiMailData;

/**
 * MyMailAction.
 *
 * @Copyright Stimulsoft
 */
public class MyMailAction extends StiMailAction {

    @Override
    public void init(StiMailData mailData, StiMailProperties mailConf) {
        System.out.println("must override this method to specify your own
init");
        this.mailData = mailData;
        this.mailConf = mailConf;
        session = getSession();
    }

    @Override

```

```
protected Session getSession() {
    System.out.println("must override this method to specify your own
Session");
    Properties props = getProperties();
    return Session.getInstance(props);
}

@Override
protected Properties getProperties() {
    System.out.println("must override this method to specify your own
mail Properties");
    Properties props = new Properties();
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.starttls.enable", "true");
    return props;
}

@Override
protected Message getMessage() throws MessagingException {
    System.out.println("must override this method to specify your own
mail Message");
    Message message = new MimeMessage(session);
    message.setRecipients(Message.RecipientType.TO,
    InternetAddress.parse(mailConf.getFrom()));
    message.setRecipients(Message.RecipientType.CC,
    InternetAddress.parse(mailConf.getRecipients()));

    message.setSubject(mailConf.getSubject());

    BodyPart text = getTextPart();
    BodyPart body = getFilePart();

    Multipart mp = new MimeMultipart();
    mp.addBodyPart(text);
    mp.addBodyPart(body);

    message.setContent(mp);
    return message;
}

@Override
protected BodyPart getTextPart() throws MessagingException {
    System.out.println("must override this method to specify your own
mail TextPart");
    MimeBodyPart text = new MimeBodyPart();
    text.setText(mailConf.getBody(), "UTF-8", "plain");
    return text;
}

@Override
protected BodyPart getFilePart() throws MessagingException {
```

```

        System.out.println("must override this method to specify your own
        mail FilePart");
        PreencodedMimeBodyPart body = new
        PreencodedMimeBodyPart("base64");
        body.setFileName(mailData.getFileName());
        body.setContent(mailData.getData(), mailData.getMIMEType());
        return body;
    }

    private Transport getTransport() throws MessagingException {
        System.out.println("must override this method to specify your own
        mail Transport");
        Transport transport = session.getTransport("smtp");
        transport.connect(mailConf.getHost(), mailConf.getSmtpPort(),
        mailConf.getUserName(),
        mailConf.getPassword());
        return transport;
    }

    @Override
    public void sendMessage() throws MessagingException {
        System.out.println("must override this method to specify your own
        send Message");
        Message message = getMessage();
        Transport transport = getTransport();
        transport.sendMessage(message, message.getAllRecipients());
        transport.close();
    }
}

```

Listing MyRenderReportAction.java:

```

package my.actions;

import java.io.IOException;

import com.stimulsoft.base.exception.StiException;
import com.stimulsoft.flex.StiRenderReportAction;
import com.stimulsoft.report.StiReport;

public class MyRenderReportAction extends StiRenderReportAction {

    @Override
    public StiReport render(StiReport report) throws IOException,
    StiException {
        System.out.println("must override this method to specify your own
        render report");
        return report.render();
    }
}

```

Template JDBC connections

```
jdbc.driver={myDriver};
jdbc.url={myConnectionString};
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>

20 Java HTML5 Designer

The Java designer will be described in this section.

20.1 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:

Listing 1. Contents of 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>
        <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).

Listing 2. Contents of 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.StiXmlTableFieldsRequest"%>
<%@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>
<stiwebdesigner:resources />
<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
        StiXmlTableFieldsRequest tables =
StiDataColumnsUtil.parseXSDSchema(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,
String reportName, HttpServletRequest request){
try{
    FileOutputStream fos = new
FileOutputStream(savePath + reportName + ".mrt");
    StiSerializeManager.serializeReport(report, fos);
    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

```
<%@ 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

```
<stiwebdesigner:webdesigner handler="${handler}" report="${report}" />
```

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\Datasources (if required).

» public void onOpenReportTemplate(StiReport report, HttpServletRequest request);

Occurred on opening StiReport. Method intended for populate report with Database\Datasources (if required).

» public void onNewReportTemplate(StiReport report, HttpServletRequest request);

Occurred on new StiReport. Method intended for populate report with Database\Datasources (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.

20.2 Template JDBC Connections

```
jdbc.driver={myDriver};
jdbc.url={myConnectionString};
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>

21 Java HTML5 Viewer

21.1 Installation

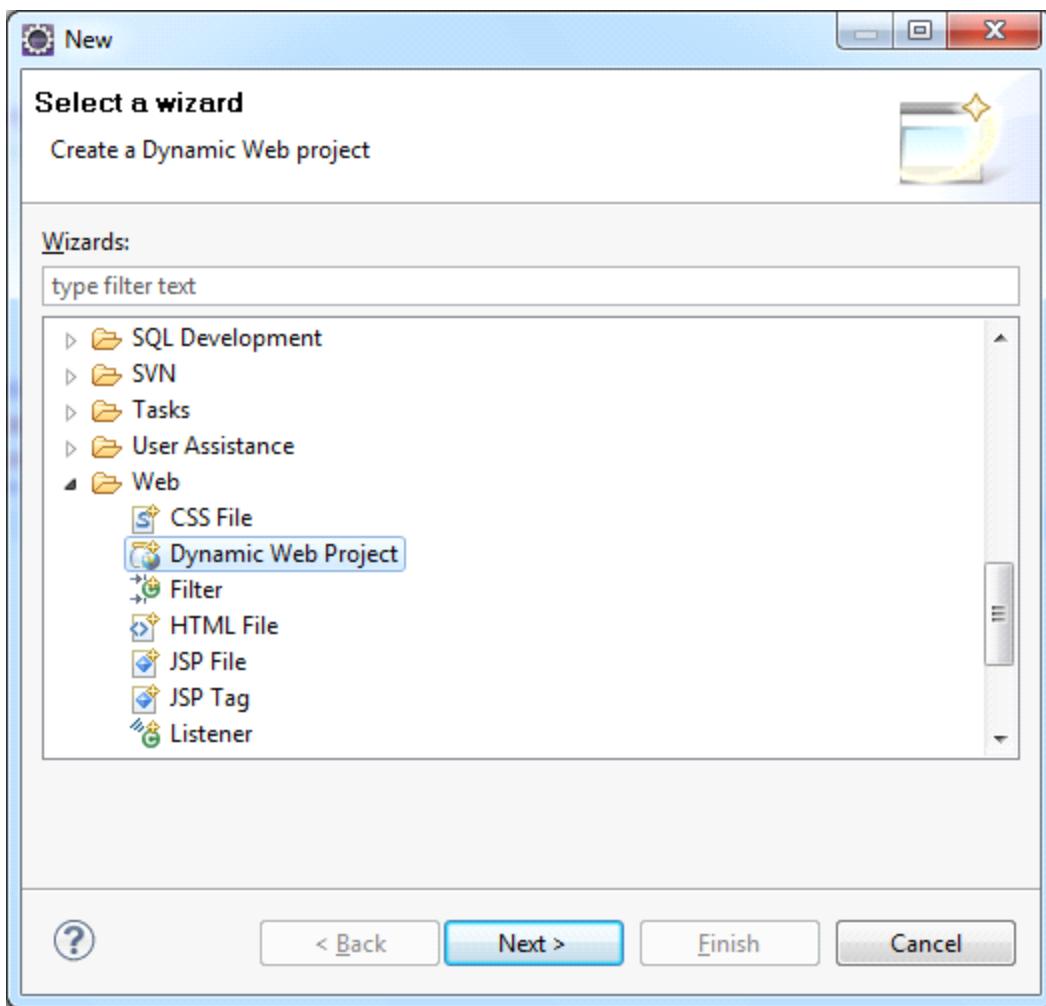
Installation:

➤ Download and install **Java™ SE** version 1.5 or higher (for the version 1.5 jaxb-impl and jaxb-api libraries are required).

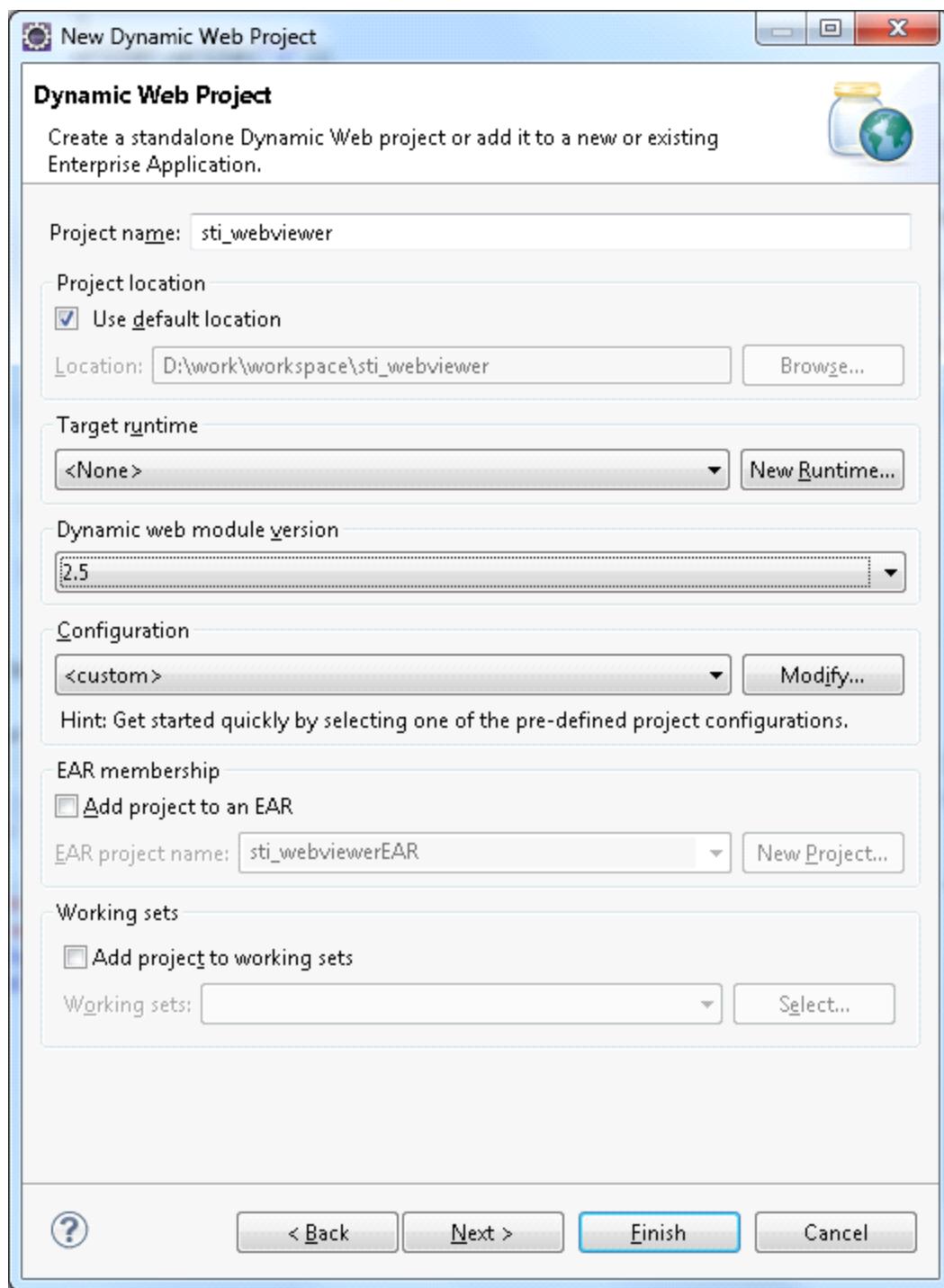
- » Download and install **EclipsePlatform**.
- » Download an archive with **jar** files on [stimulsoft](#).

21.2 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 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:

