SharpShooter Reports.Silverlight Getting Started

Last modified on: January 13, 2012

Table of Contents

Introduction

The purpose of this user manual is to demonstrate the main points of using SharpShooter Reports.Silverlight software and provide necessary knowledge for getting started with the component. You will get step by step recommendations on how to create a web application using SharpShooter Reports.Silverlight. You will also look into the process of creating and configuring a service, designing a report and, finally, integrating a report viewer into the application pages.

Product Prerequisites

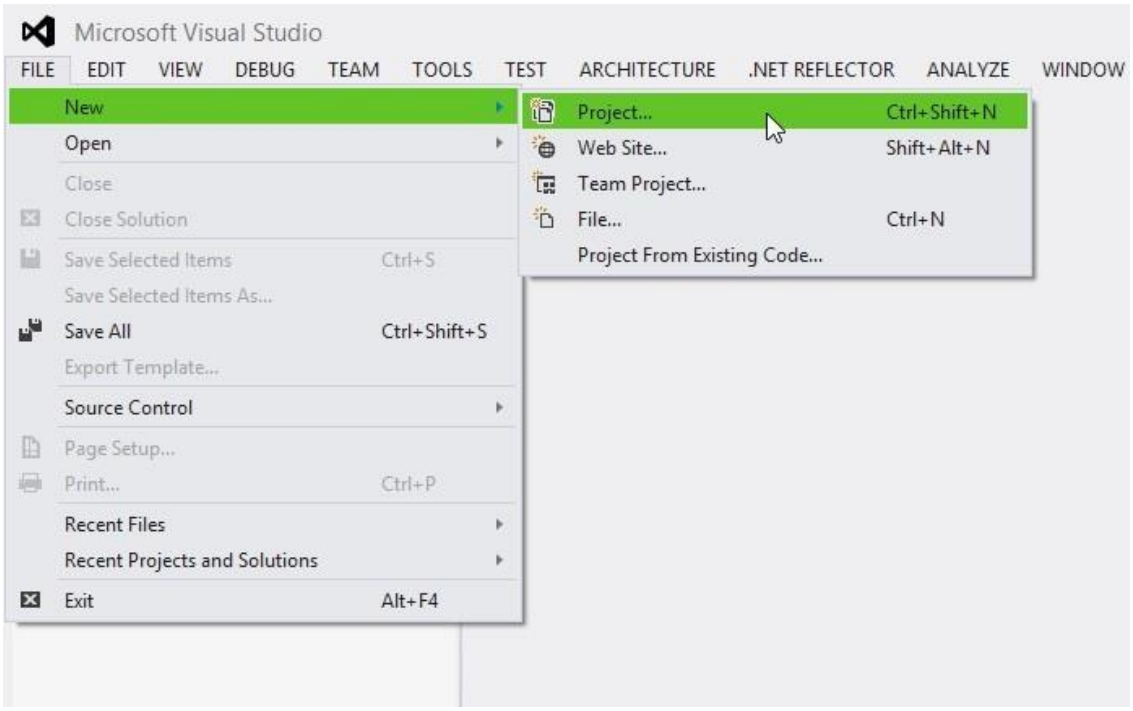
In order to get the sample working the following software should be installed on your computer:

- MS Visual Studio 2012
- .NET Framework 4.5
- ASP.NET 2.0
- Silverlight 5.0
- Microsoft Silverlight 5 Toolkit
- SharpShooter Reports.Silverlight 6.0.0.0 or higher.

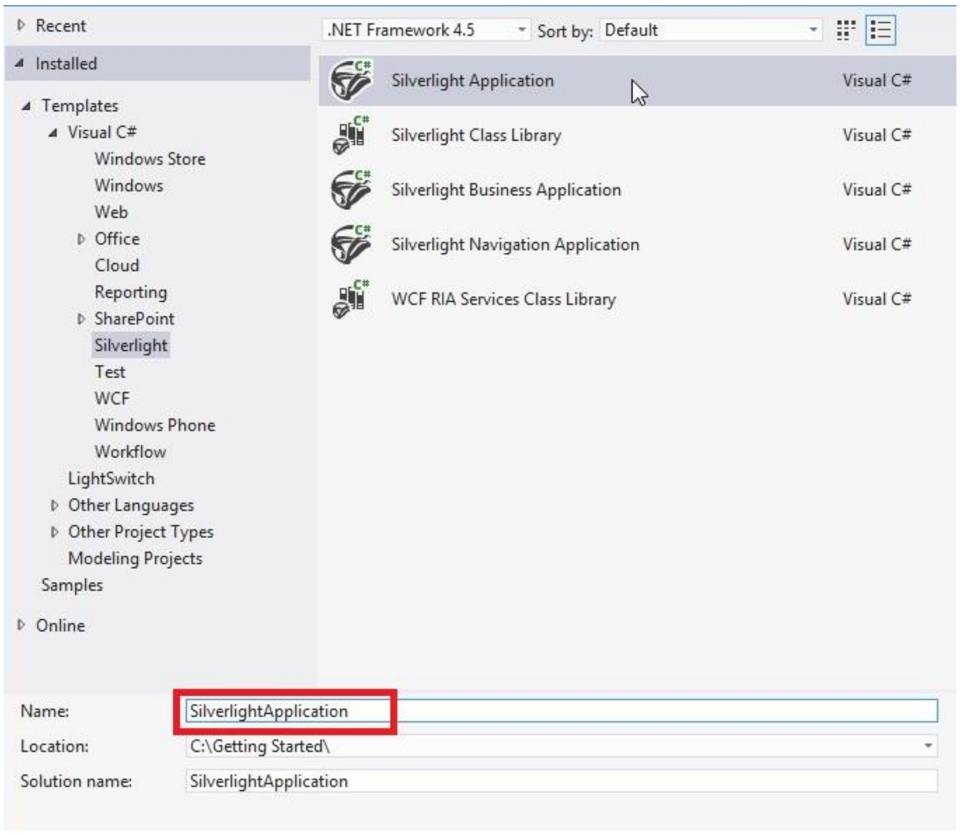
Creating a Sample Application

Step 1. Creating a Project

1) To begin with, we should create a new Silverlight Application project in MS Visual Studio.



2) Let's call our project a 'SilverlightApplication'



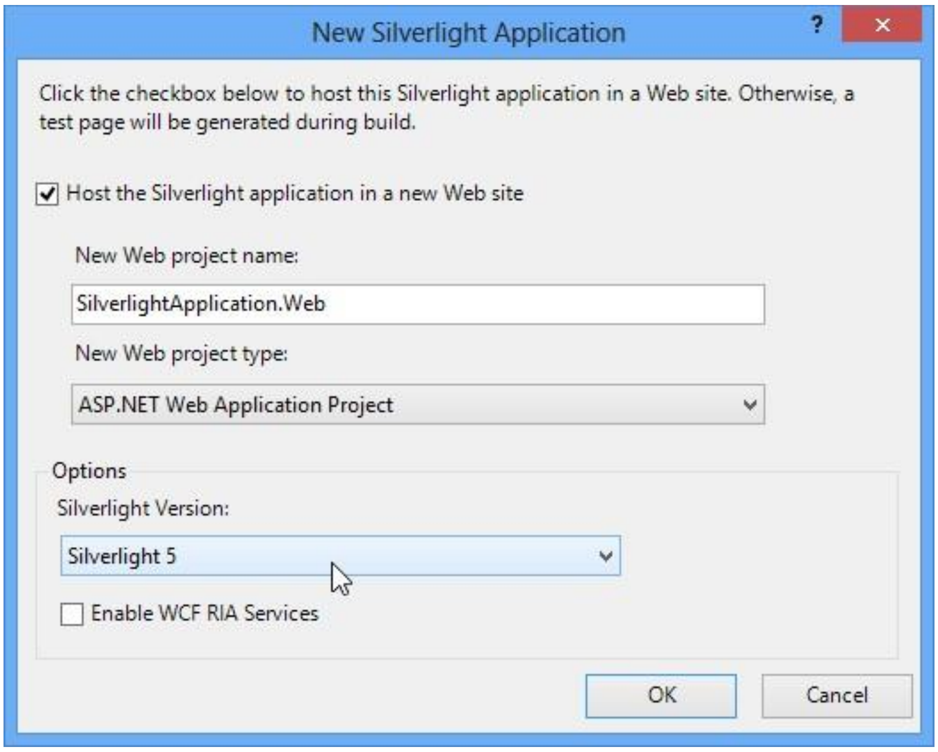
Recent
Installed
Templates
Visual C#
Windows Store
Windows
Web
Office
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Windows Phone
Workflow
LightSwitch
Other Languages
Other Project Types
Modeling Projects
Samples
Online

.NET Framework 4.5 Sort by: Default

Silverlight Application	Visual C#
Silverlight Class Library	Visual C#
Silverlight Business Application	Visual C#
Silverlight Navigation Application	Visual C#
WCF RIA Services Class Library	Visual C#

Name: SilverlightApplication
Location: C:\Getting Started\
Solution name: SilverlightApplication

3) While creating a project we should check 'Host the Silverlight application in a new Web site' option and set the name of the web-project to 'SilverlightApplication.Web'. Additionally, don't forget to specify 'Silverlight 5' version as a targeted platform.



New Silverlight Application

Click the checkbox below to host this Silverlight application in a Web site. Otherwise, a test page will be generated during build.

☑ Host the Silverlight application in a new Web site

New Web project name: SilverlightApplication.Web

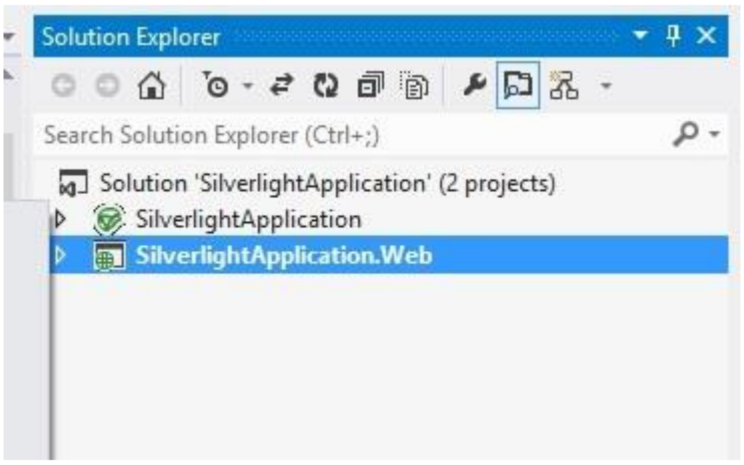
New Web project type: ASP.NET Web Application Project

Options

Silverlight Version: Silverlight 5

☐ Enable WCF RIA Services

4) A new Visual Studio solution will be created after that. It will contain two projects: client side 'SilverlightApplication' and server side 'SilverlightApplication.Web'.