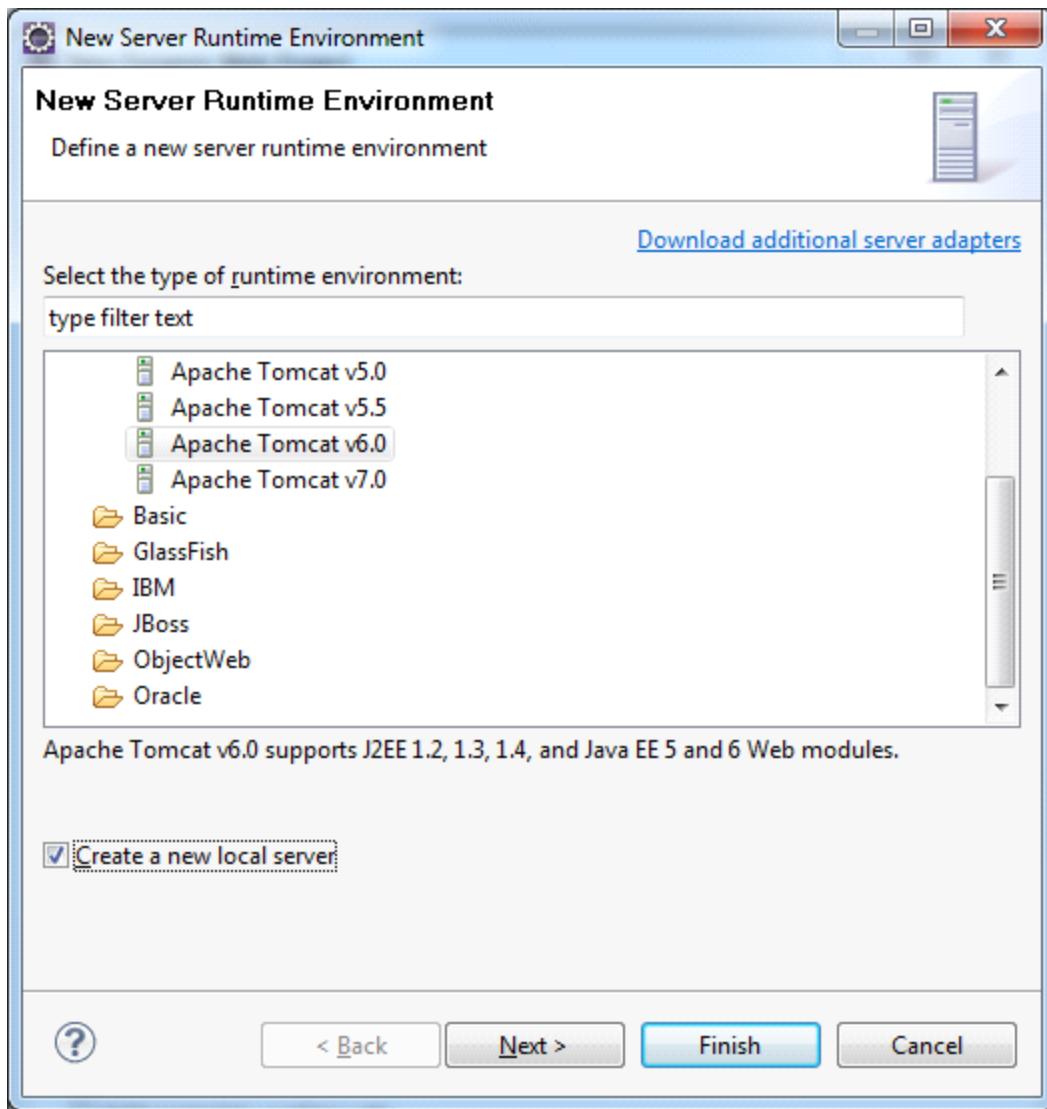


► Target a runtime

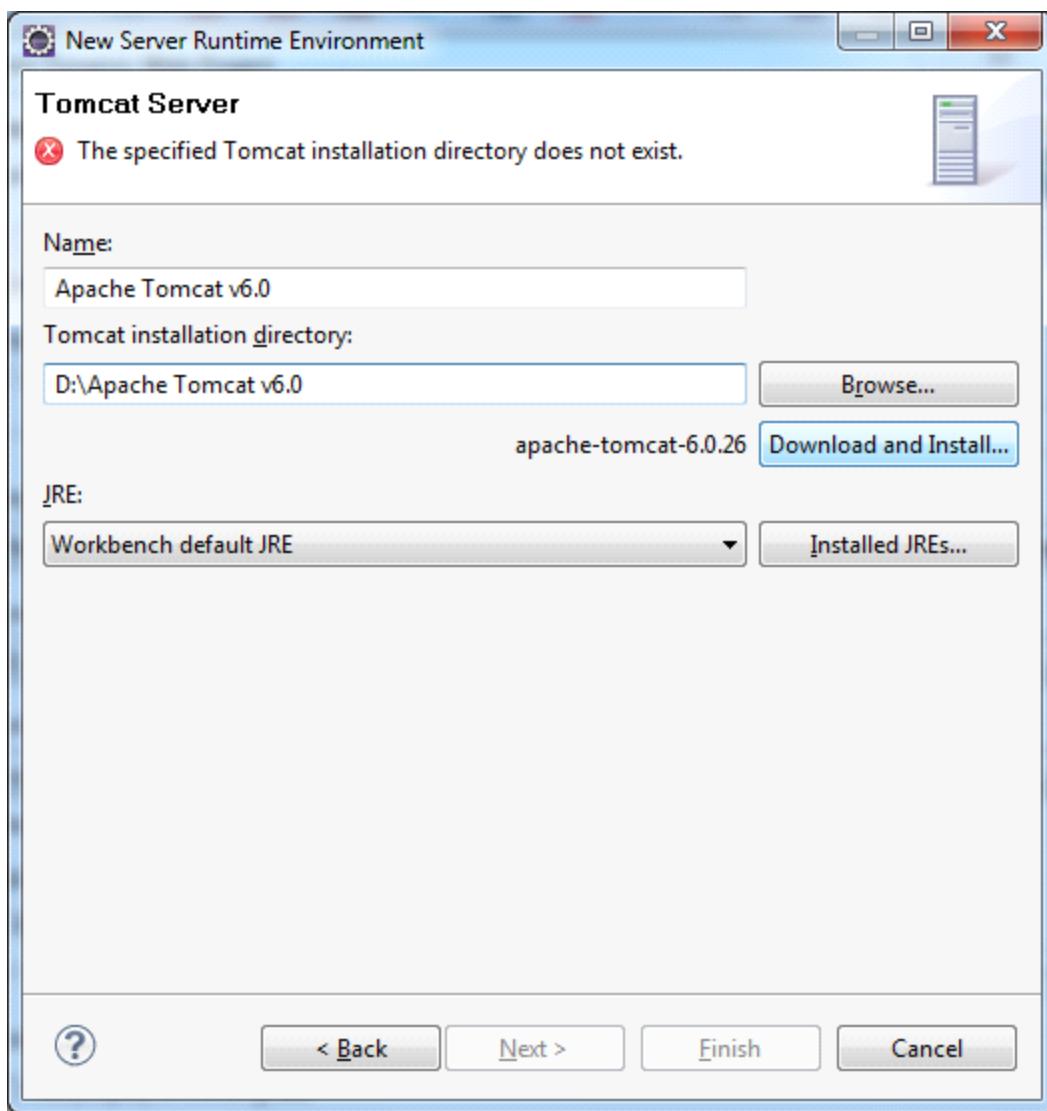
Under Target Runtime, you see <None>, as shown in Figure 1, because you haven't created a runtime yet for Apache Tomcat. Click New Runtime to open the New Target Runtime Wizard. Select Apache Tomcat of the required version from the list of available, check the Create a new local server as shown in Figure 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 list. Check Create a new local server as shown on Figure 3, then click Next.Create a new server runtime:

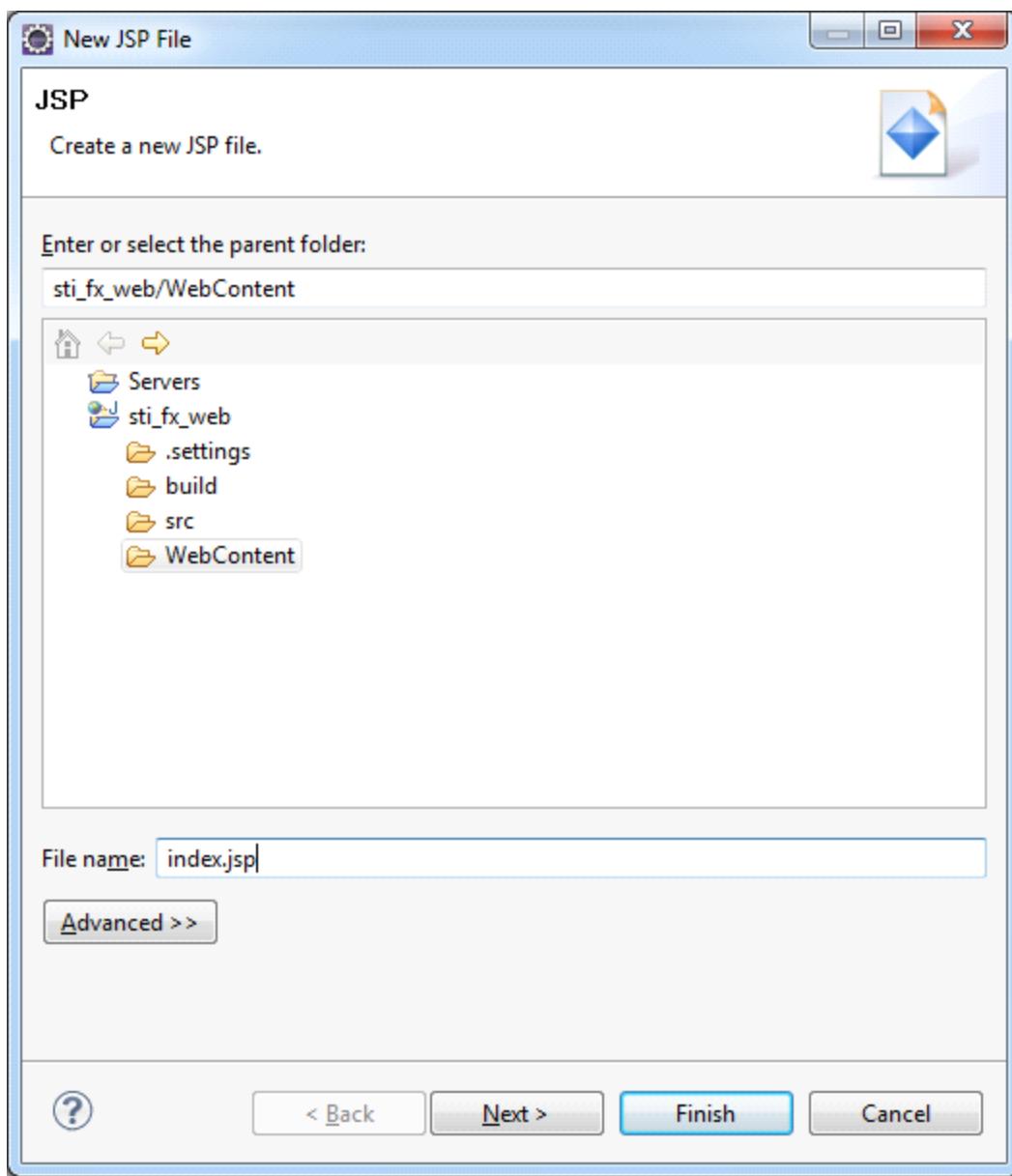


Then define the Tomcat installation directory, in which Apache Tomcat is installed, or in which one needs to install it, as shown on Figure 4. If it is not installed, then click Download and Install. After all fields are specified, click Finish. Define the server location:



21.3 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:



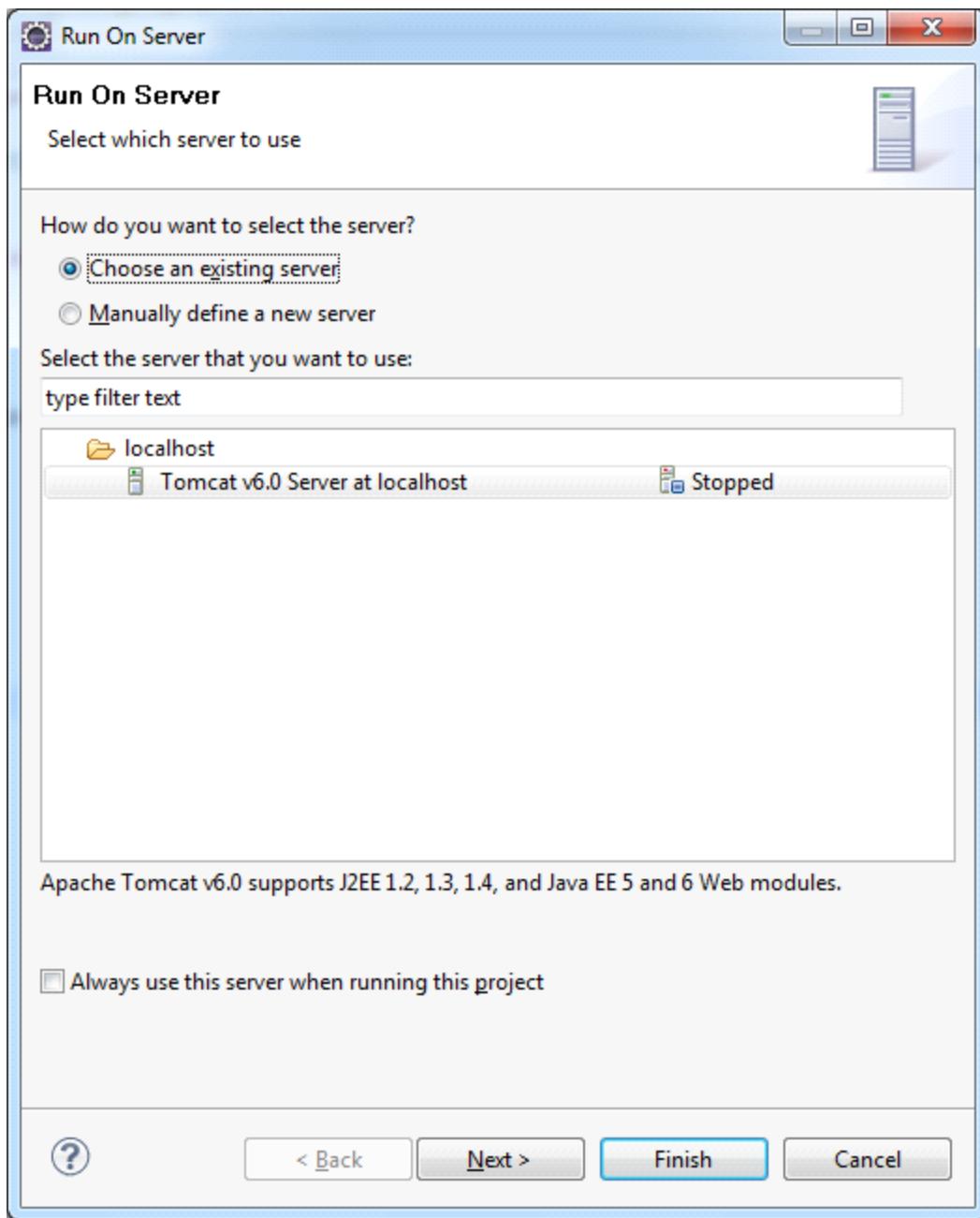
Now open the **index.jsp** and edit it so that it displays the current date. The code page is specified in the Listing 1.