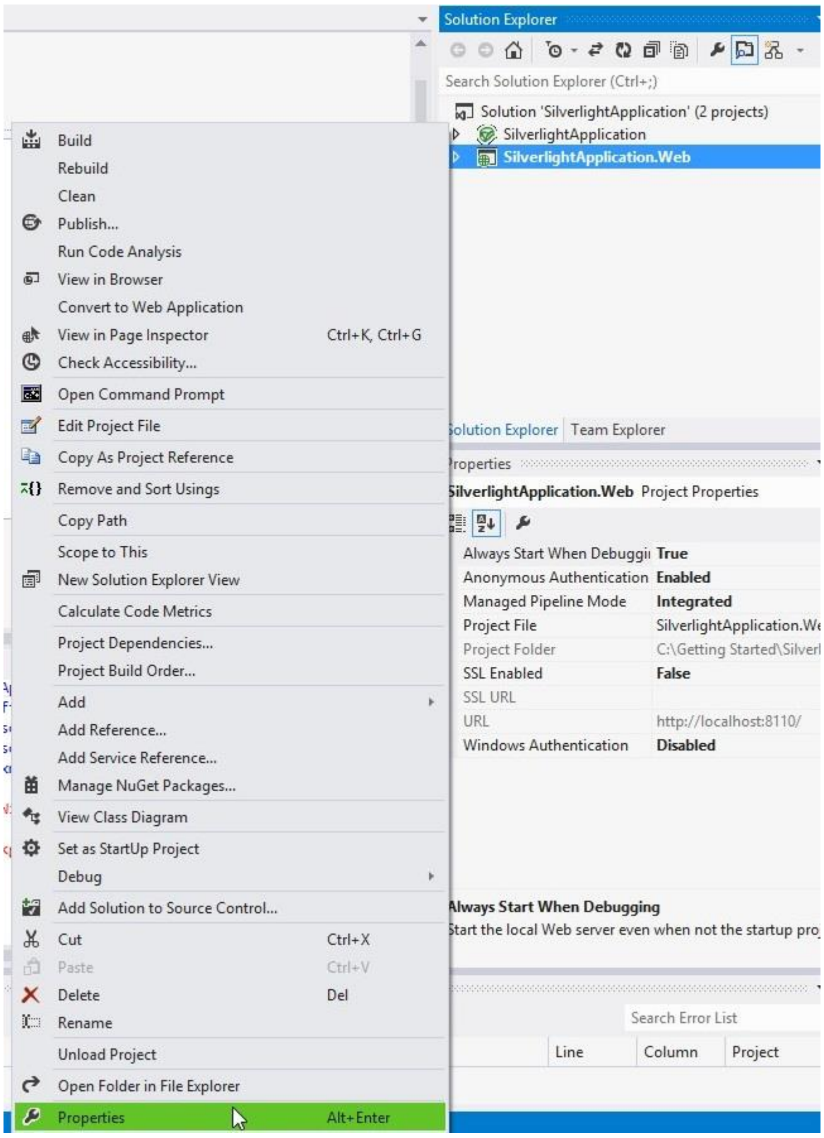
Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'SilverlightApplication' (2 projects)
SilverlightApplication
SilverlightApplication.Web

Step 2. Configuring Server Side Part

1) Set the web application to use static port 5555. To do this, open SilverlightApplication.Web properties

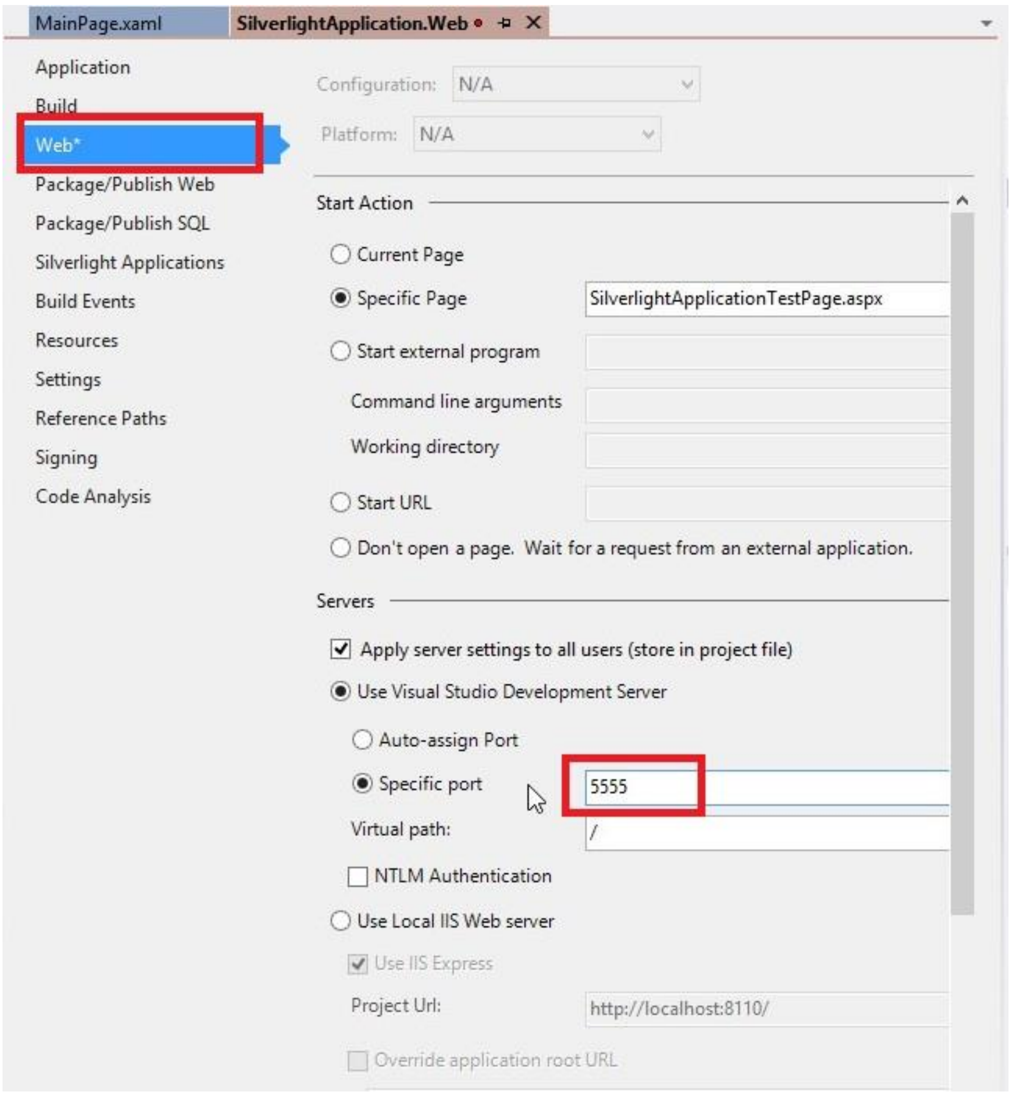


SilverlightApplication.Web Project Properties

Always Start When Debuggi	True
Anonymous Authentication	Enabled
Managed Pipeline Mode	Integrated
Project File	SilverlightApplication.We
Project Folder	C:\Getting Started\Silverl
SSL Enabled	False
SSL URL	
URL	http://localhost:8110/
Windows Authentication	Disabled

Always Start When Debugging
Start the local Web server even when not the startup pro

2) Check 'Specific port' option and set '5555' as a value in the Web tab of the SilverlightApplication.Web properties.



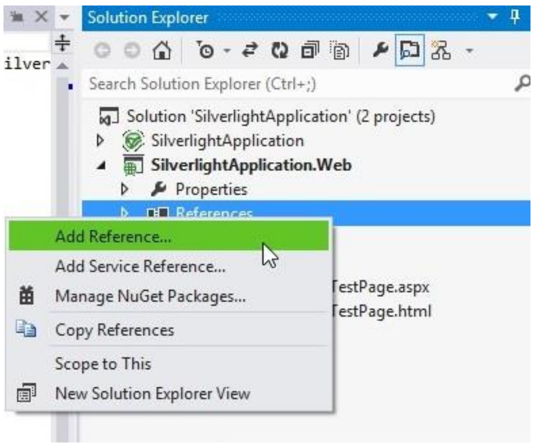
Configuration: N/A
Platform: N/A

Start Action
- Current Page
- [✓ Specific Page] SilverlightApplicationTestPage.aspx
- Start external program
- Command line arguments
- Working directory
- Start URL
- Don't open a page. Wait for a request from an external application.

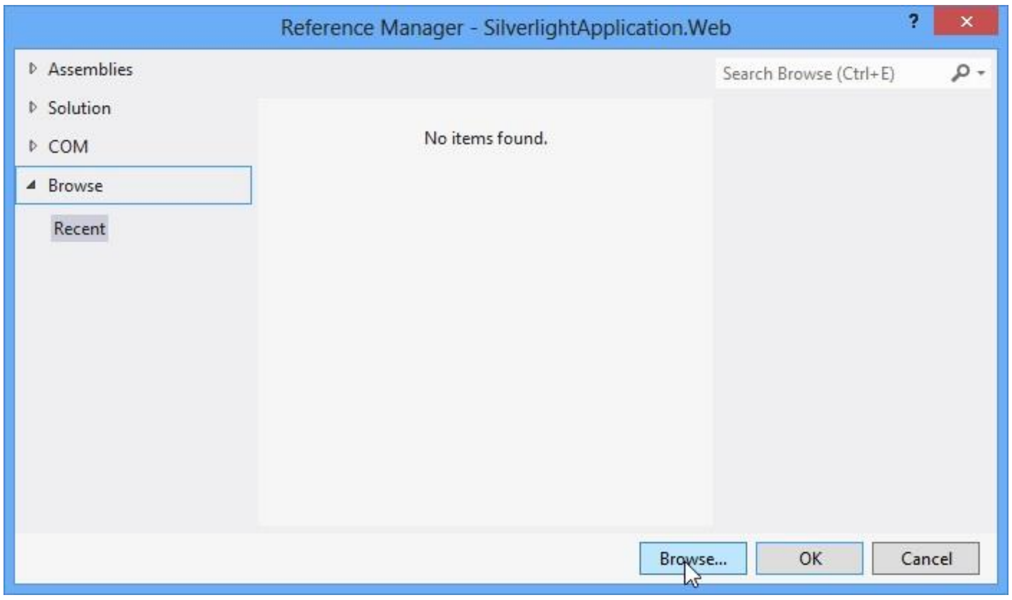
Servers
- [✓] Apply server settings to all users (store in project file)
- [✓ Use Visual Studio Development Server]
  - Auto-assign Port
  - [✓ Specific port] 5555
  - Virtual path: /
  - NTLM Authentication
- Use Local IIS Web server
  - [✓] Use IIS Express
  - Project Url: http://localhost:8110/
  - Override application root URL

Step 3. Adding and Setting Up a WCF Service

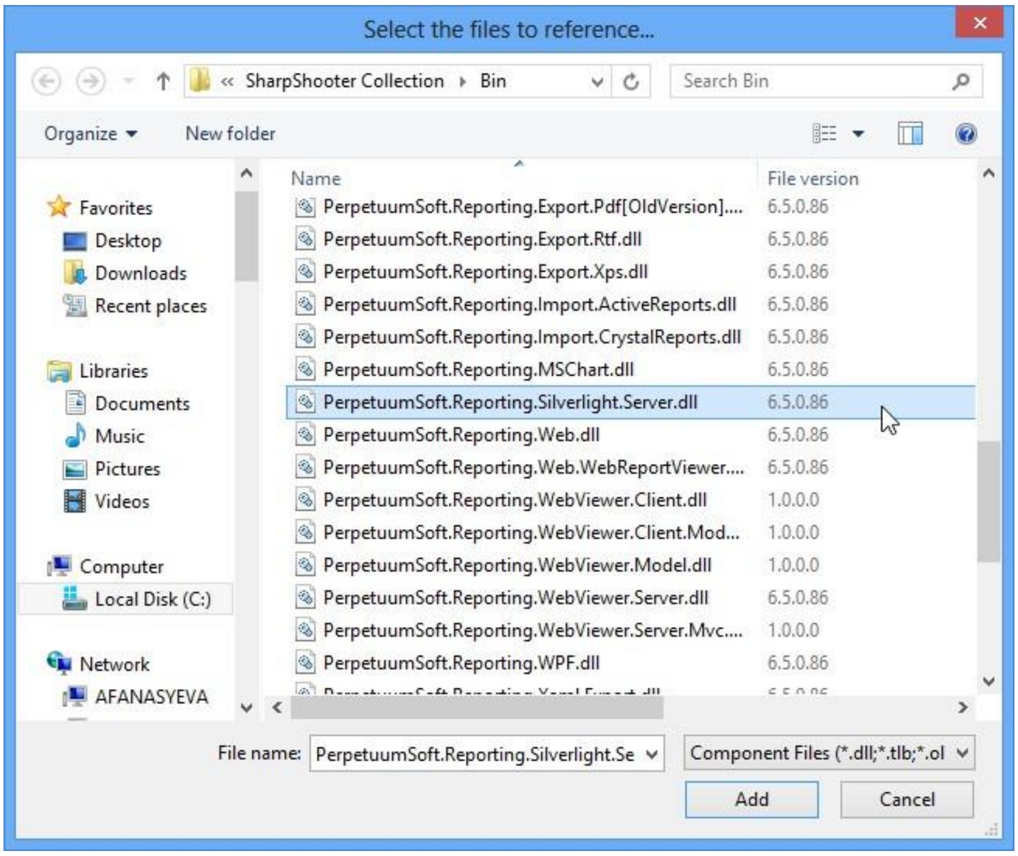
1) In order to make the client interact with the server, we need a special service. Before we create this service we need to add a reference to 'PerpetuumSoft.Reporting.Silverlight.Server.dll'. Right-click 'References' node in the SilverlightApplication.Web project and choose 'Add Reference' in the popup menu.