Listing 1. Contents of index.jsp

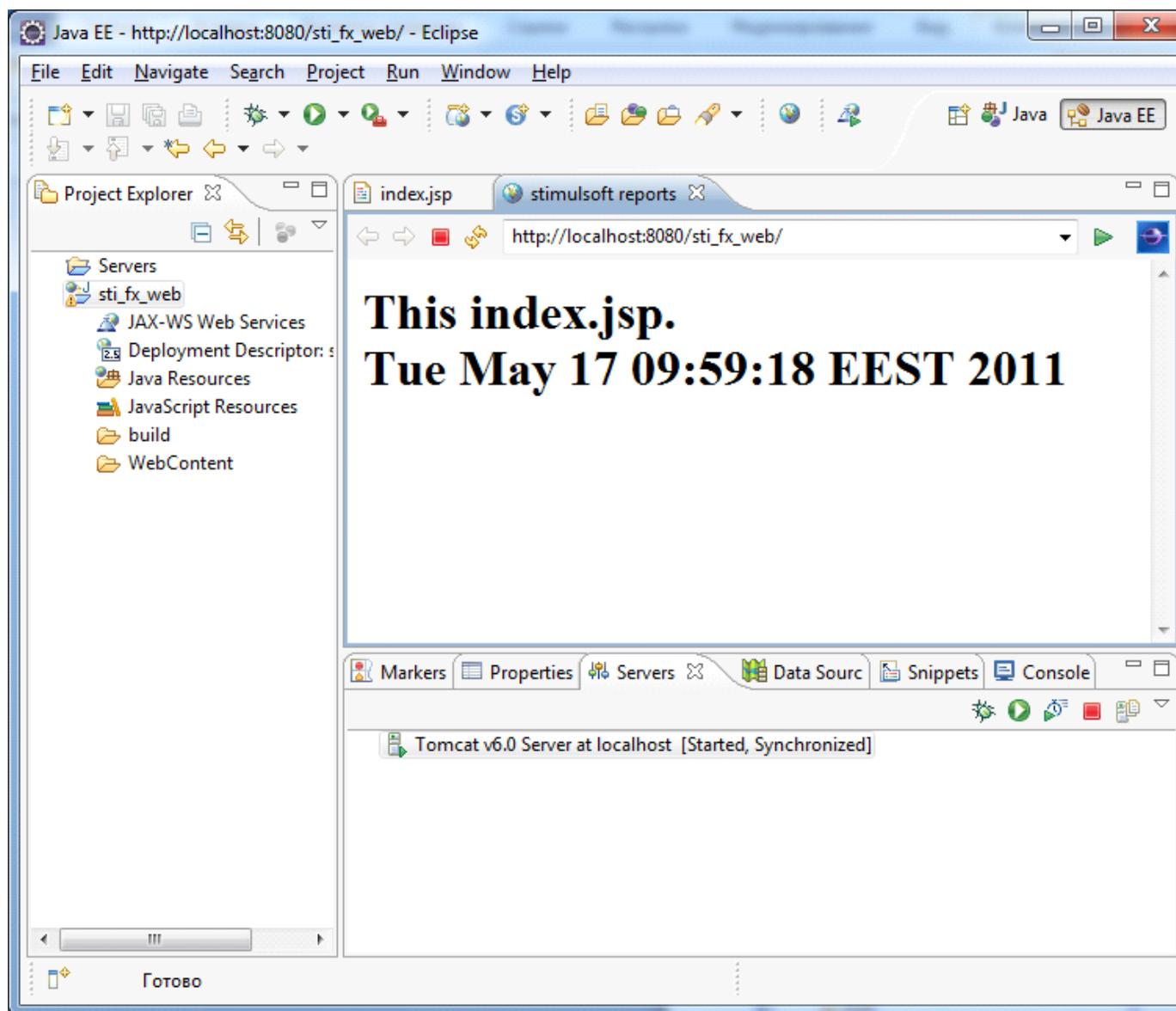
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>stimulsoft reports</title>
</head>
```

```
<body>
    <%java.util.Date date = new java.util.Date();%>
    <h1>
        This index.jsp.<br>
        <%=date.toString()%>
    </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**:

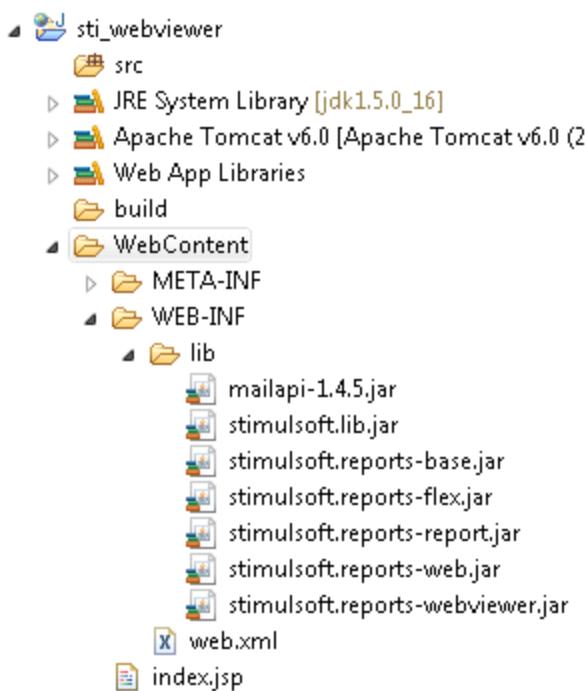


As a result, you receive the following (see Figure 7). This page will be available from any browser at <http://localhost:8080/{ProjectName}> (where the **{ProjectName}** name of the created project, in our case **sti_webviewer**):



21.4 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):



Next, open the web.xml for editing, it should look like in Listing 2:

Listing 2. Contents of 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.StiWebViewActionServlet</servlet-class>

```

```

</servlet>
<servlet-mapping>
    <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).

Listing 3. Contents of 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.

Listing 4. Custom Stimulsoft tag

```
<%@ taglib uri="http://stimulsoft.com/webviewer" prefix="stiwebviewer"%>
```

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.

21.5 Description of Webviewer Tag

```
<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:\reports\TwoSimpleLists.mdc with custom parameters)

```
<!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
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>
```

My first report!

Print Save Page 1 of 5 100% Whole Report

Two Simple Lists

The sample demonstrates how to create two simple list reports.

Stimulsoft Date: 2011

Companies

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) 555-7847	Sales Associate
16 Consolidated Holdings	Berkeley Gardens 12 Brewery	(171) 555-2282	Sales Representative
17 Drachenblut Delikatessen	Walserweg 21	0241-039123	Order Administrator
18 Du monde entier	67, rue des Cinquante Otages	40.67.88.88	Owner
19 Eastern Connection	35 King George	(171) 555-0297	Sales Agent
20 Ernst Handel	Kirchgasse 6	7675-3425	Sales Manager
21 Familia Arquibaldo	Rua Orós, 92	(11) 555-9857	Marketing Assistant
22 FISSA Fabrica Inter. Salchichas S.A.	C/ Moralzarzal, 86	(91) 555 94 44	Accounting Manager
23 Folies gourmandes	104, chaussée de Tournai	20.16.10.16	Assistant Sales Agent
24 Folk och få HB	Åkergratan 24	0895-34 67 21	Owner
25 Frankenversand	Berliner Platz 43	089-0877310	Marketing Manager
26 France restauration	54, rue Royale	40.32.21.21	Marketing Manager
27 Franchi S.p.A.	Via Monte Bianco 34	011-4988260	Sales Representative

Co	Co
39 Königlich Essen	
40 La corne d'abondance	
41 La maison d'Asie	
42 Laughing Bacchus	
43 Lazy K Country Store	
44 Lehmanns Markt	
45 Let's Stop N Shop	
46 LILA-Supermerca	
47 LINO-Delicatessen	
48 Lonesome Pine	
49 Magazzini Alimentari	
50 Maison Dewey	
51 Mère Paillarde	
52 Morgenstern Ges	
53 North/South	
54 Océano Atlántico	
55 Old World Delica	
56 Ottlies Käselade	
57 Paris spécialités	
58 Pericles Comida	
59 Piccolo und mehr	
60 Princesa Isabel V	
61 Que Delicia	
62 Queen Cozinha	
63 QUICK-Stop	
64 Rancho grande	
65 Rattlesnake Cam	
66 Reggiani Caseifici	
67 Ricardo Adocicado	
68 Richter Supermark	

21.6 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.
- » 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 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**;
- » **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 which indicates timeout in minutes for execute dummy request to avoid end session, default is 0 (disabled);

21.7 Template JDBC Connections

```
jdbc.driver={myDriver};  
jdbc.url={myConnectionString};  
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://[:port]/[database]  
jdbc.username={myUserName };  
jdbc.password={ myUserPassword };
```

<http://jdbc.postgresql.org/documentation/80/connect.html>

22 WinRT Viewer

ViewerRT is used to display reports in the WinRT components. The component can display a report, zoom it, save rendered reports to various exporting formats and print them.

22.1 How to Show Report?

Add the following code to display the rendered report (*.mdc, *.mdz, *.mdx):

XAML:

```
<Page x:Class="Demo.RT.BlankPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:viewerRT="using:Stimulsoft.Report.Viewer.RT">

    <viewerRT:StiViewerControl x:Name="viewerControl"/>
</Page>
```

C#:

```
namespace Demo.RT
{
    public sealed partial class BlankPage : Page
    {
        #region Handlers
        async private void BlankPage_Loaded(object sender, RoutedEventArgs e)
        {
            StorageFolder folder =
                Windows.Storage.KnownFolders.DocumentsLibrary;
            StorageFile storageFile = await
                folder.GetFileAsync("SimpleList.mdc");

            StiReport report = new StiReport();
            await report.LoadDocumentAsync(storageFile);
            viewerControl.Report = report;
        }
        #endregion

        public BlankPage()
        {
            this.InitializeComponent();
            this.Loaded += BlankPage_Loaded;
        }
    }
}
```

If the report has not been rendered, i.e. the report template is saved (*.mrt, *.mrz, *.mrx), then enter the following code:

XAML:

```
<Page x:Class="Demo.RT.BlankPage"
```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:viewerRT="using:Stimulsoft.Report.Viewer.RT">

<viewerRT:StiViewerControl x:Name="viewerControl"/>
</Page>

```

C#:

```

namespace Demo.RT
{
    public sealed partial class BlankPage : Page
    {
        #region Handlers
        async private void BlankPage_Loaded(object sender, RoutedEventArgs e)
        {
            StorageFolder folder =
                Windows.Storage.KnownFolders.DocumentsLibrary;
            StorageFile storageFile = await
                folder.GetFileAsync("SimpleList.mrt");

            StiReport report = new StiReport();
            await report.LoadAsync(storageFile);
            await report.RenderAsync();

            viewerControl.Report = report;
        }
        #endregion

        public BlankPage()
        {
            this.InitializeComponent();
            this.Loaded += BlankPage_Loaded;
        }
    }
}

```

You can open a rendered report by clicking the **Open** button.

22.2 Saving Report From Code

The report can be saved from the project code. Here is an example of the code to save a report:

XAML:

```
<Page x:Class="Demo.RT.BlankPage">
```

```

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:viewerRT="using:Stimulsoft.Report.Viewer.RT">

<viewerRT:StiViewerControl x:Name="viewerControl"/>
</Page>

```

C#:

```

namespace Demo.RT
{
    public sealed partial class MainPage : Page
    {
        #region Handlers
        async private void buttonSaveReport_Click(object sender,
        RoutedEventArgs e)
        {
            StiReport report = new StiReport();
            StorageFolder folder =
                Windows.Storage.KnownFolders.DocumentsLibrary;
            StorageFile storageFile = await
                folder.CreateFileAsync("Report1.mdc");

            await report.SaveDocumentAsync(storageFile);
        }
        #endregion

        public MainPage()
        {
            this.InitializeComponent();
            this.Loaded += MainPage_Loaded;
        }
    }
}

```

23 WinRT Designer

The report designer for **WinRT** is designed for creating and editing reports in **Windows 8** and works with **ARM** and Intel versions. The UI of the designer provides for user convenient control, wide range of tools, components and instruments for creating reports, visual design and previewing. The operations in the designer can be done using touch (hand, stylus) or mouse. The report designer supports Ribbon interface 2013.

23.1 Working with Report Code

In this topic we will review the examples of working from code for the report designer **WinRT**.

▷ Loading a report:

In order to load the report designer for further editing it is necessary to assign a report to the Report property of the designer. If you do not assign anything to this property, the designer will be loaded

```
var file = await
Windows.Storage.KnownFolders.DocumentsLibrary.GetFileAsync("report.mrt")
;
var report = new StiReport();
await report.LoadAsync(file);
```

Loading a created report (**MDC** file):

```
var file = await
Windows.Storage.KnownFolders.DocumentsLibrary.GetFileAsync("report.mdc")
;
var report = new StiReport();
await report.LoadDocumentAsync(file);
```

▷ Saving a report:

After creating or editing a report template you should save the changes. Below is a method of saving a report code in the *.mrt file:

```
var file = await
Windows.Storage.KnownFolders.DocumentsLibrary.CreateFileAsync("report.mr
t", CreationCollisionOption.ReplaceExisting);
var report = new StiReport();
await report.SaveAsync(file);
```

Saving a created report (**MDC** file):

```
var file = await
Windows.Storage.KnownFolders.DocumentsLibrary.CreateFileAsync("report.md
c", CreationCollisionOption.ReplaceExisting);
var report = new StiReport();
await report.SaveDocumentAsync(file);
```

24 Exports

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.

24.1 Available File Formats

A list of supported file formats is represented in the table below. All exports are joined into groups.

Export Name
PDF (Portable Document Format)
XPS (XML Paper Specification)
HTML (HyperText Markup Language)
HTML5 (HyperText Markup Language)
MHTML (MIME HTML)
TXT (Text File)
RTF (Rich Text)
Microsoft Word 2007/2010
ODT (Open Document Text)
Microsoft Excel
Microsoft Excel Xml
Microsoft Excel 2007/2010
Microsoft Power Point 2007/2010
ODS (Open Document Spreadsheet)
CSV (Comma Separated Values)
DBF (DataBase File)
XML (eXtensible Markup Language)
DIF (Data Interchange Format)
SYLK (Symbolic Link)
BMP (Bitmap)
GIF (Graphics Interchange Format)
PNG (Portable Network Graphics)
TIFF (Tagged Image File Format)
JPEG (Joint Photographic Experts Group)
PCX (PCExchange)

WMF (Windows MetaFile)
SVG (Scalable Vector Graphics)

24.2 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.

24.2.1 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:

```
StiPdfExportSettings pdfSettings = new StiPdfExportSettings();
report.ExportDocument(StiExportFormat.Pdf, "MyReport.Pdf",
pdfSettings);
```

The following code is 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:

```
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:

```
MemoryStream stream = new MemoryStream();
report.ExportDocument(StiExportFormat.Pdf, stream);
```

! Notice: 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:

```
Stimulsoft.Report
```

24.2.2 Export 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.
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 HtmlTable 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.
Word2007	export to Microsoft Word 2007. This format is supported starting

	with Microsoft Office 2007.
Odt	export to the OpenDocument Writer file.
Spreadsheets:	
Excel	export to Microsoft Excel. The file is created using the BIFF (Binary Interchange File Format).
ExcelXml	export to Microsoft Excel Xml. The file is created using the Xml. This format is supported starting with Microsoft Office 2003.
Excel2007	export to Microsoft Excel 2007. This format is supported starting with Microsoft Office 2007.
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.

24.2.3 Export Service

The way to create the export service is shown below. See the code:

```
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**:

```
service.Export(report, "MyReport.pdf");
```

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.