This will open a ‘Reference Manager’. In order to add a correct library version reference we should browse the \Bin folder of the installed product and find ‘PerpetuumSoft.Reporting.Silverlight.Server.dll’ in it.



The library version we are referencing to in our project is 6.5.0.86 as shown in the screenshot below.

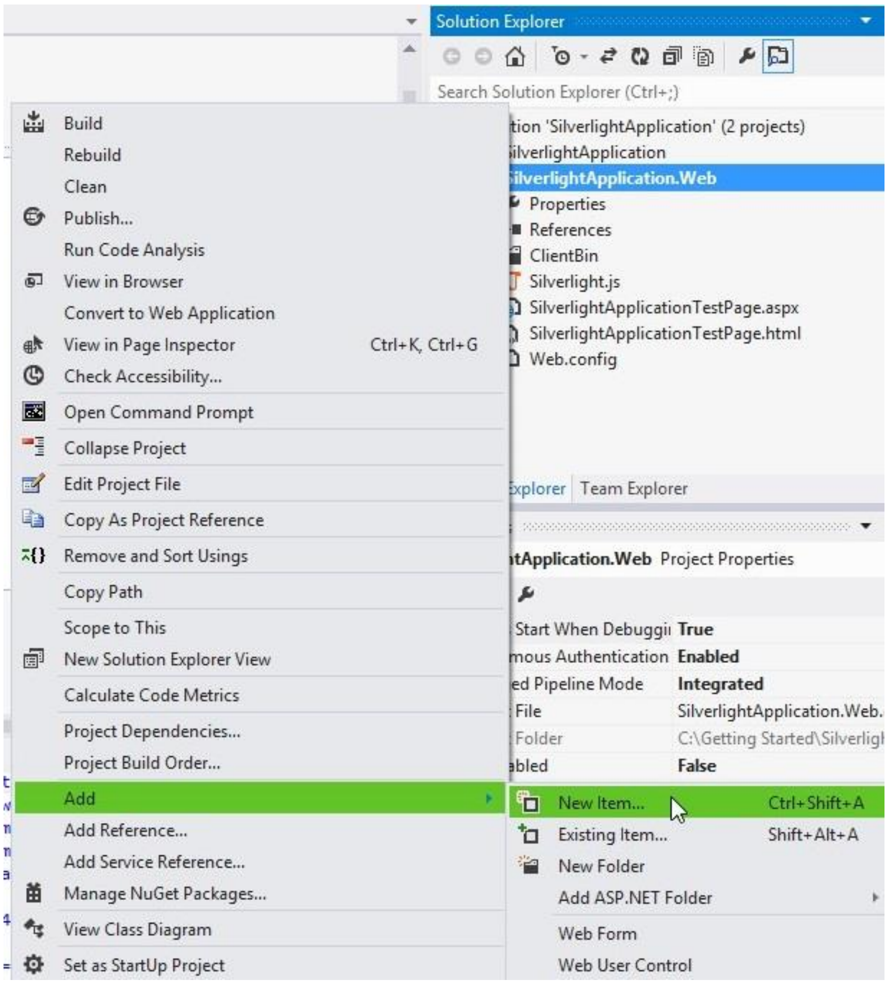


Name	File version
PerpetuumSoft.Reporting.Export.Pdf[OldVersion]....	6.5.0.86
PerpetuumSoft.Reporting.Export.Rtf.dll	6.5.0.86
PerpetuumSoft.Reporting.Export.Xps.dll	6.5.0.86
PerpetuumSoft.Reporting.Import.ActiveReports.dll	6.5.0.86
PerpetuumSoft.Reporting.Import.CrystalReports.dll	6.5.0.86
PerpetuumSoft.Reporting.MSChart.dll	6.5.0.86
PerpetuumSoft.Reporting.Silverlight.Server.dll	6.5.0.86
PerpetuumSoft.Reporting.Web.dll	6.5.0.86
PerpetuumSoft.Reporting.Web.WebReportViewer....	6.5.0.86
PerpetuumSoft.Reporting.WebViewer.Client.dll	1.0.0.0
PerpetuumSoft.Reporting.WebViewer.Client.Mod...	1.0.0.0
PerpetuumSoft.Reporting.WebViewer.Model.dll	1.0.0.0
PerpetuumSoft.Reporting.WebViewer.Server.dll	6.5.0.86
PerpetuumSoft.Reporting.WebViewer.Server.Mvc....	1.0.0.0
PerpetuumSoft.Reporting.WPF.dll	6.5.0.86

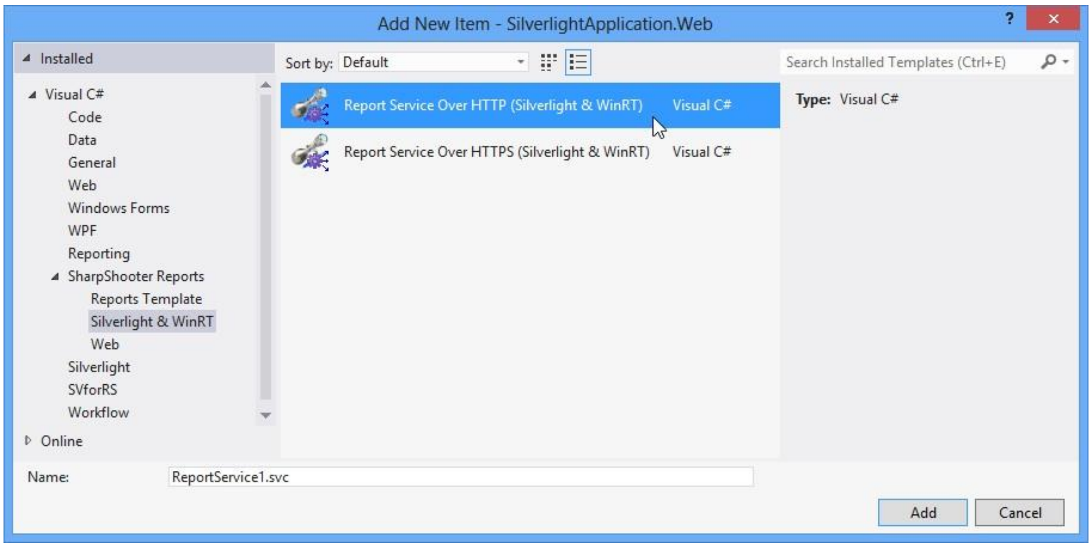
File name: PerpetuumSoft.Reporting.Silverlight.Se
Component Files (*.dll;*.tlb;*.ol

2) After that, we should add a Report Service Over HTTP template to the SilverlightApplication.Web project.

In order to do that, right-click SilverlightApplication.Web project and choose Add->New Item in the popup menu.



Select the Report Service Over HTTP (Silverlight & WinRT) template and specify name of the service as ‘ReportService1.svc’. Then click ‘Add’ button.

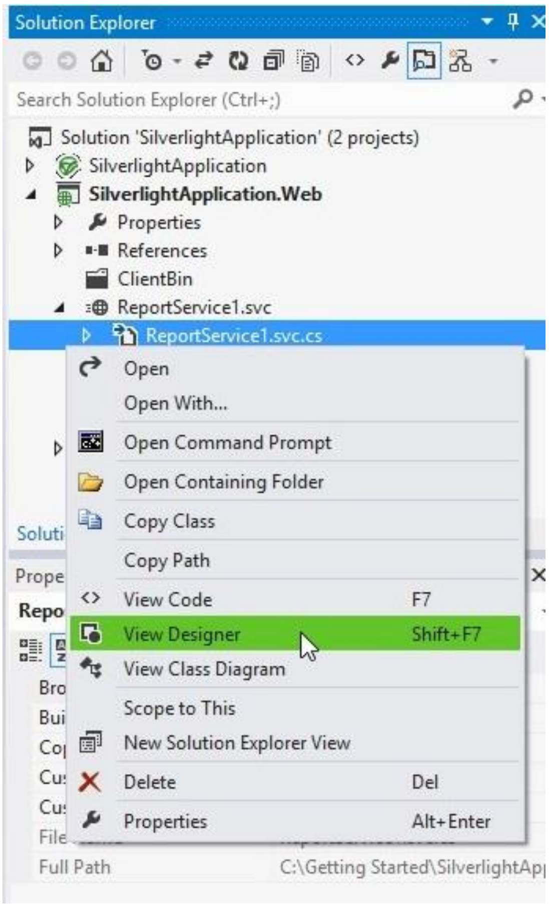


Name: ReportService1.svc

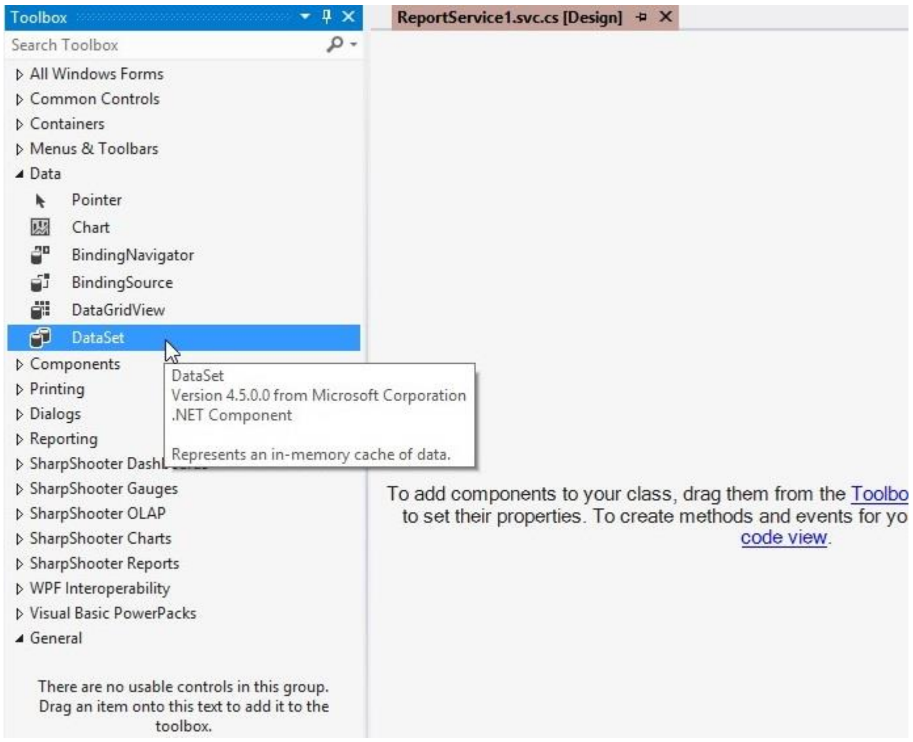
The ReportService class will extend the PerpetuumSoft.Reporting.Silverlight.Server.ReportServiceBase class containing implementation of WCF Service for Silverlight ReportViewer.

Step 4. Creating a Report Template

1) First of all we need to create a data source structure. In order to do that, right-click ReportService1.svc.cs and choose View Designer in the popup menu.



2) To populate our sample application with the data we add a Dataset control from the Toolbox by double click on it.



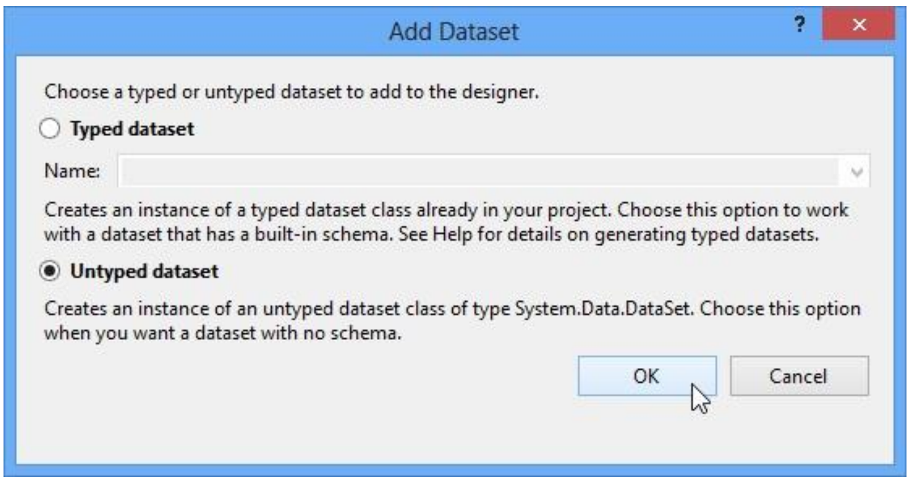
DataSet
Version 4.5.0.0 from Microsoft Corporation
.NET Component

Represents an in-memory cache of data.

There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

To add components to your class, drag them from the Toolbo
to set their properties. To create methods and events for yo
code view.

3) In the opened window we select the 'Untyped' dataset to simplify the process of adding data to our sample.



Add Dataset

Choose a typed or untyped dataset to add to the designer.

( ) **Typed dataset**

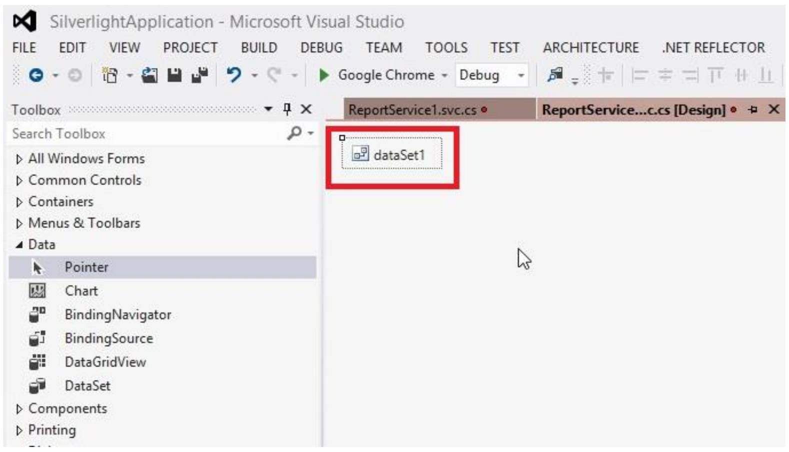
Name:

Creates an instance of a typed dataset class already in your project. Choose this option to work with a dataset that has a built-in schema. See Help for details on generating typed datasets.

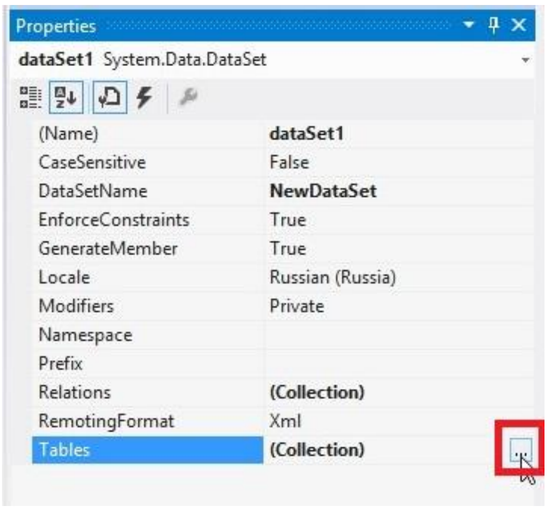
[✓ **Untyped dataset**]

Creates an instance of an untyped dataset class of type System.Data.DataSet. Choose this option when you want a dataset with no schema.

After that the dataset node (dataSet1) will appear in the designer.

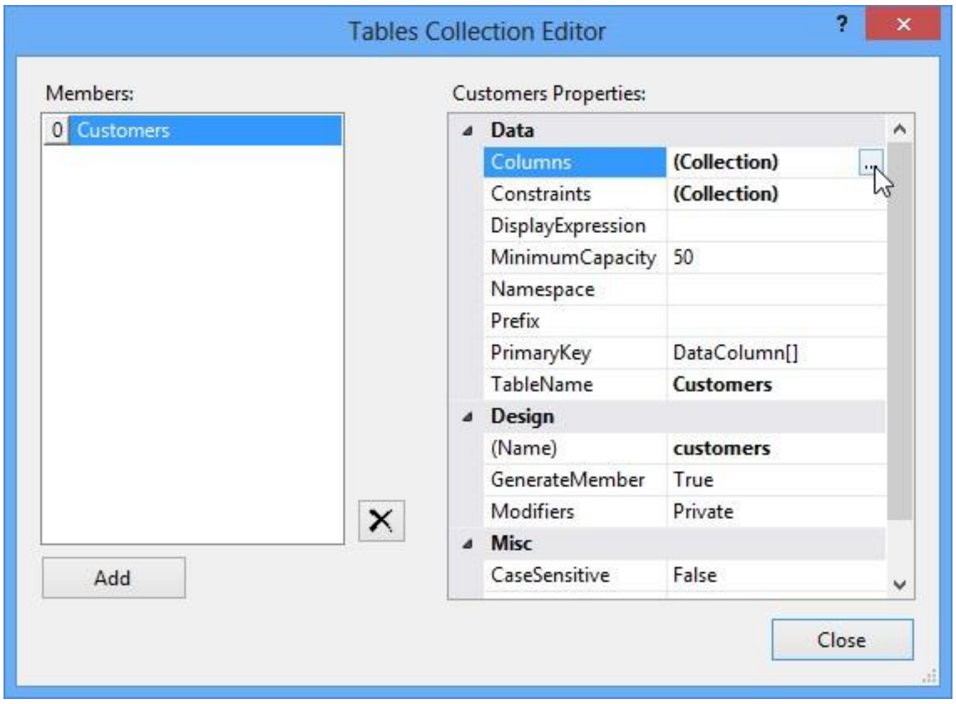


4) Next, open a Tables Collection Editor (by clicking … button on the right of the Tables property).



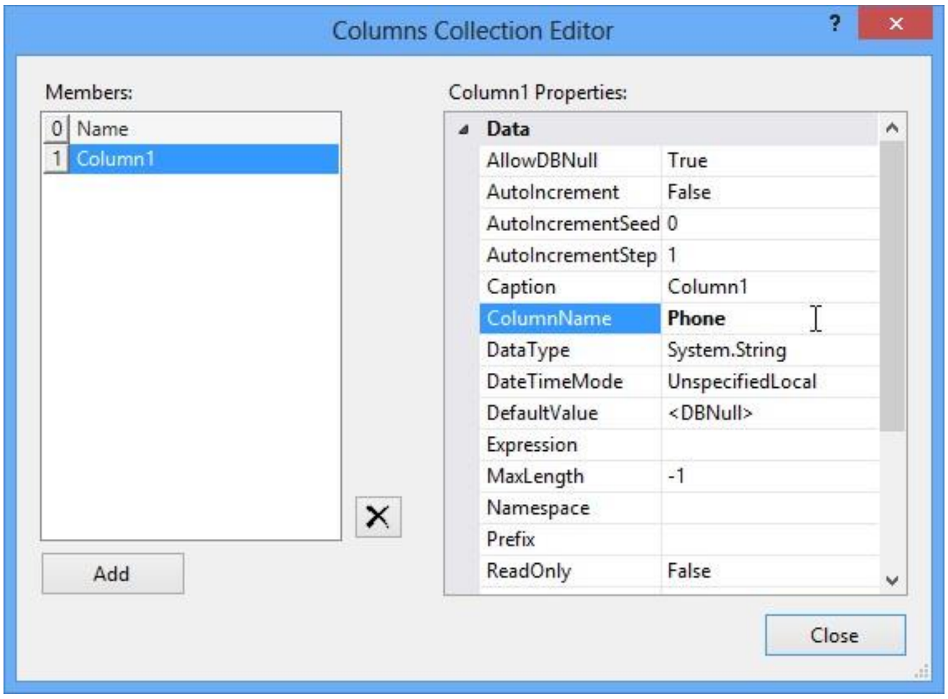
(Name)	dataSet1
CaseSensitive	False
DataSetName	NewDataSet
EnforceConstraints	True
GenerateMember	True
Locale	Russian (Russia)
Modifiers	Private
Namespace	
Prefix	
Relations	(Collection)
RemotingFormat	Xml
Tables	(Collection)

5) Add ‘Customers’ table to the dataSet1 (click “Add” button and set ‘TableName’ property value to ‘Customers’ and ‘Name’ property value to ‘customers’).



Data	
Columns	(Collection)
Constraints	(Collection)
DisplayExpression	
MinimumCapacity	50
Namespace	
Prefix	
PrimaryKey	DataColumn[]
TableName	Customers
Design	
(Name)	customers
GenerateMember	True
Modifiers	Private
Misc	
CaseSensitive	False

6) After that, open a Columns Collection Editor (by clicking … button on the right of the Columns property) and add two columns by clicking ‘Add’ button and setting ‘ColumnName’ property value to ‘Name’ and ‘Phone’.