⚠️ Notice: 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:

```
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 letter, the exported report will be attached to the Email letter. The code above requires connection of the following names from the **Stimulsoft.Report.dll** assemblies:

```
Stimulsoft.Report
Stimulsoft.Report.Export
```

24.2.4 All Export Services

The **StiExportFormat** enumeration describes export formats. Brief information of exports is represented below.

Export services to Adobe PDF and Microsoft XPS:

- ▶ StiPdfExportService
- ▶ StiXpsExportService

Export services to HTML and MHT:

- ▶ StiHtmlExportService
- ▶ StiMhtExportService

Export services to Microsoft Excel and Open Document Calc:

- ▶ StiExcelXmlExportService
- ▶ StiExcelExportService
- ▶ StiExcel2007ExportService
- ▶ StiOdsExportService

Export services to text formats:

- ▶ StiTxtExportService
- ▶ StiRtfExportService
- ▶ StiWord2007ExportService

► StiOdtExportService

Export services to data:

► StiCsvExportService
► StiDbfExportService
► StiXmlExportService
► StiDifExportService
► StiSylkExportService

Export services to graphic formats:

► StiBmpExportService
► StiGifExportService
► StiJpegExportService
► StiPcxExportService
► StiPngExportService
► StiTiffExportService
► StiEmfExportService

24.3 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. These are formats to **PDF** (Portable Document Format), **Microsoft Power Point 2007/2010** and **XPS** (XML Paper Specification).

24.3.1 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 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.

24.3.1.1 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:

```
StiOptions.Export.Pdf.ReduceFontFileSize = false;
```

24.3.1.2 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

24.3.1.2.1 Digital Signature from Code

The **StiPdfExportSettings** class is used to control digital signature. It has the following properties:

```
public bool UseDigitalSignature
public bool UseLocalMachineCertificates
public bool GetCertificateFromCryptoUI
public string SubjectNameString
```

By default:

```
UseDigitalSignature = false;
UseLocalMachineCertificates = true;
GetCertificateFromCryptoUI = true;
SubjectNameString = string.Empty;
```

A sample how to use these properties is shown below:

```
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);
```

24.3.1.3 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.

24.3.1.3.1 Using Parameters of Encryption from Code

Using the **StiPdfExportSettings** class it is possible to set the encryption parameters from code. The following properties of this class are used:

```
public string PasswordInputUser
public string PasswordInputOwner
public StiUserAccessPrivileges UserAccessPrivileges
public StiPdfEncryptionKeyLength KeyLength
```

The **StiUserAccessPrivileges** enumeration contains the following elements (flags):

- » None,
- » PrintDocument,
- » ModifyContents,
- » CopyTextAndGraphics,
- » AddOrModifyTextAnnotations,
- » All

The **StiPdfEncryptionKeyLength** enumeration contains the following elements:

- » Bit40,
- » Bit128

By default the values set as follow:

```
PasswordInputUser = string.Empty;
PasswordInputOwner = string.Empty;
UserAccessPrivileges = StiUserAccessPrivileges.All;
KeyLength = StiPdfEncryptionKeyLength.Bit40;
```

An example of using:

```
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);
```

24.3.1.4 Editable Fields

To enable the export of editable fields it is necessary to set the static property

```
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**.

24.3.1.5 Export Settings

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 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
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
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 **ReduceFontSize**.

24.3.1.6 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 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 (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

24.3.2 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.

24.3.2.1 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

24.3.2.2 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.ReduceFontSize**.

Name	Type	Description
ReduceFontSize	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

24.3.3 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.

24.3.3.1 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

24.3.3.2 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.ReduceFontSize**.

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

24.4 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.

24.4.1 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, 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>).

24.4.1.1 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
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

24.4.1.2 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
ReplaceSpecialCharacters	bool	change symbols '<', '>', '&', '\"' on < > &quot; by default true

24.4.2 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.

24.4.2.1 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 StiHtmlExportQuality.High
Encoding	Encoding	file encoding; by default Encoding.UTF8

Name	Type	Description
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

24.5 Text Formats

This chapter describes exports formats of text files. In other words the files which are used to create text documents.

24.5.1 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.

24.5.1.1 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.

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

Name	Type	Description
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

24.5.1.2 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, **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 check false status; by default "-"

24.5.2 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.

24.5.2.1 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

24.5.2.2 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
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true

Name	Type	Description
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

24.5.3 Word 2007/2010

Microsoft Word is a text processing software produced 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.

24.5.3.1 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

24.5.3.2 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
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

24.5.4 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.

When exporting the report is converted into a single table. The document is easily editable but some objects can be changed.

24.5.4.1 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) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution, dot per inch; may have any value, by default 100

24.5.4.2 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
RemoveEmptySpaceAtBottom	bool	remove empty space on the bottom of a page; by default true

24.6 Spreadsheets

This group of exports create spreadsheets. They are exports to both different formats of Microsoft Excel and to OpenOffice Calc.

24.6.1 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.

24.6.1.1 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) to 1.0 (the highest quality); by default 0.75
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 set to true.

24.6.1.2 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 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

24.6.2 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.

24.6.2.1 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) to 1.0 (the highest quality); by default 0.75
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**.

24.6.2.2 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

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

24.6.3 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.

24.6.3.1 Export Settings

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) to 1.0 (the highest quality); by default 0.75
ImageResolution	float	image resolution, dot per inch; may have any value, by default 100

24.6.3.2 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

24.7 Data

This is a group of file formats which are used to store table data.

24.7.1 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.

! Notice: 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.

24.7.1.1 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

SkipColumnHeaders	bool	skip headers of columns; by default false
-------------------	------	---

24.7.1.2 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 is used; by default - empty string

24.7.2 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.

⚠️ Notice: Only data can be exported to the DBF format, in other words only the components, which are placed on data bands.

24.7.2.1 Controlling Exports

The following elements can be specified in the Tag field to control export:

- ▷ **DataType** [: **FieldLength** [: **DecimalPartLength**]]
- ▷ **ExportType** : "FieldName"
- ▷ **Column**: "FieldName" "DataString"

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

! Notice: 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}"
```

24.7.2.2 Export Settings

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
-----------------	-----------------	---

24.7.3 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 that is used to store any data. Visually this structure can be represented as the tree. XML supports Unicode and other encoding.

! Notice: Only those data (components) are exported to the XML format which are placed on data bands.

24.7.3.1 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 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}"
```

24.7.4 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.

24.7.4.1 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

24.7.5 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.

! Notice: 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.

24.7.5.1 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

24.8 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 monochrome image is supported only to the BMP, GIF, PCX, PNG, TIFF format. So the DitheringType property works only for these exports.

24.8.1 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 StiMonochromeDitheringType.FloydSteinberg

25 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.

25.1 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:

```
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:

```
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:

```
StiReport masterReport = new StiReport();
masterReport.Load("d:\\master-detail.mrt");

StiReport report = new StiReport();
report.RegData(dataSet);

report.MasterReport = masterReport.SaveToString();
report.Design();
```

26 Scripts

Stimulsoft Reports supports a choice of languages for report generation.

26.1 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

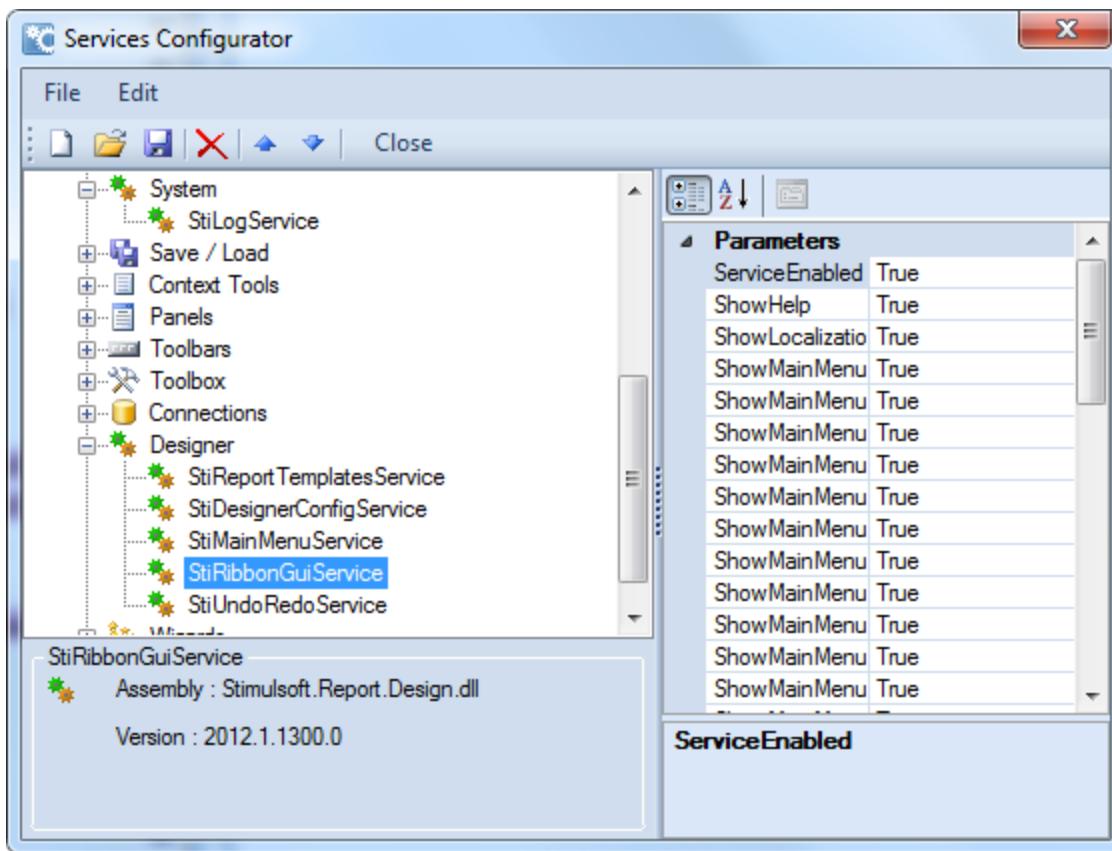
In the Language group select a new programming language and press Ok. The current programming language will then be changed.

Important: 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.

Changing The Default Language

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 drag the StiCSharpLanguage service down one position with the mouse or use the up and down buttons to re-order the languages.



26.2 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.

Note: The report code is generated in **C#** or **VB.Net** programming language. All events and any other code in this report must be written in the currently selected language.

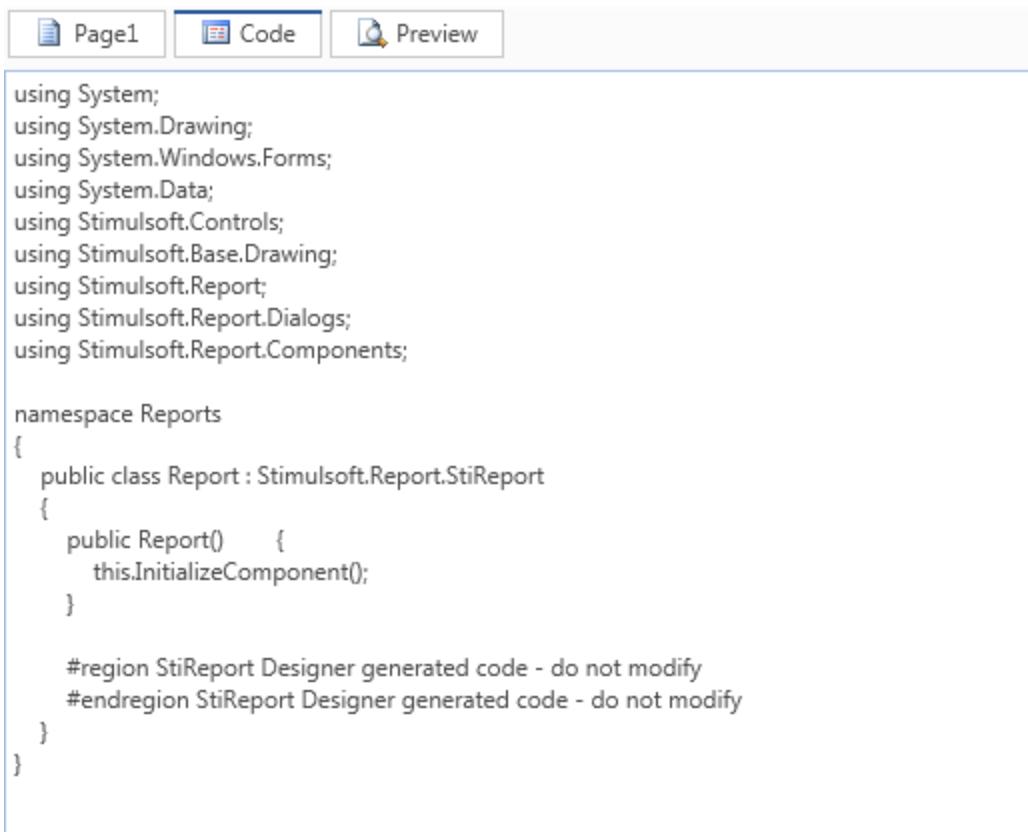
When rendering reports, compilation of the report class occurs first. After that the compiled report is

executed.

Note: The report code is compiled using the .NET Framework compiler.

Reviewing and Editing the Report Code

To see the report code click the Code tab in the designer. The code will then be displayed:



```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Data;
using Stimulsoft.Controls;
using Stimulsoft.Base.Drawing;
using Stimulsoft.Report;
using Stimulsoft.Report.Dialogs;
using Stimulsoft.Report.Components;

namespace Reports
{
    public class Report : Stimulsoft.Report.StiReport
    {
        public Report()
        {
            InitializeComponent();
        }

        #region StiReport Designer generated code - do not modify
        #endregion StiReport Designer generated code - do not modify
    }
}
```

To edit the code, simply start typing in the appropriate place.

Important: 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:

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.

27 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.