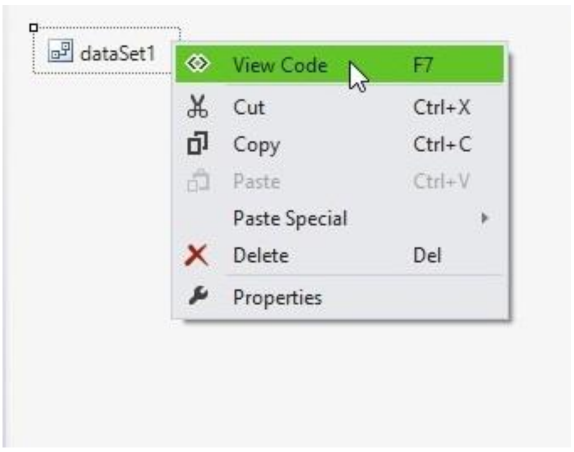
Members:

0	Name
1	Column1

Column1 Properties:

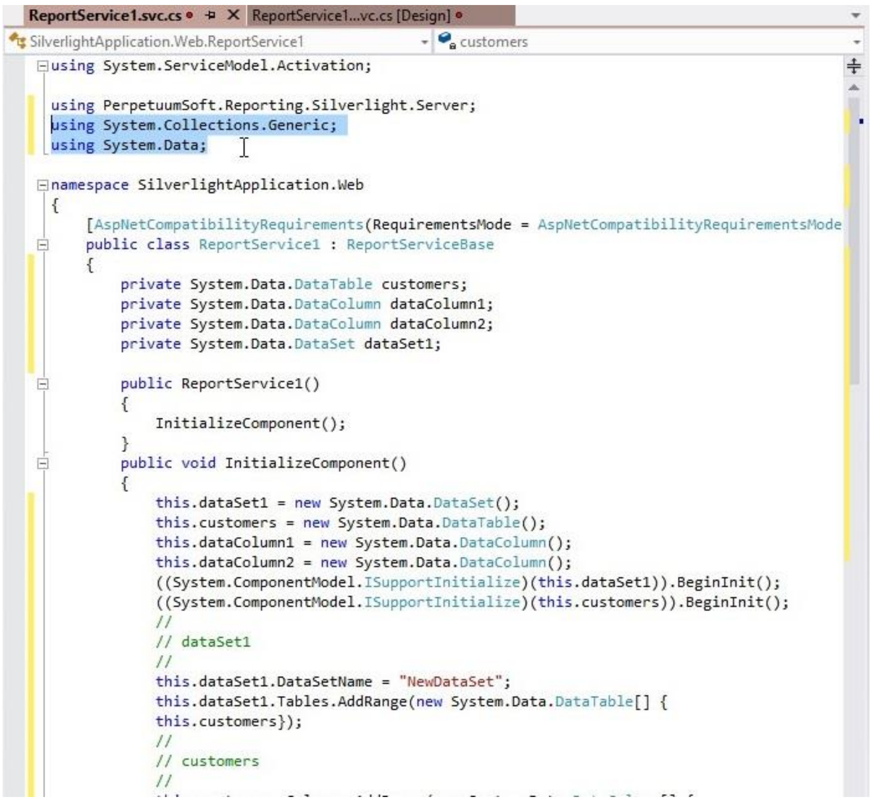
Data	
AllowDBNull	True
AutoIncrement	False
AutoIncrementSeed	0
AutoIncrementStep	1
Caption	Column1
ColumnName	**Phone**
DataType	System.String
DateTimeMode	UnspecifiedLocal
DefaultValue	<DBNull>
Expression	
MaxLength	-1
Namespace	
Prefix	
ReadOnly	False

7) Now we have data structure defined and it’s necessary to fill our sample ‘Customers’ table with the data itself. Since the data of our sample table will be populated programmatically we need to switch to source code view by right clicking the designer area and selecting ‘View Code’ item in the contextual menu.



8) Add the following namespaces to your code

System.Collections.Generic
System.Data

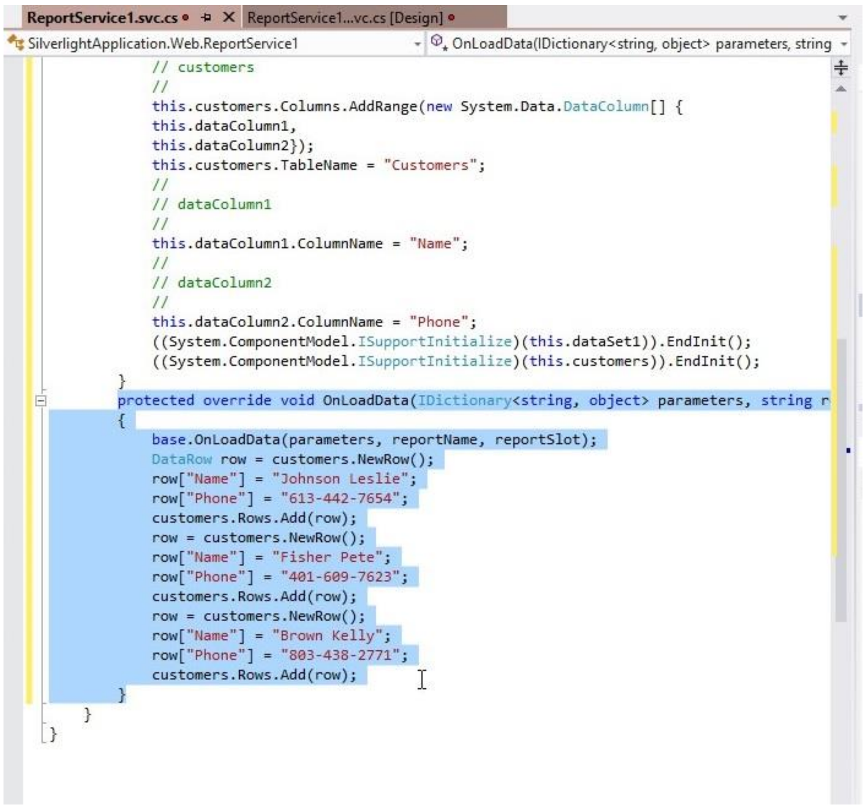


ReportService1.svc.cs
ReportService1...vc.cs [Design]
SilverlightApplication.Web.ReportService1
customers

```
using System.ServiceModel.Activation;

using PerpetuumSoft.Reporting.Silverlight.Server;
using System.Collections.Generic;
using System.Data;

namespace SilverlightApplication.Web
{
    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode
    public class ReportService1 : ReportServiceBase
    {
        private System.Data.DataTable customers;
        private System.Data.DataColumn dataColumn1;
        private System.Data.DataColumn dataColumn2;
        private System.Data.DataSet dataSet1;

        public ReportService1()
        {
            InitializeComponent();
        }
        public void InitializeComponent()
        {
            this.dataSet1 = new System.Data.DataSet();
            this.customers = new System.Data.DataTable();
            this.dataColumn1 = new System.Data.DataColumn();
            this.dataColumn2 = new System.Data.DataColumn();
            ((System.ComponentModel.ISupportInitialize)(this.dataSet1)).BeginInit();
            ((System.ComponentModel.ISupportInitialize)(this.customers)).BeginInit();
            //
            // dataSet1
            //
            this.dataSet1.DataSetName = "NewDataSet";
            this.dataSet1.Tables.AddRange(new System.Data.DataTable[] {
            this.customers});
            //
            // customers
            //
```

9) Now we will fill in the values via overriding OnLoadData method of the ReportService class.

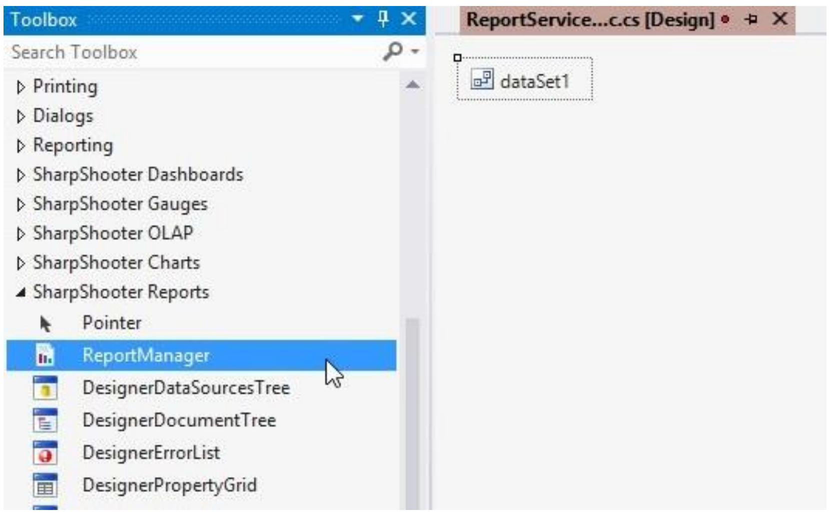
```
protected override void OnLoadData(IDictionary<string, object> parameters, string reportName,
PerpetuumSoft.Reporting.Components.ReportSlot reportSlot)
{
    base.OnLoadData(parameters, reportName, reportSlot);
    DataRow row = customers.NewRow();
    row["Name"] = "Johnson Leslie";
    row["Phone"] = "613-442-7654";
    customers.Rows.Add(row);
    row = customers.NewRow();
    row["Name"] = "Fisher Pete";
    row["Phone"] = "401-609-7623";
    customers.Rows.Add(row);
    row = customers.NewRow();
    row["Name"] = "Brown Kelly";
    row["Phone"] = "803-438-2771";
    customers.Rows.Add(row);
}
```


ReportService1.svc.cs
ReportService1...vc.cs [Design]
SilverlightApplication.Web.ReportService1
OnLoadData(IDictionary<string, object> parameters, string

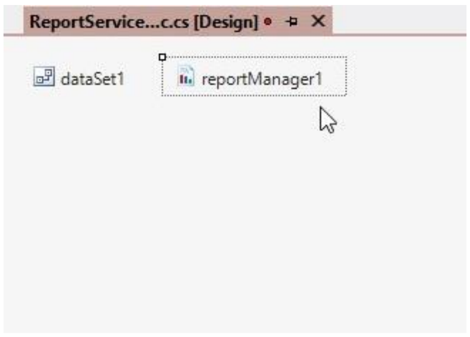
```
            // customers
            //
            this.customers.Columns.AddRange(new System.Data.DataColumn[] {
            this.dataColumn1,
            this.dataColumn2});
            this.customers.TableName = "Customers";
            //
            // dataColumn1
            //
            this.dataColumn1.ColumnName = "Name";
            //
            // dataColumn2
            //
            this.dataColumn2.ColumnName = "Phone";
            ((System.ComponentModel.ISupportInitialize)(this.dataSet1)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.customers)).EndInit();
        }
        protected override void OnLoadData(IDictionary<string, object> parameters, string r
        {
            base.OnLoadData(parameters, reportName, reportSlot);
            DataRow row = customers.NewRow();
            row["Name"] = "Johnson Leslie";
            row["Phone"] = "613-442-7654";
            customers.Rows.Add(row);
            row = customers.NewRow();
            row["Name"] = "Fisher Pete";
            row["Phone"] = "401-609-7623";
            customers.Rows.Add(row);
            row = customers.NewRow();
            row["Name"] = "Brown Kelly";
            row["Phone"] = "803-438-2771";
            customers.Rows.Add(row);
        }
    }
}
```

Note: Information about the queried document and its parameters is passed in the method's parameters.

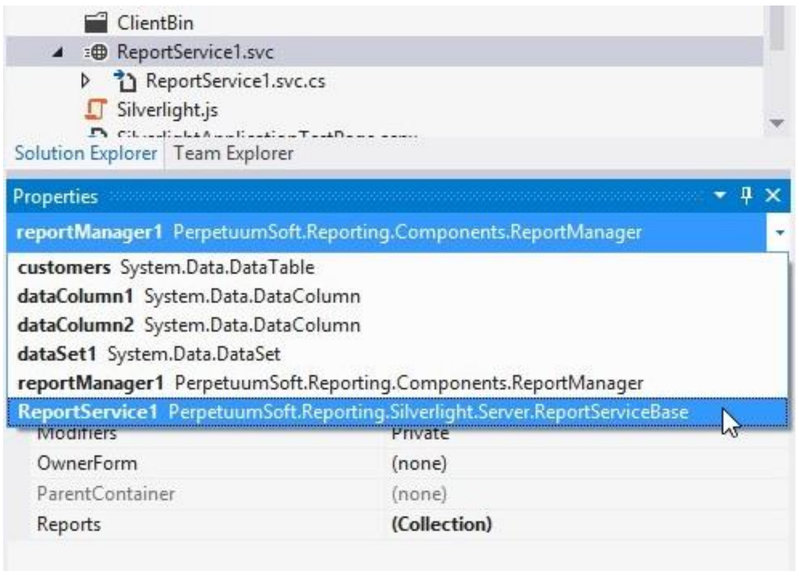
10) Now add the ‘ReportManager’ component (by double clicking ‘ReportManager’ on the ToolBox). This component is responsible for report generation.



After that the ReportManager node (reportManager1) will appear in the designer.



11) Set the ReportManager property of the ReportService service. In order to do that, open ReportService properties in the Properties window.

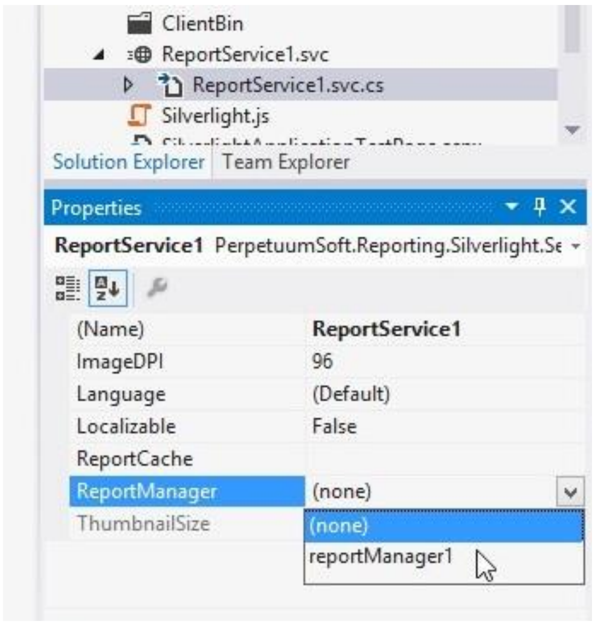


reportManager1 PerpetuumSoft.Reporting.Components.ReportManager

customers	System.Data.DataTable
dataColumn1	System.Data.DataColumn
dataColumn2	System.Data.DataColumn
dataSet1	System.Data.DataSet
reportManager1	PerpetuumSoft.Reporting.Components.ReportManager
ReportService1	PerpetuumSoft.Reporting.Silverlight.Server.ReportServiceBase

Modifiers	Private
OwnerForm	(none)
ParentContainer	(none)
Reports	(Collection)

And select reportManager1 from the list.



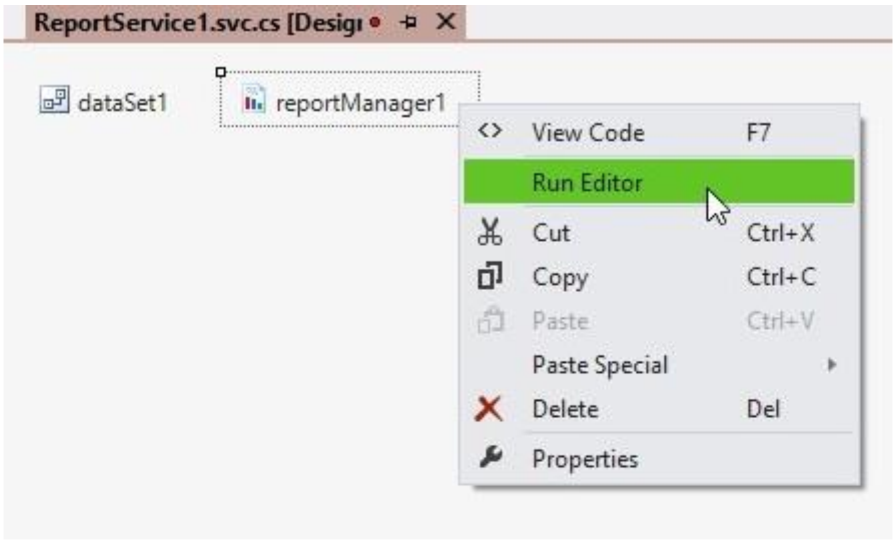
ReportService1 PerpetuumSoft.Reporting.Silverlight.Se

(Name)	ReportService1
ImageDPI	96
Language	(Default)
Localizable	False
ReportCache	
ReportManager	(none)
ThumbnailSize	

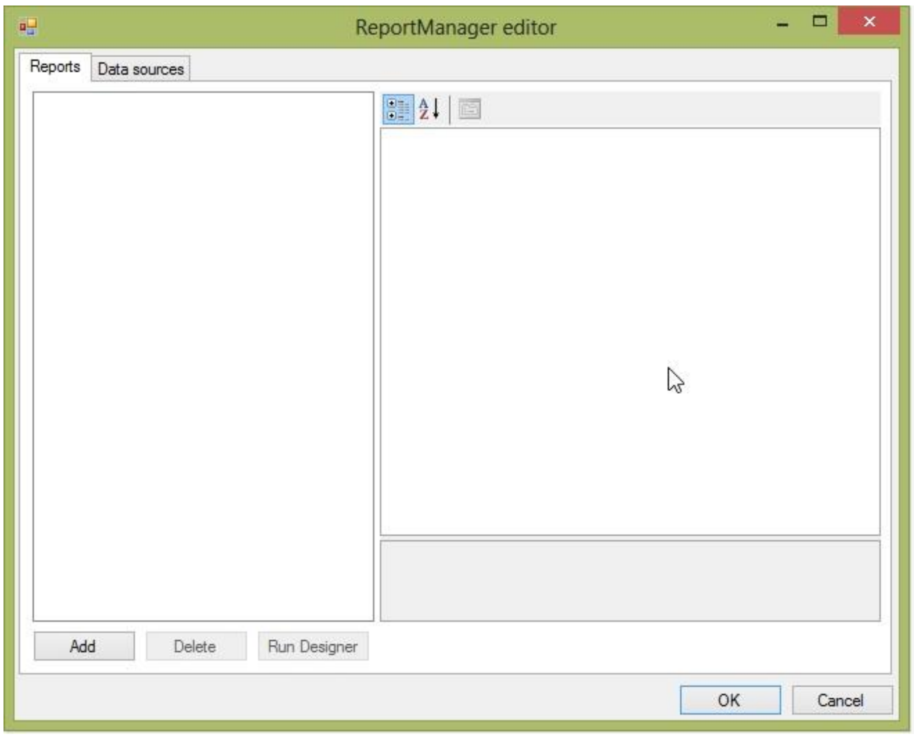
(none)
reportManager1

Step 5. Creating a Report

1) Now we will create a report. To do this, run Report Manager Editor by right clicking reportManager1 and choosing Run Editor option.

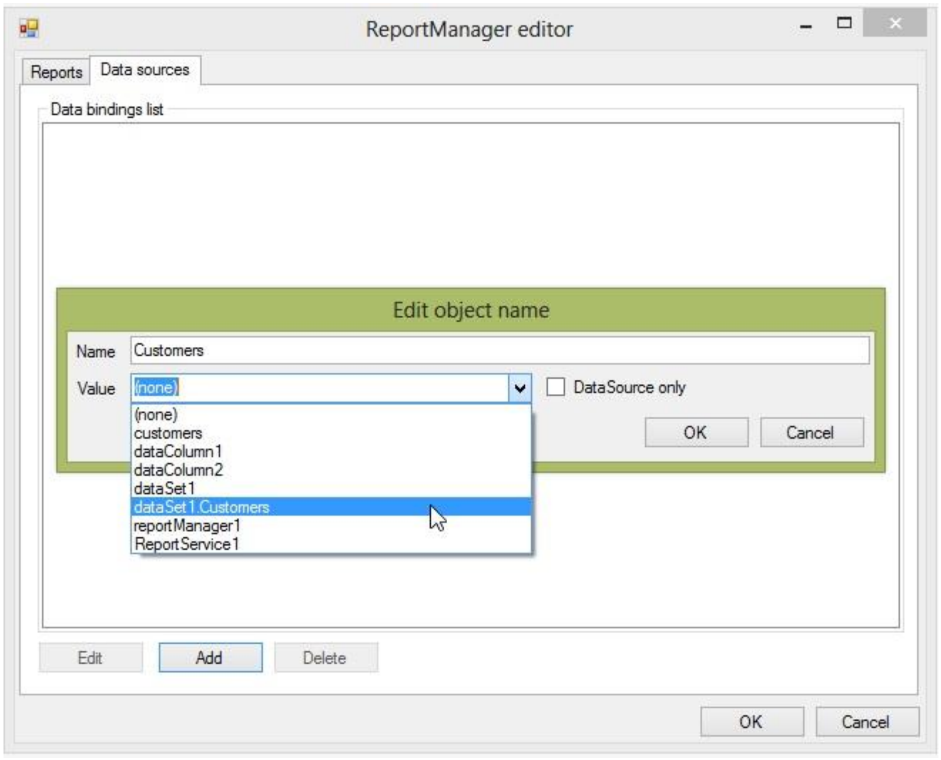


A Report Manager editor will appear.

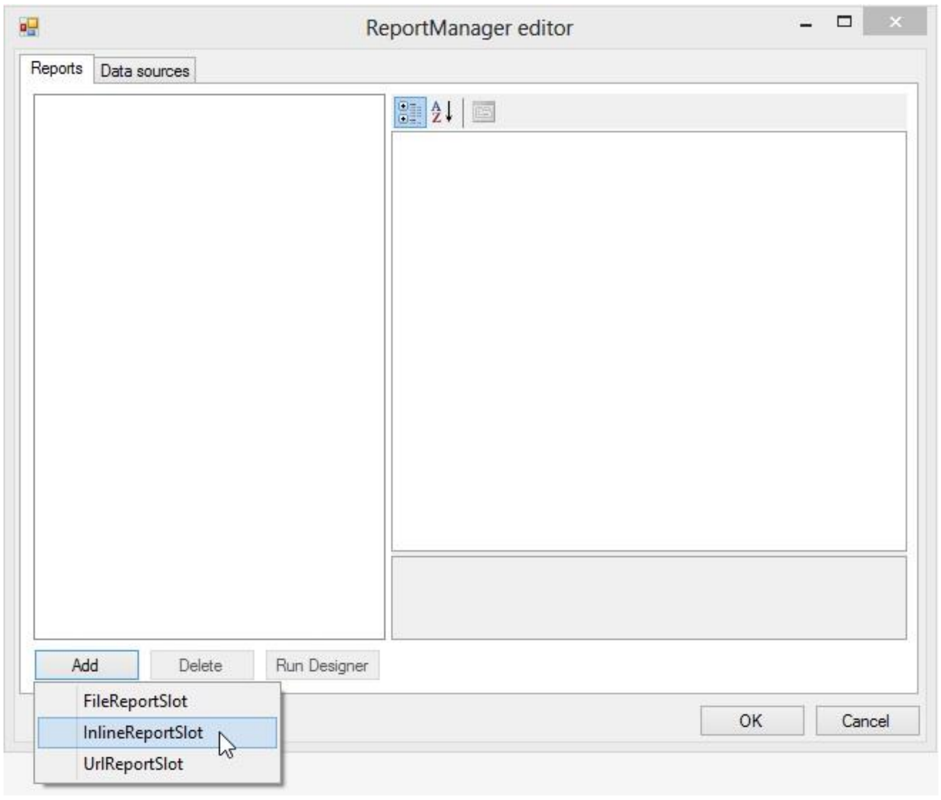


Before we start creating a report template we need to add a data source from which the report will be generated.