27.1 WinForms Report 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:

```
StiOptions.Viewer.RightToLeft = StiRightToLeftType.No;
```

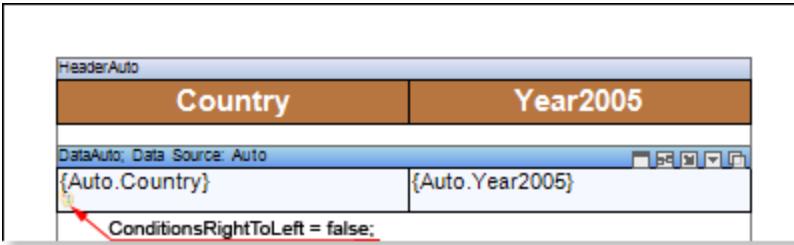
If the **RightToLeft** property is set to **Yes**, then viewer items and report pages in a viewer window are displayed from right to left. The code for setting the right to left mode is shown below:

```
StiOptions.Viewer.RightToLeft = StiRightToLeftType.Yes;
```

27.2 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**, **InheritedRightToLeft**, **InteractionCollapsingRightToLeft**, **InteractionSortRightToLeft**, **OrderAndQuickInfoRightToLeft**, **FiltersRightToLeft**, **QuickButtonsRightToLeft**. They belong to the **StiOptions.Viewer.Pins** class. Consider these features in more detail:

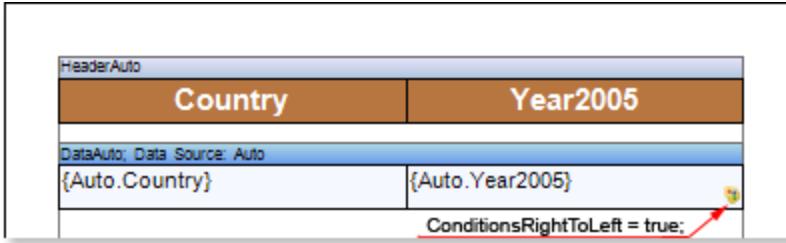
► The mode of displaying the **Conditions** icon depends on the value of the **ConditionsRightToLeft** property. For example, if to place a **Condition** in the **DataBand**, then the **Conditions** icon will be displayed by default in the lower left corner of this **DataBand**, because the **ConditionsRightToLeft** 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:



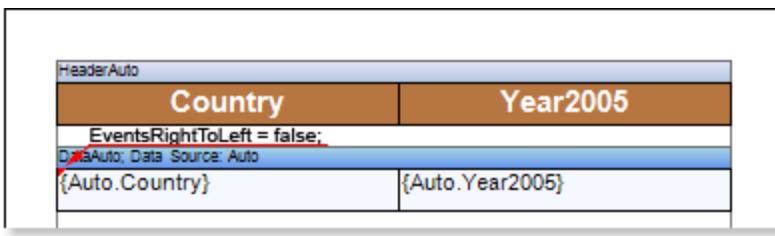
Set the **ConditionsRightToLeft** property to true to change the position of the icon:

```
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:



► 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.

```
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:

HeaderAuto	
Country	Year2005
{Auto.Country}	{Auto.Year2005}

► 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:

InheritedRightToLeft = false;	
DataProducts; Data Source: Products	
{Products.ProductName}	{Products.UnitPrice}

If the **InheritedRightToLeft** property is set to **true**

```
StiOptions.Viewer.Pins.InheritedRightToLeft = true;
```

the **Inherited** icon will appear in the right to left mode (see the picture below):

InheritedRightToLeft = true;	
DataProducts; Data Source: Products	
{Products.ProductName}	{Products.UnitPrice}

► 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:

InteractionCollapsingRightToLeft = false;	
ProductName	UnitsInStock
Côte de Blaye	17
Chartreuse verte	69
Steeleye Stout	20
Guaraná Fantástica	20
Sasquatch Ale	111
Chai	39

If the **InteractionCollapsingRightToLeft** property is set to **true**

```
StiOptions.Viewer.Pins.InteractionCollapsingRightToLeft = true;
```

Collapsing icons will appear in the right to left mode (see the picture below):

InteractionCollapsingRightToLeft = true;	
ProductName	UnitsIn Stock
	1
Côte de Blaye	17
Chartreuse verte	69
Steakhouse Stout	20
Guaraná Fantástica	20
Sasquatch Ale	111
Chai	39
	2

► 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:

InteractionSortRightToLeft = false;	
ProductName	UnitsIn Stock
Alice Mutton	0
Aniseed Syrup	13
Boston Crab Meat	123
Camembert Pierrot	19
Carnarvon Tigers	42

If the **InteractionSortRightToLeft** property is set to **true**

```
StiOptions.Viewer.Pins.InteractionSortRightToLeft = true;
```

the **InteractionSort** icon will appear in the right to left mode (see the picture below):

InteractionSortRightToLeft = true;	
ProductName	UnitsIn Stock
Zaanse koeken	36
Wimmers gute Semmelknödel	22
Vegie-spread	24
Valkoinen suklaa	65
Uncle Bob's Organic Dried Pears	15

► 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:

HeaderBand1	
0.0 ProductName	0.2 UnitsInStock
DataProducts; Data Source: Products	
1.0 [Products2.ProductName]	1.2 [Products2.UnitsInStock]
{Products1.ProductName}	{Products1.UnitsInStock}

If the **OrderAndQuickInfoRightToLeft** property is set to **true**:

```
StiOptions.Viewer.Pins.InteractionSortRightToLeft = true;
```

the **Show Order** icon will appear in the right to left mode (see the picture below):

HeaderBand1	
Product Name	Units In Stock
DataProducts; Data Source: Products	
[Products2.ProductName] 1.0	[Products2.UnitsInStock] 1.2
{Products1.ProductName}	{Products1.UnitsInStock}

► 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:

DataBand1; Data Source: Customers	
CustomerID	CustomerName

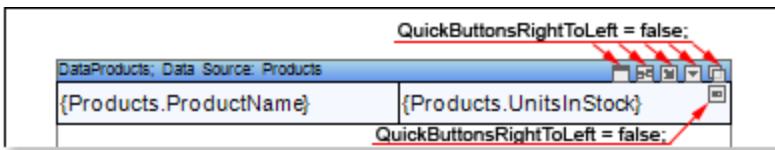
If the **FiltersRightToLeft** property is set to **true**:

```
StiOptions.Viewer.Pins.FiltersRightToLeft = true;
```

the **Filters** icon will appear in the right to left mode (see the picture below):

DataBand1; Data Source: Customers	
CustomerID	CustomerName

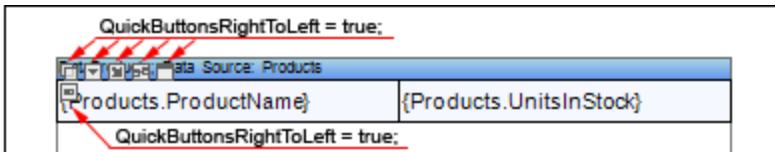
► The **QuickButtonsRightToLeft** property is used to change the location of the **QuickButtons** icon. By default, this property is set to **false**, i.e. the icon appears in the left to right mode. The picture below shows an example of a report template with the **Filters** icon in the left to right mode:



If the **QuickButtonsRightToLeft** property is set to **true**:

```
StiOptions.Viewer.Pins.QuickButtonsRightToLeft = true;
```

the **QuickButtons** icon will appear in the right to left mode (see the picture below):



27.3 WPF Report Designer and Viewer

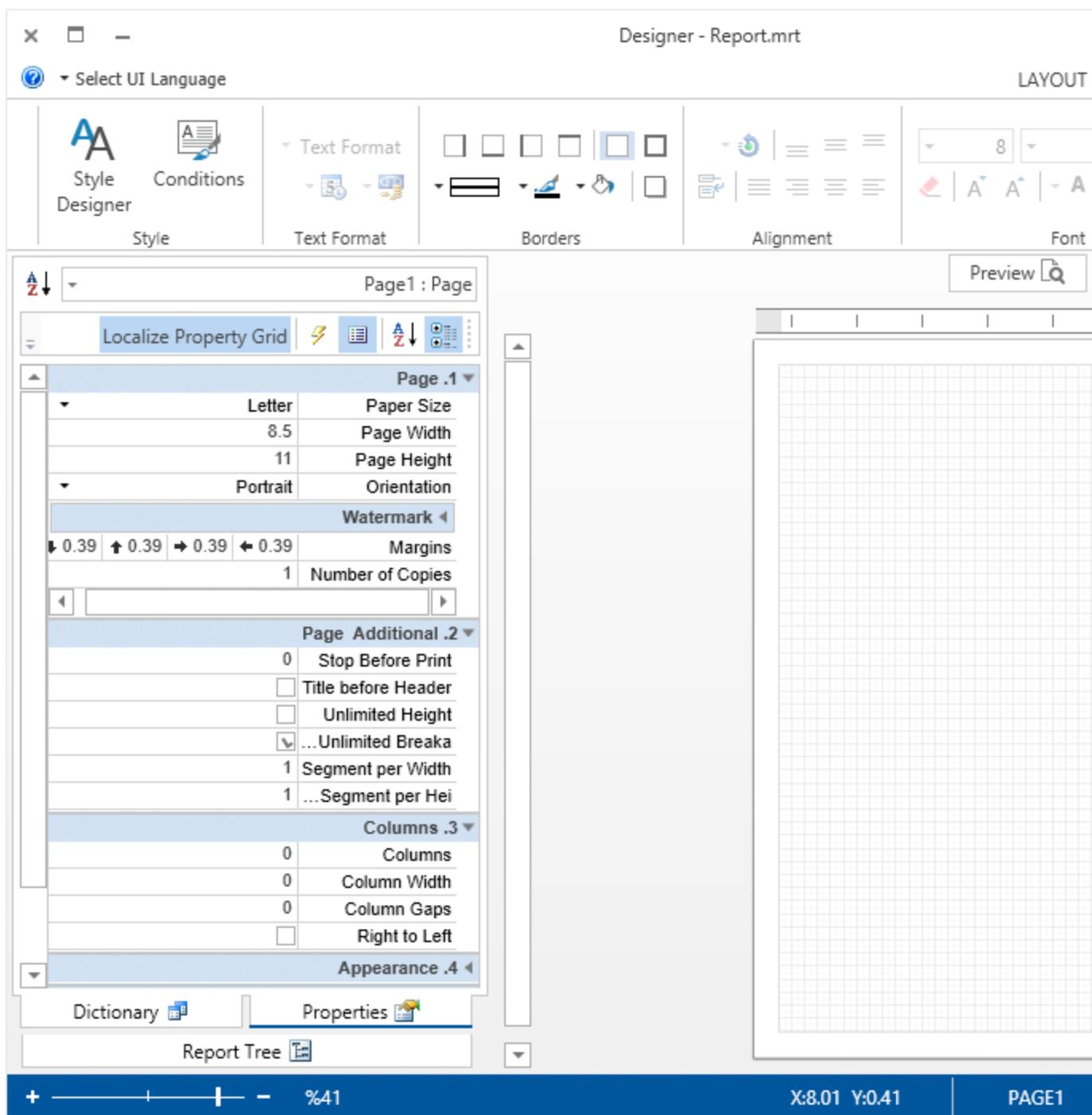
In the designer and WPF report viewer it is possible to set the mode of showing controls in the right to left direction. By default, all controls are displayed in the left to right order. It is possible to change this. The **FlowDirection** property is used for this. If the property is set to **LeftToRight**, then controls are shown from left to right (see the code below).

```
FlowDirection="LeftToRight"
```

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.

```
FlowDirection="RightToLeft"
```

The picture below shows the right to left order of showing controls in the designer and viewer.



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.

28 Deployment

28.1 Assemblies in Reports.Net

Stimulsoft Reports.Net is delivered with the following assemblies:

Assemblies	Description
Stimulsoft.Base.dll	The baseline assembly of the report generator.
Stimulsoft.Editor.dll	The Text editor used in the report designer.
Stimulsoft.Controls.dll	Controls used in the report generator.
Stimulsoft.Controls.Win.dll	Controls used in the report generator.
Stimulsoft.Report.dll	This baseline assembly contains the base functionality of the report generator.
Stimulsoft.Report.Chart.dll	This baseline assembly contains the core of the chart component.
Stimulsoft.Design.dll	This assembly contains Visual Studio design-time part of the StiReport class and StiDesignerControl class.
Stimulsoft.Report.WinForms.dll	This assembly contains WinForms Viewer.
Stimulsoft.Report.Design.dll	This assembly contains WinForms Designer.
Stimulsoft.Report.Check.dll	This assembly contains core of Report Checker tool.
Stimulsoft.Report.Helper.dll	This assembly contains additional information about localization of the report engine.
Stimulsoft.Database.dll	This dll file is used for creating and editing a connections string. Also this dll contains QueryBuilder.
Stimulsoft.Report.Web.dll	This assembly helps showing reports in ASP.NET and contains Web Viewer.
Demo Applications:	
Demo.exe	This application demonstrates the basic capabilities of Stimulsoft Reports.Net.
Tools:	
Configurator.exe	This application is used for editing the Stimulsoft Reports.Net configuration.
Installer.exe	This application helps to install Stimulsoft Reports.Net into GAC. Also it helps in using NGEN utility.
Browser.exe	This application is used for reports browsing.

Designer.exe	Stimulsoft Reports.Net report designer.
StyleDesigner.exe	Application that helps to design reports styles.
ReportChecker.exe	The utility is used for checking reports on issues.
Viewer.exe	Stimulsoft Reports.Net report viewer.
Import.CrystalReports.exe	This application is used for converting the CrystalReports templates to the Stimulsoft Reports.Net templates.
Import.ActiveReports.exe	This application is used for converting the ActiveReports templates to the Stimulsoft Reports.Net templates.
Import.FastReports.exe	This application is used for converting the FastReport.Net templates to the Stimulsoft Reports.Net templates.
Import.Rdl.exe	This application is used for converting the RDL (Report Definition Language) templates to the Stimulsoft Reports.Net templates.
Import.ReportSharpShooter.exe	This application is used for converting the Report Sharp-Shooter templates to the Stimulsoft Reports.Net templates.
Import.Rtf.exe	This application is used for converting the RTF (Rich Text Format) files to the Stimulsoft Reports.Net templates.
Import.XtraReports.exe	This application is used for converting the XtraReports templates to the Stimulsoft Reports.Net templates.

28.2 Assemblies in Reports.Wpf

Stimulsoft Reports.Wpf is delivered with the following assemblies:

Assemblies	Description
Stimulsoft.Base.dll	The baseline assembly of the report generator.
Stimulsoft.Editor.dll	The Text editor used in the report designer.
Stimulsoft.Controls.dll	Controls used in the report generator.
Stimulsoft.Controls.Wpf.dll	This assembly contains additional WPF controls.
Stimulsoft.Report.Wpf.dll	This assembly contains the WPF Viewer.
Stimulsoft.Report.Wpf.Design.dll	This assembly contains Visual Studio design-time part for the WPF viewer and WPF designer.
Stimulsoft.Report.Xbap.dll	This assembly contains the XBAP Viewer.
Stimulsoft.Report.Xbap.Design.dll	This assembly contains Visual Studio design-time part for the XBAP viewer.

Stimulsoft.Report.WpfDesigner.dll	This assembly contains WPF Designer.
Stimulsoft.Report.Check.dll	This assembly contains core of Report Checker tool.
Stimulsoft.Report.Helper.dll	This assembly contains additional information about localization of the report engine.
Stimulsoft.Database.dll	This dll file is used for creating and editing a connections string. Also this dll contains QueryBuilder.

WPF theme assemblies:

Stimulsoft.Report.Wpf.BlackTheme.dll	This assembly contains the Black theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2003BlueTheme.dll	This assembly contains the Office 2003 Blue theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2003OliveGreenTheme.dll	This assembly contains the Office 2003 Olive Green theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2003SilverTheme.dll	This assembly contains Office 2003 Silver theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2007BlackTheme.dll	This assembly contains Office 2007 Black theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2007BlueTheme.dll	This assembly contains Office 2007 Blue theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2007SilverTheme.dll	This assembly contains Office 2007 Silver theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2010BlueTheme.dll	This assembly contains Office 2010 Blue theme for the WPF report viewer and WPF report designer.

Demo Applications:

Demo.exe	This application shows the basic features of Stimulsoft Reports.Wpf
-----------------	---

Tools:

Installer.exe	This application helps to install Stimulsoft Reports.Net into GAC. Also it helps in using NGEN utility.
Designer.exe	Stimulsoft Reports.Wpf report designer.
StyleDesigner.exe	Application that helps to design reports styles.
ReportChecker.exe	The utility is used for checking reports on issues.

Import.CrystalReports.exe	This application is used for converting the CrystalReports templates to the Stimulsoft Reports.Net templates.
Import.ActiveReports.exe	This application is used for converting the ActiveReports templates to the Stimulsoft Reports.Net templates.
Import.FastReports.exe	This application is used for converting the FastReport.Net templates to the Stimulsoft Reports.Net templates.
Import.Rdl.exe	This application is used for converting the RDL (Report Definition Language) templates to the Stimulsoft Reports.Net templates.
Import.ReportSharpShooter.exe	This application is used for converting the Report Sharp-Shooter templates to the Stimulsoft Reports.Net templates.
Import.Rtf.exe	This application is used for converting the RTF (Rich Text Format) files to the Stimulsoft Reports.Net templates.
Import.XtraReports.exe	This application is used for converting the XtraReports templates to the Stimulsoft Reports.Net templates.

28.3 Assemblies in Reports.Web

Stimulsoft Reports.Web is delivered with the following assemblies:

Assemblies	Description
Stimulsoft.Base.dll	The baseline assembly of the report generator.
Stimulsoft.Editor.dll	The Text editor used in the report designer.
Stimulsoft.Controls.dll	Controls used in the report generator.
Stimulsoft.Controls.Win.dll	Controls used in the report generator.
Stimulsoft.Report.dll	This baseline assembly contains the base functionality of the report generator.
Stimulsoft.Report.Check.dll	This assembly contains a core of the Report Checker tool.
Stimulsoft.Report.Chart.dll	This baseline assembly contains a core of the chart component.
Stimulsoft.Report.Helper.dll	This assembly contains additional information about localization of report engine.
Stimulsoft.Report.Design.dll	This assembly contains WinForms Designer.
Stimulsoft.Report.Design.WebViewer.dll	This assembly contains WebViewer.Fx for using in WinForms report designer for showing a report with help of the Web preview tab.
Stimulsoft.Database.dll	This dll is used for creating and editing connections string. Also this dll contains QueryBuilder.

Stimulsoft.Report.Web.dll	This assembly helps to show reports in ASP.NET and contains Web Viewer.
Stimulsoft.Report.WebFx.dll	This assembly helps to show reports in ASP.NET with help of Adobe Flash and contains WebViewer.Fx.
Stimulsoft.Report.WebDesigner.dll	This assembly contains Web Designer.

Demo application:

Demo.Web.exe	This application demonstrates the potential of Stimulsoft Reports.Web.
---------------------	--

Tools:

Import.CrystalReports.exe	This application is used to convert the CrystalReports templates to the Stimulsoft Reports templates.
Installer.exe	This application helps to install Stimulsoft Reports into GAC. Also it helps in using NGEN utility.
Designer.exe	Stimulsoft Reports report designer.
ReportChecker.exe	The utility for checking reports on issues.
StyleDesigner.exe	This application helps to design report styles.
Import.ActiveReports.exe	This application is used for converting the ActiveReports templates to the Stimulsoft Reports.Net templates.
Import.FastReports.exe	This application is used for converting the FastReport.Net templates to the Stimulsoft Reports.Net templates.
Import.Rdl.exe	This application is used for converting the RDL (Report Definition Language) templates to the Stimulsoft Reports.Net templates.
Import.ReportSharpShooter.exe	This application is used for converting the Report Sharp-Shooter templates to the Stimulsoft Reports.Net templates.
Import.Rtf.exe	This application is used for converting the RTF (Rich Text Format) files to the Stimulsoft Reports.Net templates.
Import.XtraReports.exe	This application is used for converting the XtraReports templates to the Stimulsoft Reports.Net templates.

28.4 Assemblies in Reports Designer.Web

Stimulsoft Reports Designer.Web is delivered with the following assemblies:

Assemblies	Description
------------	-------------

Stimulsoft.Report.WebDesign.dll	Assembly that contains the Web Designer.
--	--

28.5 Assemblies in Reports.Silverlight

Assemblies	Description
Stimulsoft.Base.dll	The baseline assembly of the report generator.
Stimulsoft.Editor.dll	The Text editor used in the report designer.
Stimulsoft.Controls.dll	Controls used in the report generator.
Stimulsoft.Controls.Win.dll	Controls used in the report generator.
Stimulsoft.Report.dll	This baseline assembly contains the base functionality of the report generator.
Stimulsoft.Report.Check.dll	This assembly contains a core of the Report Checker tool.
Stimulsoft.Report.Chart.dll	This baseline assembly contains a core of the chart component.
Stimulsoft.Report.Helper.dll	This assembly contains additional information about localization of report engine.
Stimulsoft.Report.Design.dll	This assembly contains WinForms Designer.
Stimulsoft.Report.Design.SLViewer.dll	This assembly contains Silverlight Viewer for using in WinForms report designer for showing reports with help of the Silverlight preview tab.
Stimulsoft.Database.dll	This dll is used for creating and editing connections string. Also this dll contains QueryBuilder.

Silverlight Assemblies:

Stimulsoft.Base.SL.dll	This is a baseline assembly of the report generator. Its Silverlight based assembly.
Stimulsoft.Compression.SL.dll	The assembly provides compression methods for using in the report generator. It is a Silverlight based assembly.
Stimulsoft.Controls.SL.dll	Controls which are used in the report generator. It is a Silverlight based assembly.
Stimulsoft.Report.SL.dll	This baseline assembly contains the base functionality of the report generator. It is a Silverlight based assembly.
Stimulsoft.Report.Check.SL.dll	This assembly contains core of the Report Checker tool for Silverlight.

Stimulsoft.Report.Helper.SL.dll	This assembly contains additional information about localization of the report engine for Silverlight.
Stimulsoft.Report.SLDesign.dll	This assembly contains Silverlight Report Designer. It is a Silverlight based assembly.
Stimulsoft.Report.Design.SLViewer.dll	This assembly contains Silverlight Viewer for using in the WinForms report designer for showing report with help of Silverlight preview tab.
Stimulsoft.Report.Viewer.SL.dll	This assembly contains Silverlight Reports Viewer. It is a Silverlight based assembly.
Stimulsoft.Report.WebSL.dll	This assembly contains Silverlight Reports Viewer (Client/Server version) – ASP.NET based control.
Stimulsoft.Report.WebDesignerSL.dll	This assembly contains Silverlight Report Designer (Client/Server version) – ASP.NET based control.

Demo application:

Demo.SL	This application demonstrates the basic capabilities of Stimulsoft Reports.Silverlight. It is a Silverlight based application.
Demo.Web.SL	This application demonstrates the basic capabilities of Stimulsoft Reports.Silverlight. It is a Client-Server based application.

Tools:

Designer.SL	Stimulsoft Reports.Silverlight report designer.
Viewer.SL	Stimulsoft Reports.Silverlight reports viewer.
OptionsHelper.exe	This application is used for creating Options.xml file for Designer.exe application.
Import.CrystalReports.exe	This application is used to convert the CrystalReports templates to the Stimulsoft Reports templates.
Installer.exe	This application helps to install Stimulsoft Reports into GAC. Also it helps in using NGEN utility.
Designer.exe	Stimulsoft Reports report designer.
ReportChecker.exe	The utility for checking reports on issues.
StyleDesigner.exe	This application helps to design report styles.
Import.ActiveReports.exe	This application is used for converting the ActiveReports templates to the Stimulsoft Reports.Net templates.
Import.FastReports.exe	This application is used for converting the FastReport.Net templates to the Stimulsoft Reports.Net templates.
Import.Rdl.exe	This application is used for converting the RDL (Report Definition

	Language) templates to the Stimulsoft Reports.Net templates.
Import.ReportSharpShooter.exe	This application is used for converting the Report Sharp-Shooter templates to the Stimulsoft Reports.Net templates.
Import.Rtf.exe	This application is used for converting the RTF (Rich Text Format) files to the Stimulsoft Reports.Net templates.
Import.XtraReports.exe	This application is used for converting the XtraReports templates to the Stimulsoft Reports.Net templates.

28.6 Assemblies in Reports Designer.Silverlight

Stimulsoft Reports Designer.Silverlight is delivered with the following assemblies:

Assemblies	Description
Stimulsoft.Report.WebDesignerSL.dll	Assembly which contains Silverlight report designer (Client/Server version) – Asp.Net based control.

28.7 Assemblies in Reports.Ultimate

Assemblies	Description
Stimulsoft.Base.dll	The baseline assembly of the report generator.
Stimulsoft.Editor.dll	The Text editor used in the report designer.
Stimulsoft.Controls.dll	Controls used in the report generator.
Stimulsoft.Controls.WinForms.dll	Controls used in the report generator.
Stimulsoft.Report.dll	This baseline assembly contains the base functionality of the report generator.
Stimulsoft.Report.Chart.dll	This baseline assembly contains the core of the chart component.
Stimulsoft.Design.dll	This assembly contains Visual Studio design-time part of the StiReport class and StiDesignerControl class.
Stimulsoft.Report.WinForms.dll	This assembly contains WinForms Viewer.
Stimulsoft.Report.Design.dll	This assembly contains WinForms Designer.
Stimulsoft.Report.Check.dll	This assembly contains core of Report Checker tool.
Stimulsoft.Report.Helper.dll	This assembly contains additional information about localization of the

II	report engine.
Stimulsoft.Database.dll	This dll file is used for creating and editing a connections string. Also this dll contains QueryBuilder.
Stimulsoft.Report.Web.dll	This assembly helps showing reports in ASP.NET and contains Web Viewer.
Stimulsoft.Report.WebDesigner.dll	This assembly contains Web Designer.
Stimulsoft.Report.WebFx.dll	This assembly helps to show reports in ASP.NET with help of Adobe Flash and contains WebViewer.Fx.
Stimulsoft.Database.Wpf.dll	This dll is used for creating and editing a connections string. Also this dll contains QueryBuilder.
Stimulsoft.Report.Design.SilverlightViewer.dll	The assembly contains Silverlight Viewer for using in WinForms report designer for showing report with help of Silverlight preview tab.
Stimulsoft.Database.Wpf.dll	This is a WPF version of Stimulsoft.Database.dll. This dll is used for creating and editing connections string. Also this dll contains QueryBuilder.
Stimulsoft.Controls.Wpf.dll	This assembly contains additional WPF controls.
Stimulsoft.Report.Wpf.dll	This assembly contains the WPF Viewer.
Stimulsoft.Report.Wpf.Design.dll	This assembly contains Visual Studio design-time part for the WPF viewer and WPF designer.
Stimulsoft.Report.Xbap.dll	This assembly contains the XBAP Viewer.
Stimulsoft.Report.Xbap.Design.dll	This assembly contains Visual Studio design-time part for the XBAP viewer.
Stimulsoft.Report.WpfDesigner.dll	This assembly contains WPF Designer.

WPF theme assemblies:

Stimulsoft.Report.Wpf.BlackTheme.dll	This assembly contains Black theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2003BlueTheme.dll	This assembly contains Office 2003 Blue theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2003OliveGreenTheme.dll	This assembly contains Office 2003 Olive Green theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2003SilverTheme.dll	This assembly contains Office 2003 Silver theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Offi	This assembly contains Office 2007 Black theme for the WPF report

ce2007BlackTheme.dll	viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2007BlueTheme.dll	This assembly contains Office 2007 Blue theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2007SilverTheme.dll	This assembly contains Office 2007 Silver theme for the WPF report viewer and WPF report designer.
Stimulsoft.Report.Wpf.Office2010BlueTheme.dll	This assembly contains Office 2010 Blue theme for the WPF report viewer and WPF report designer.