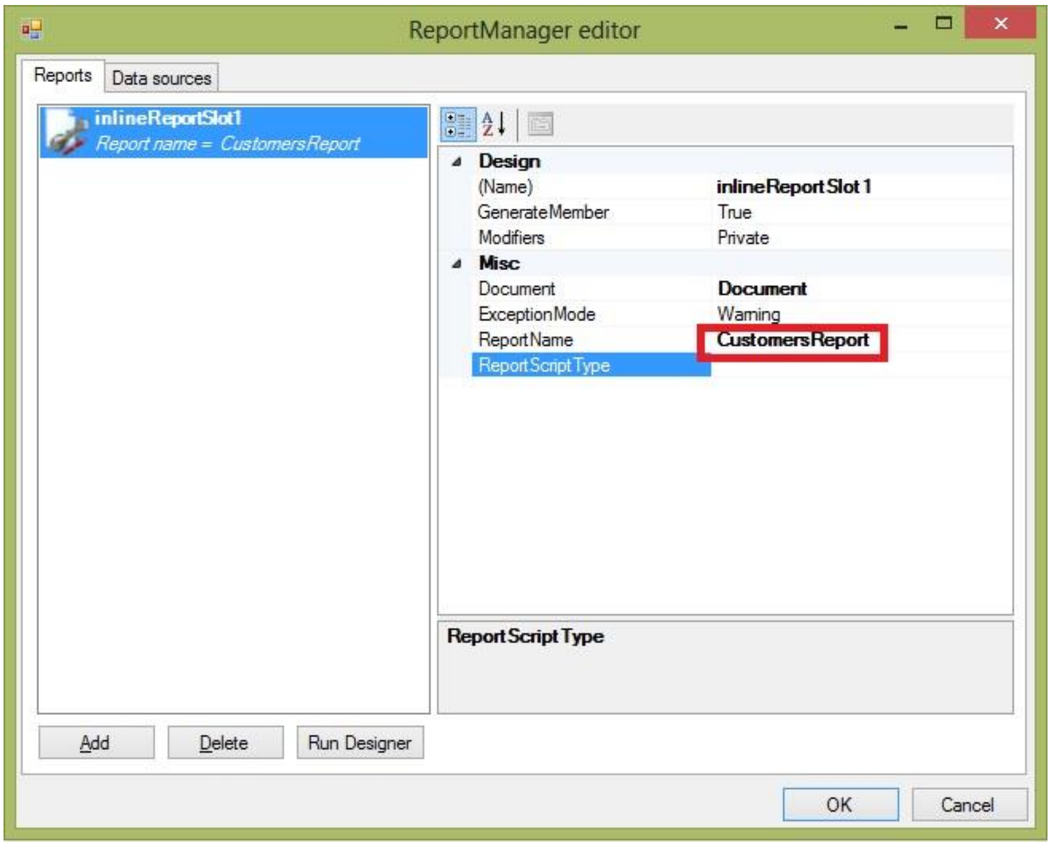
2) To do this, we need to add Customers table to the Data binding list located on the “Data sources” tab. Click “Add” button, specify ’Name’ value as ‘Customers’ in the appeared “Edit object name” form and select ‘dataSet1.Customers’ from the combo box list.



In the “Reports” tab, add a new object – “InlineReportSlot” by clicking the “Add” button.



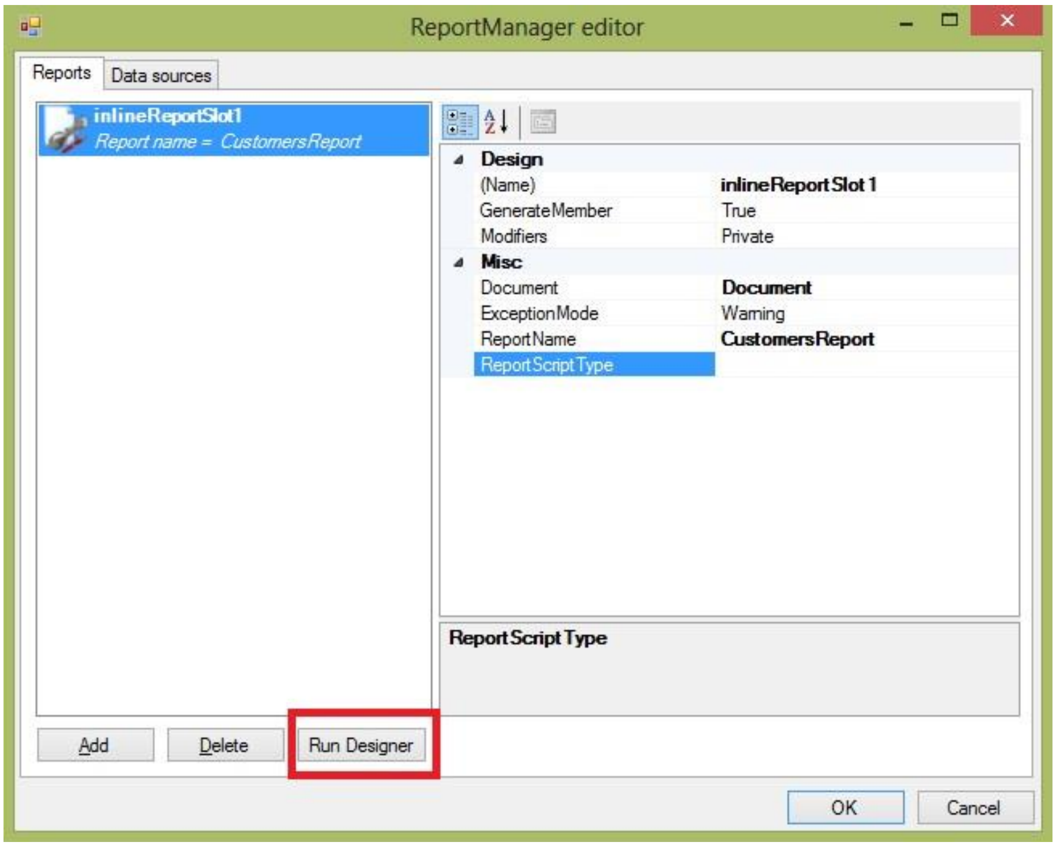
Set the ReportName property value to CustomersReport. *(Note: you will get the required document from the Report Manager exactly by that name*).



Design
(Name) inlineReportSlot1
GenerateMember True
Modifiers Private
Misc
Document Document
ExceptionMode Warning
ReportName CustomersReport
ReportScriptType

ReportScriptType

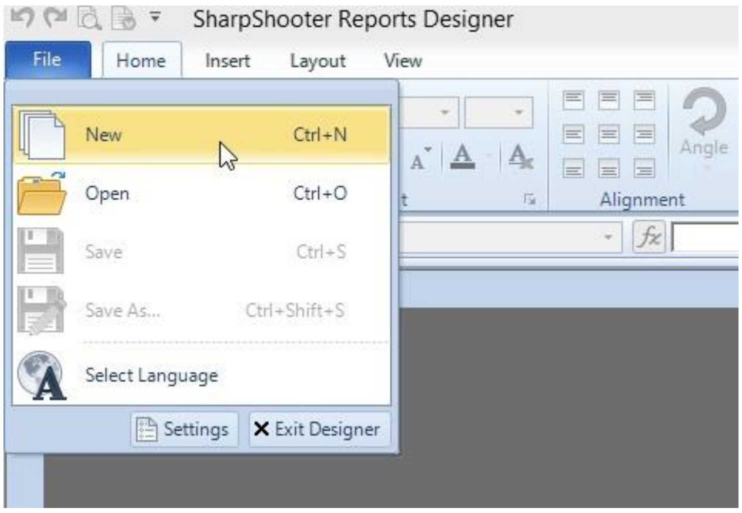
Then press the “Run Designer” button to launch Report Designer.



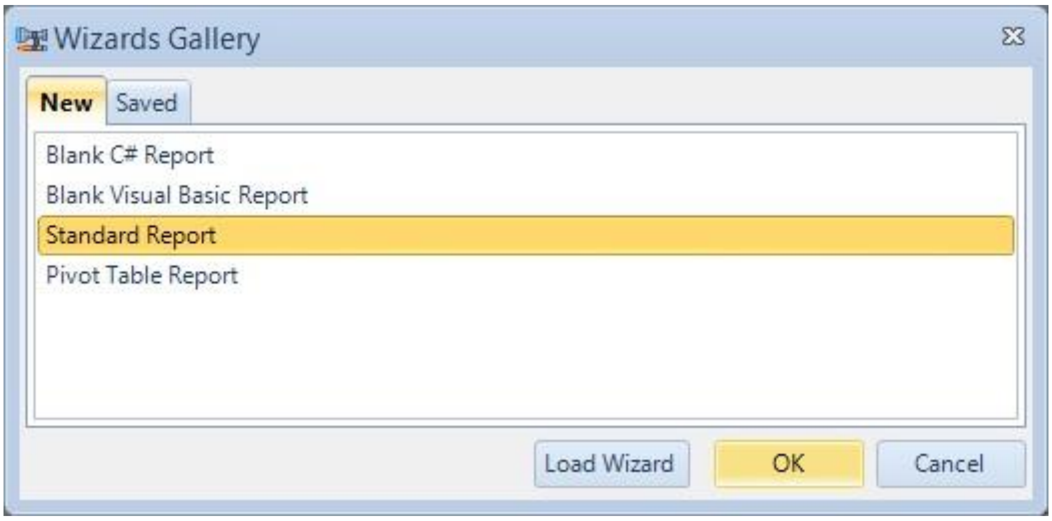
Design
(Name) inlineReportSlot1
GenerateMember True
Modifiers Private
Misc
Document Document
ExceptionMode Warning
ReportName CustomersReport
ReportScriptType

ReportScriptType

In the opened Report Designer window select the File\New menu item, and the form shown on the screen below will appear.

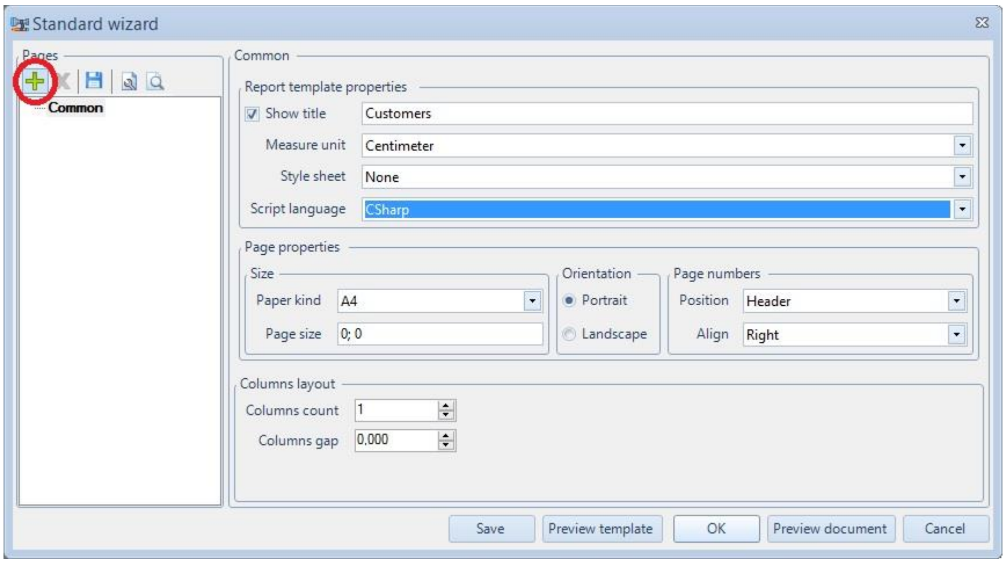


In the opened Wizards Gallery select Standard Report in the "New" tab and press OK.



The Standard Wizard window will appear on the screen.

Set document parameters as shown in the following picture and add a data source using the "Add" button ( ).