Silverlight Assemblies:

Stimulsoft.Base.SL.dll	The baseline assembly of the report generator. It is a Silverlight based assembly.
Stimulsoft.Compression.SL.dll	This assembly provides compression methods for using in the report generator. It is a Silverlight based assembly.
Stimulsoft.Controls.SL.dll	Controls used in the report generator. It is a Silverlight based assembly.
Stimulsoft.Report.SL.dll	The baseline assembly contains the basic functionality of the report generator. It is a Silverlight based assembly.
Stimulsoft.Report.Check.SL.dll	This assembly contains core of the Report Checker tool for Silverlight.
Stimulsoft.Report.Helper.SL.dll	This assembly contains additional information about localization of the report engine for Silverlight.
Stimulsoft.Report.SLDesigner.dll	This assembly contains Silverlight Report Designer. It is a Silverlight based assembly.
Stimulsoft.Report.Viewer.SL.dll	This assembly contains Silverlight Reports Viewer. It is a Silverlight based assembly.
Stimulsoft.Report.WebSL.dll	This assembly contains Silverlight Reports Viewer (Client/Server version) – ASP.NET based control.
Stimulsoft.Report.WebDesignerSL.dll	This assembly contains Silverlight Report Designer (Client/Server version) – ASP.NET based control.

Demo Applications:

Demo.exe	This application demonstrates the basic capabilities of Stimulsoft Reports.Net.
Demo.Web.exe	This application demonstrates the potential of Stimulsoft Reports.Ultimate in Web.
Demo.Wpf.exe	This application demonstrates the potential of Stimulsoft Reports.Ultimate in WPF.
Demo.SL	This application demonstrates the potential of Stimulsoft

	Reports.Ultimate in Silverlight. It uses Native mode.
Demo.Web.SL	This application demonstrates the potential of Stimulsoft Reports.Ultimate in Silverlight. It uses Client-Server mode.
Tools:	
Configurator.exe	This application is used for editing the Stimulsoft Reports.Net configuration.
Installer.exe	This application helps to install Stimulsoft Reports.Net into GAC. Also it helps in using NGEN utility.
Browser.exe	This application is used for reports browsing.
Designer.exe	Stimulsoft Reports.Net report designer.
StyleDesigner.exe	Application that helps to design reports styles.
ReportChecker.exe	The utility is used for checking reports on issues.
Viewer.exe	Stimulsoft Reports.Net report viewer.
Import.CrystalReports.exe	This application is used for converting the CrystalReports templates to the Stimulsoft Reports.Net templates.
Import.ActiveReports.exe	This application is used for converting the ActiveReports templates to the Stimulsoft Reports.Net templates.
Import.FastReports.exe	This application is used for converting the FastReport.Net templates to the Stimulsoft Reports.Net templates.
Import.Rdl.exe	This application is used for converting the RDL (Report Definition Language) templates to the Stimulsoft Reports.Net templates.
Import.ReportSharpShooter.exe	This application is used for converting the Report Sharp-Shooter templates to the Stimulsoft Reports.Net templates.
Import.Rtf.exe	This application is used for converting the RTF (Rich Text Format) files to the Stimulsoft Reports.Net templates.
Import.XtraReports.exe	This application is used for converting the XtraReports templates to the Stimulsoft Reports.Net templates.
OptionsHelper.exe	This application is used for creating Options.xml file for Designer.exe application.
Designer.Wpf.exe	The report designer that uses the WPF technology.
StyleDesigner.Wpf.exe	This application helps to design reports styles. It uses WPF technology.
ReportChecker.Wpf.exe	The utility used for checking reports on issues. It uses WPF technology.
Designer.SL	The report designer that uses the Silverlight technology.
Viewer.exe	Stimulsoft Reports report viewer.

Viewer.Wpf.exe	Stimulsoft Reports report viewer. It uses WPF technology.
Viewer.SL	Stimulsoft Reports report viewer. It uses Silverlight technology.

28.8 Assemblies in Reports.Fx for Flex

Stimulsoft Reports.Fx for Flex is delivered with the following assemblies:

Assemblies	Description
Stimulsoft_ViewerFx.swc	This library contains the report engine and report viewer.
Stimulsoft_DesignerFx.swc	This library contains the report designer.
DesignerFx for Flex.exe	Stimulsoft Reports.Fx standalone report designer.
DemoFx for Flex.exe	The Adobe AIR application which demonstrates main features of Stimulsoft Reports.Fx.

28.9 Assemblies in Reports.Fx for PHP

Stimulsoft Reports.Fx for PHP is delivered with the following assemblies:

Assemblies	Description
DesignerFx for PHP.exe	Stimulsoft Reports.Fx standalone report designer.
DemoFx for PHP.exe	An Adobe AIR application which demonstrates basic features of Stimulsoft Reports.Fx.
localization (folder)	This folder contains all localization files. The list of localization files is indicated automatically when loading the designer or viewer.
config.xml	A configuration file which contains all available options.
swfobject.js playerProductInstall.swf	These files are used for loading the Flash content and Flash player updates.
designer.html viewer.html DesignerFx_PHP.swf ViewerFx_PHP.swf index.php handler.php localization.php database_xml database_mssql.php database_mysql.php	Stimulsoft Reports.Fx web based report designer and report viewer files.

database_odbc.php	
database_oracle.php	
database_pg.php	

28.10 Assemblies in Reports.Fx for Java

Stimulsoft Reports.Fx for Java is delivered with the following assemblies:

Assemblies	Description
stimulsoft.reports.fx-core-2011.1.1000.jar	This library contains the main part of the product.
stimulsoft.reports.fx-designer-2011.1.1000.jar	This library contains the report designer.
stimulsoft.reports.fx-viewer-2011.1.1000.jar	This library contains the report viewer.
stimulsoft.reports.fx-web-2011.1.1000.jar	This library is used for the report viewer and report designer in a web application.
stimulsoft.reports.fx-swt-2011.1.1000.jar	This library for using the report viewer and report designer in an swt application.
DesignerFx for Java.exe	Stimulsoft Reports.Fx standalone report designer.
DemoFx for Java.exe	An Adobe AIR application which demonstrates basic features of Stimulsoft Reports.Fx.

28.11 Working With Assemblies

There is no possibility of creating a method that compiles into memory and returns the assembly. This is because of the restrictions of .NET Framework. If, in the application, the .NET assembly is loaded, then the assembly is locked until the application closes even if the assembly is loaded from a stream. In this case, the assembly is stored in the **temp** directory of the **windows**, and then loaded into memory. Loading the report assemblies (like any other .NET assembly) is a file on disk in a particular place and memory for this, blocked by the assembly.

There are some ways to solve the issue:

- ▶ Use the reports in the application as **C#** classes (keeping them as classes in the report designer). In this case, the report is compiled with your application, and, therefore, it does not need compilation and loading assemblies. When updating, it is inconvenient to administer reports.
- ▶ Use reports as a compiled assembly, i.e. save reports as assemblies of the report designer. In this case, when updating reports, it is also too inconvenient to administer them.

- ▶ Do not use reports compilation. But use the interpretation of the expressions in the report. Interpreter mode can be enabled by means of the **CalculationMode** report property. However, one should consider that the expressions interpreter works slower than compiled expression. The interpreter cannot interpret all expressions of **C#** and **VB.NET**. Also, do not use events in the report components.
- ▶ Compilation and report rendering in another application domain. In this case, after rendering the report its assembly can be unloaded along with the entire application domain from memory. The locked assembly file can be removed after unload. To create a report, you can use the **CreateReportInNewAppDomain()** method of the **StiReport** class, for its upload use the **UnloadReportAppDomain()** method. However, in this case, the **RegData** method and **RegBusinessObject** work very slowly (as data marshaling in another domain is used). It is better request data from the report. Creating and deleting the application domain takes some time.
- ▶ One of the most effective ways is a report compilation when first calling the report. And with each next calling - report loading from the previously compiled assembly. In this case, the report is stored in memory only once, the assembly on disk only once. Before running the application the folder where reports are stored can be removed (before using report) as all reports will be recreated (when accessed). Here is the code that uses this method. New rendered report is created if it does not, or if the report has been modified, and if the version of the **.NET Framework**, which has been collected for the assembly, and the version under which the application is running is not the same. With this method, only the first run is slow.

Code Sample:

```
StiReport report = new StiReport();
report.Load(file);

string folder =
Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
folder = Path.Combine(folder, "Stimulsoft\\CompiledReports");
folder = Path.Combine(folder,
System.Runtime.InteropServices.RuntimeEnvironment.GetSystemVersion().ToString());
string compiledReportFile = Path.Combine(folder,
report.GetReportAssemblyCacheName());

if (File.Exists(compiledReportFile))
    report = StiReport.GetReportFromAssembly(compiledReportFile, true);
else
{
    if (!Directory.Exists(folder)) Directory.CreateDirectory(folder);
    report.Compile(compiledReportFile);
}
report.Render(false);
```

- ▶ Precompilation of reports updating them. This method is a variation of the previous one. After updating reports in the application, or if they are not present, or if necessary, start the compilation process of all reports, which are used in the application. Before compiling the old version assemblies is

removed. Sometimes the process of compiling the report for a given process may take some time.

28.12 Redistributable files in Reports.Net

The following assemblies and files of **Stimulsoft Reports.Net** can be delivered with the final application:

- ▷ Stimulsoft.Base.dll
- ▷ Stimulsoft.Controls.dll
- ▷ Stimulsoft.Controls.Win.dll
- ▷ Stimulsoft.Database.dll
- ▷ Stimulsoft.Editor.dll
- ▷ Stimulsoft.Report.dll
- ▷ Stimulsoft.Report.Win.dll
- ▷ Stimulsoft.Report.Design.dll
- ▷ Stimulsoft.Report.Web.dll
- ▷ Localization files

28.13 Redistributable files in Reports.Wpf

The following assemblies and files of Stimulsoft Reports.Wpf can be delivered with the final application:

- ▷ Stimulsoft.Base.dll
- ▷ Stimulsoft.Database.Wpf.dll
- ▷ Stimulsoft.Report.dll
- ▷ Stimulsoft.Report.Check.dll
- ▷ Stimulsoft.Report.Wpf.dll
- ▷ Stimulsoft.Report.Wpf.BlackTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2003BlueTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2003OliveGreenTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2003SilverTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2007BlackTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2007BlueTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2007SilverTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2010BlueTheme.dll
- ▷ Stimulsoft.Report.WpfDesign.dll
- ▷ Stimulsoft.Report.Helper.dll
- ▷ Localization files

28.14 Redistributable files in Reports.Web

The following assemblies and files of Stimulsoft Reports.Web can be delivered with the final application:

- ▷ Stimulsoft.Base.dll
- ▷ Stimulsoft.Report.dll

- ▶ Stimulsoft.Report.Web.dll
- ▶ Stimulsoft.Report.WebDesign.dll
- ▶ Stimulsoft.Report.WebFx.dll
- ▶ Localization files

Important: The Stimulsoft.Report.Design assembly cannot be distributed. The licensing per number of developers is required.

28.15 Redistributable files in Report Designer.Web

The following assemblies and files of Stimulsoft Reports Designer.Web can be delivered with the final application:

- ▶ Stimulsoft.Report.WebDesign.dll

28.16 Redistributable files in Reports.Silverlight

The following assemblies and files of Stimulsoft Reports.Silverlight can be delivered with the final application:

- ▶ Stimulsoft.Base.SL.dll
- ▶ Stimulsoft.Compression.SL.dll
- ▶ Stimulsoft.Controls.SL.dll
- ▶ Stimulsoft.Report.SL.dll
- ▶ Stimulsoft.Report.SLDesign.dll
- ▶ Stimulsoft.Report.Viewer.SL.dll
- ▶ Stimulsoft.Report.WebSL.dll
- ▶ Stimulsoft.Report.WebDesignSL.dll
- ▶ Localization files

28.17 Redistributable files in Reports Designer.Silverlight

The following assemblies and files of Stimulsoft Reports Designer.Silverlight can be delivered with the final application:

- ▶ Stimulsoft.Report.WebDesignSL.dll

28.18 Redistributable files in Reports.Ultimate

The following assemblies and files of Stimulsoft Reports.Ultimate can be delivered with the final application:

- ▶ Stimulsoft.Controls.dll

- ▷ Stimulsoft.Controls.Win.dll
- ▷ Stimulsoft.Base.dll
- ▷ Stimulsoft.Database.dll
- ▷ Stimulsoft.Database.Wpf.dll
- ▷ Stimulsoft.Editor.dll
- ▷ Stimulsoft.Report.dll
- ▷ Stimulsoft.Report.Check.dll
- ▷ Stimulsoft.Report.Win.dll
- ▷ Stimulsoft.Report.Web.dll
- ▷ Stimulsoft.Report.WebFx.dll
- ▷ Stimulsoft.Report.Wpf.dll
- ▷ Stimulsoft.Report.Wpf.BlackTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2003BlueTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2003OliveGreenTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2003SilverTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2007BlackTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2007BlueTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2007SilverTheme.dll
- ▷ Stimulsoft.Report.Wpf.Office2010BlueTheme.dll
- ▷ Stimulsoft.Report.Design.dll
- ▷ Stimulsoft.Report.DesignHelper.dll
- ▷ Stimulsoft.Report.WebDesign.dll
- ▷ Stimulsoft.Report.WpfDesign.dll
- ▷ Stimulsoft.Base.SL.dll
- ▷ Stimulsoft.Compression.SL.dll
- ▷ Stimulsoft.Controls.SL.dll
- ▷ Stimulsoft.Report.SL.dll
- ▷ Stimulsoft.Report.SLDesign.dll
- ▷ Stimulsoft.Report.Viewer.SL.dll
- ▷ Stimulsoft.Report.WebSL.dll
- ▷ Stimulsoft.Report.WebDesignSL.dll
- ▷ Localization files

28.19 Redistributable files in Reports.Fx for Flex

The following assemblies and files of Stimulsoft Reports.Fx for Flex can be delivered with the final application:

- ▷ Stimulsoft_ViewerFx.swc
- ▷ Stimulsoft_DesignerFx.swc
- ▷ Localization files

28.20 Redistributable files in Reports.Fx for PHP

The following assemblies and files of Stimulsoft Reports.Fx for PHP can be delivered with the final application:

- » Server-side files
- » Client-side files
- » Localization files

28.21 Redistributable files in Reports.Fx for Java

The following assemblies and files of Stimulsoft Reports.Fx for Java can be delivered with the final application:

- » All *.jar files.

28.22 Deployment in Windows

For distribution Stimulsoft Reports assemblies together with the application it is enough to copy them to the folder where the application file is saved. Registration to GAC is not required. Registration of COM objects is not required. Registration in registry is not required. Using the NGEN utility is not required.

28.23 Deployment in Web

For distribution Stimulsoft Reports.Web together with a web application it is enough to copy them to the bin folder of an application. Registration to GAC is not required. Registration of COM objects is not required. Registration in registry is not required. Using the NGEN utility is not required.

28.24 Deployment in Designer.Web

For distribution Stimulsoft Reports.Web together with a web application it is enough to copy them to the bin folder of an application. Registration to GAC is not required. Registration of COM objects is not required. Registration in registry is not required. Using the NGEN utility is not required. For working with the product Stimulsoft Reports.Net report generator is required.

28.25 Deployment Reports as Files

The most frequently used way of reports deployment is deployment reports as files. Stimulsoft Reports supports two type of files: unpacked report (the .mrt file extension) and packed reports (the .mrz file extension). Packed reports have small file size but it takes much time for saving and loading them. For loading the previously saved report the following code is required:

C#:

```
StiReport report = new StiReport();
```

```
report.Load("report.mrt");
```

VB.NET:

```
Dim Report As StiReport = New StiReport()
Report.Load("report.mrt")
```

For saving the report the following will be needed:

C#:

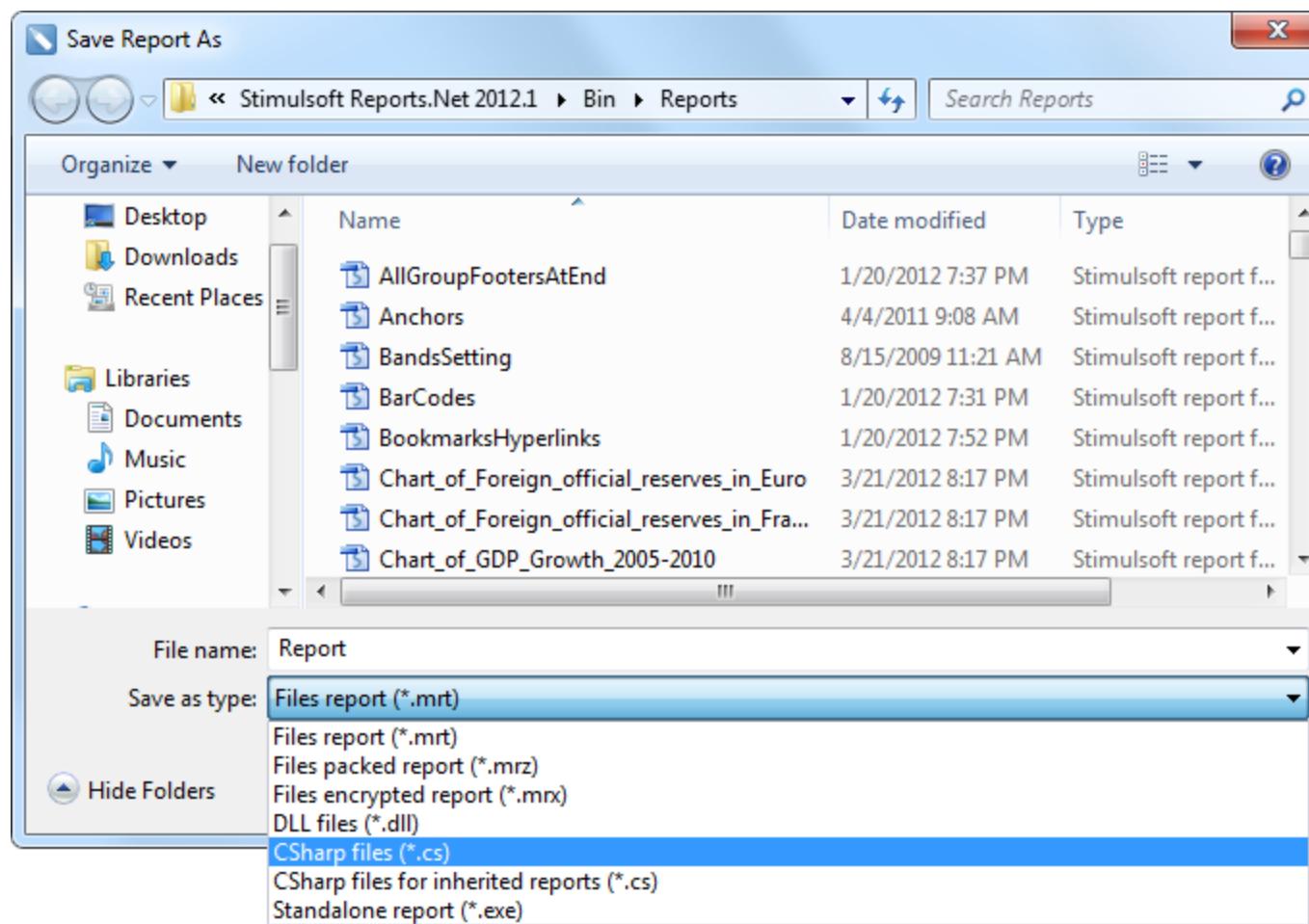
```
StiReport report = new StiReport();
report.Save("report.mrt");
```

VB.NET:

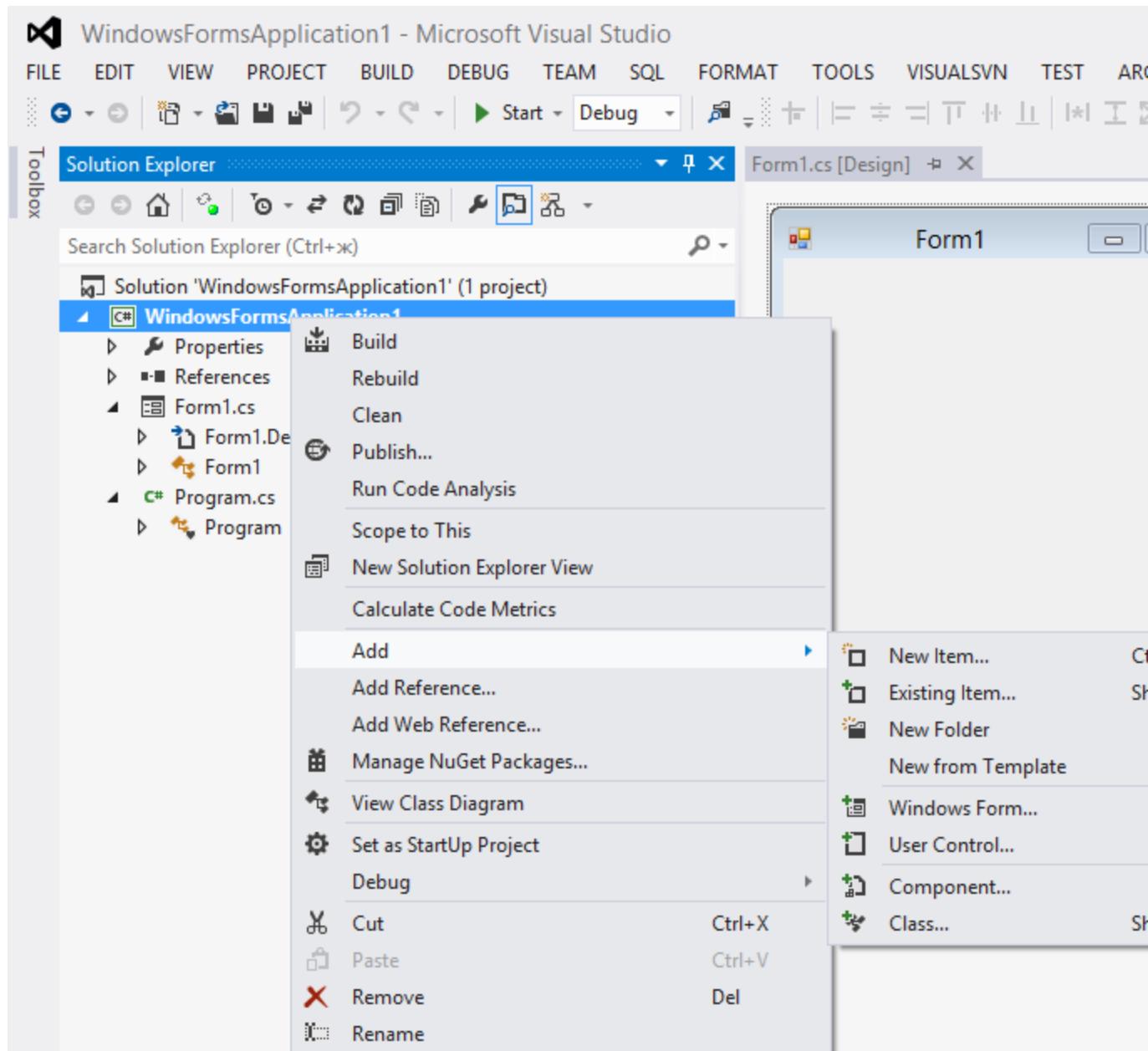
```
Dim Report As StiReport = New StiReport()
Report.Save("report.mrt")
```

28.26 Reports as Source Code

For automatic generation a report code the standard **.NET Framework** languages are used. When using **C#**, it is possible to write the same code as in **Visual Studio.NET**. The same can be said about **VB.NET**, where a report code can be saved and in the **Visual Studio.NET** project. Use the **File | Save Report As...** menu for saving a report code

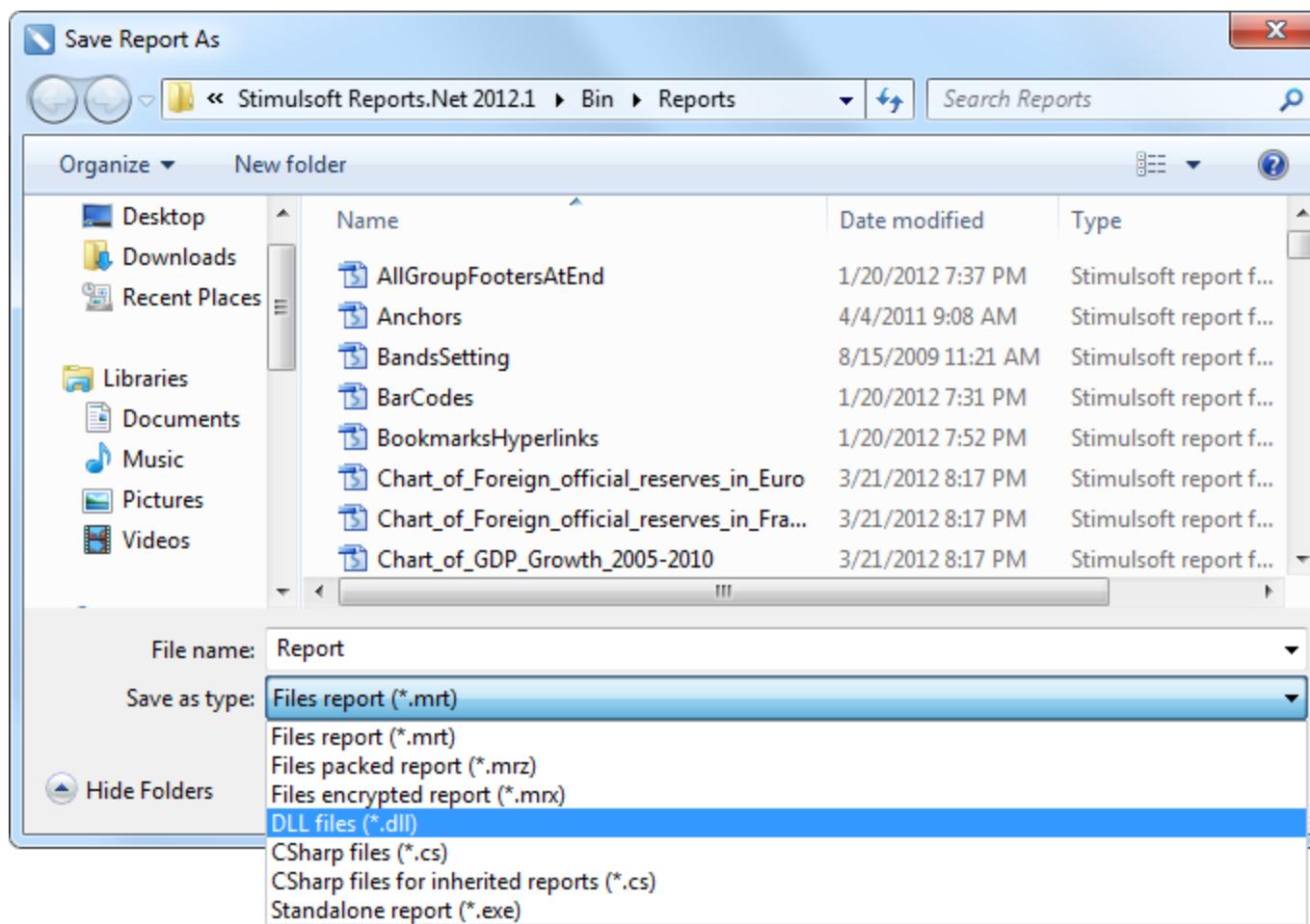


! Notice: The report code is completely compatible with C# and VB languages. Therefore, the report code can be saved and used in the Visual Studio.NET project.



28.27 Reports as Assemblies

Stimulsoft Reports generation has the unique ability to compile reports to the .Net Framework assembly. Use the **File | Save Report As...** menu for saving a report to the assembly.



For loading a report from the assembly the following code is used:

C#:

```
StiReport report = StiReport.GetReportFromAssembly("report.dll");
```

VB.NET:

```
Dim Report As StiReport = StiReport.GetReportFromAssembly("report.dll")
```

Reports, which are loaded from assembly, do not require compilation. It is impossible to edit such reports in the report designer.

28.28 Standalone Reports

It is required to use the program code to compile standalone reports. It is impossible to create standalone reports using the designer. The following code will create the standalone report:

C#:

```
StiReport report = new StiReport();
report.Load("report.mrt");
report.CompileStandaloneReport("report.exe",
StiStandaloneReportType.Show);
```

VB.NET:

```
Dim Report As New StiReport()
Report.Load("report.mrt")
Report.CompileStandaloneReport("report.exe",
StiStandaloneReportType.Show)
```

It is important to remember that for working with standalone reports the Stimulsoft Reports libraries are required. Besides, the report should get all data itself.