Standard wizard

Pages

Common

Common

Report template properties

Show title: Customers

Measure unit: Centimeter

Style sheet: None

Script language: CSharp

Page properties

Size

Paper kind: A4

Page size: 0; 0

Orientation

Portrait

Landscape

Page numbers

Position: Header

Align: Right

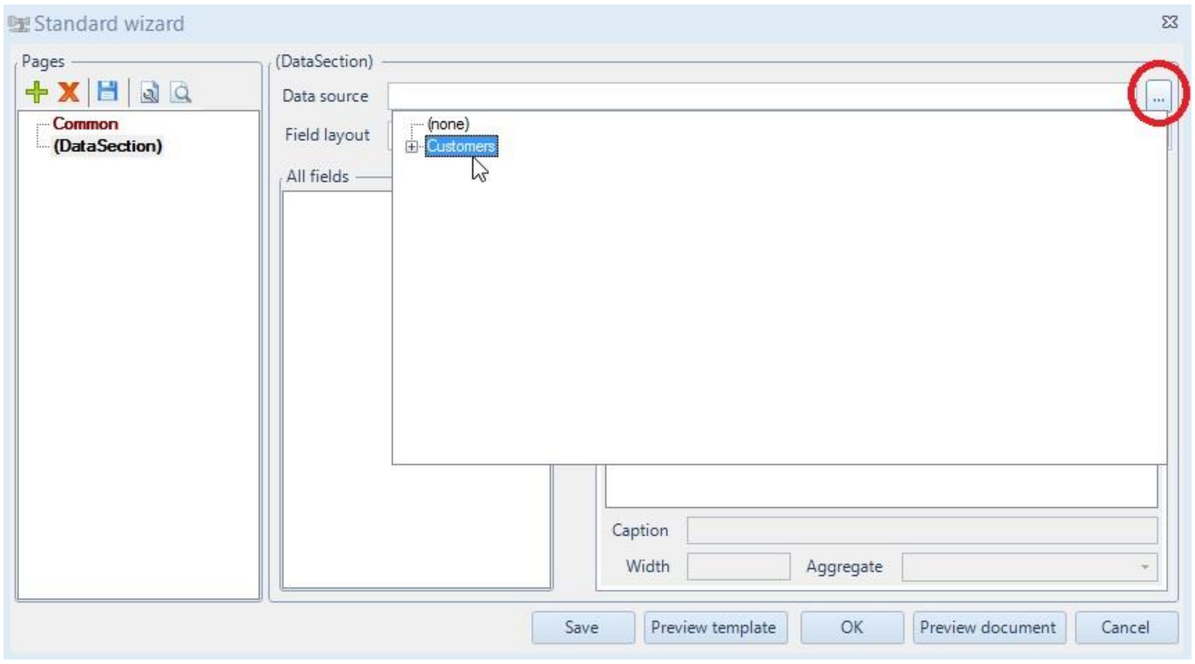
Columns layout

Columns count: 1

Columns gap: 0,000

Save | Preview template | OK | Preview document | Cancel

Click button, select “Customers” by double click in the appeared tree view.


Standard wizard

Pages

Common

(DataSection)

(DataSection)

Data source

Field layout

All fields

(none)

Customers

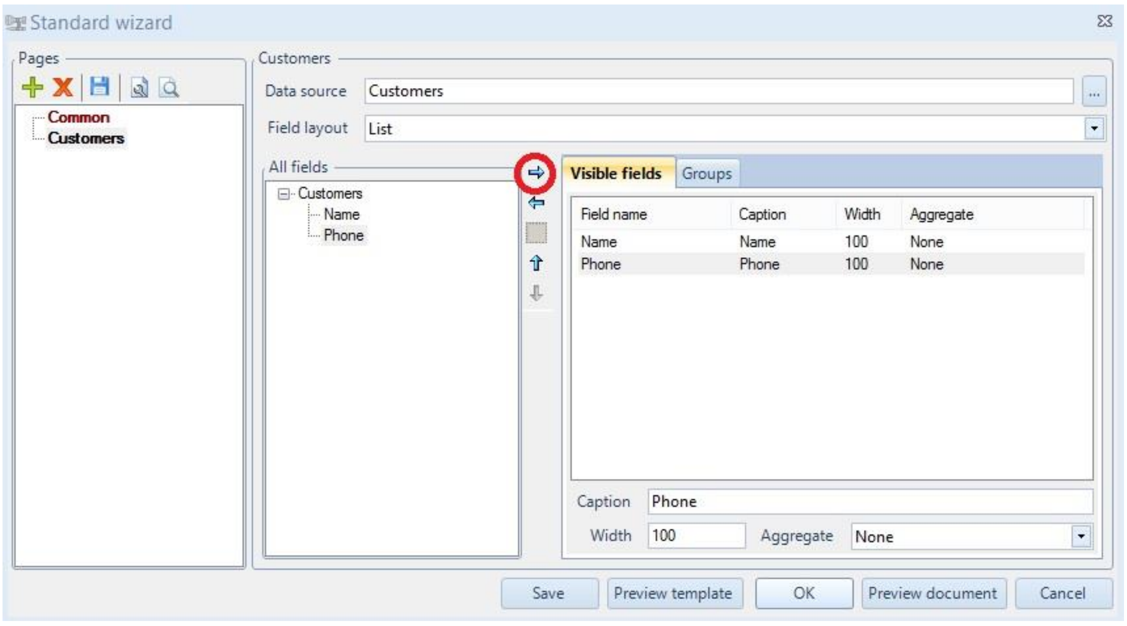
Caption

Width

Aggregate

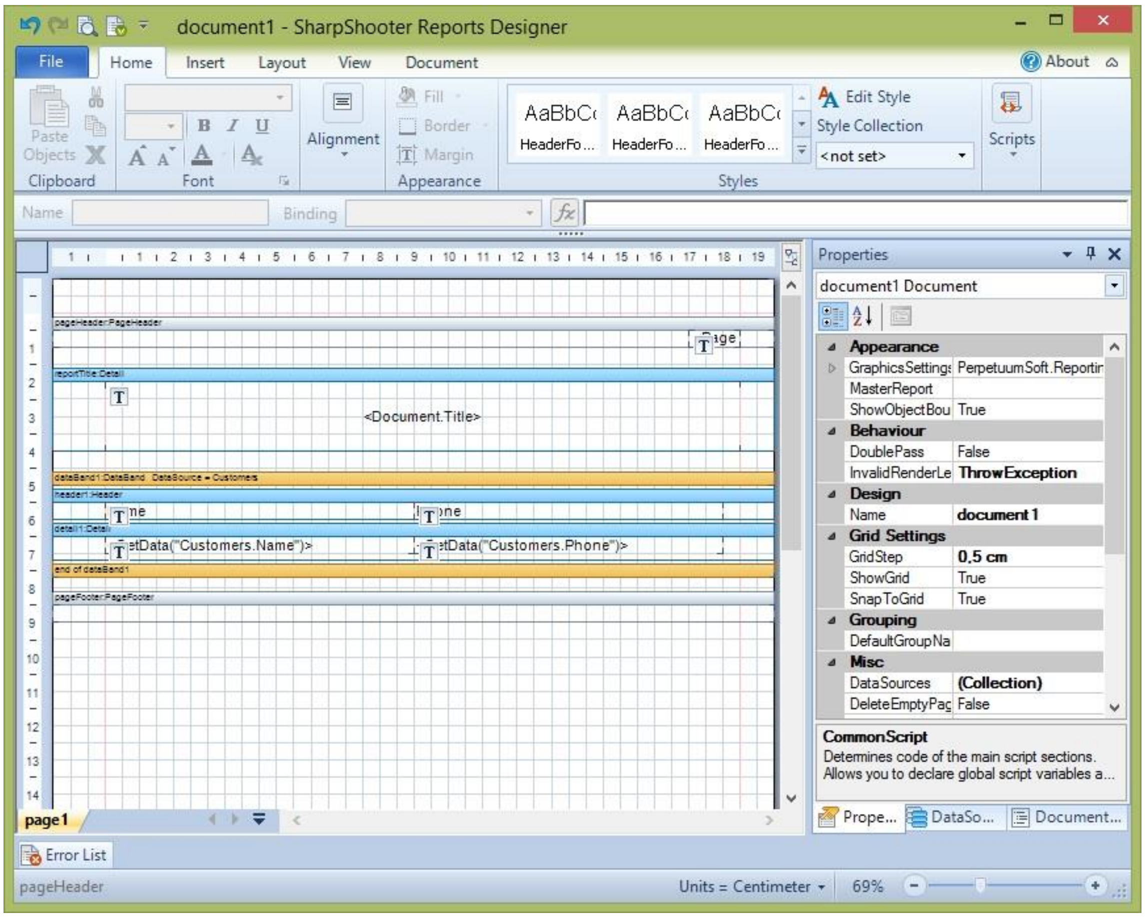
Save | Preview template | OK | Preview document | Cancel

Select the fields you want to output in the report (for example, both fields Name and Phone):



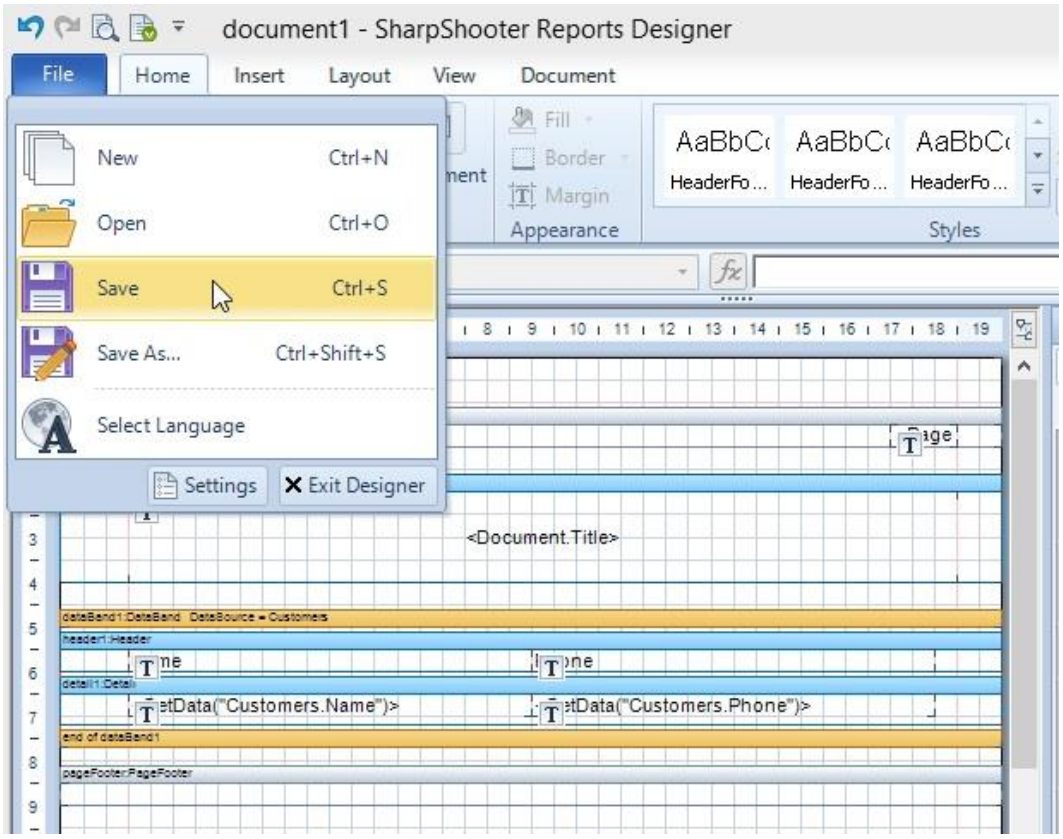
Field name	Caption	Width	Aggregate
Name	Name	100	None
Phone	Phone	100	None

Then press “OK” to create a template.



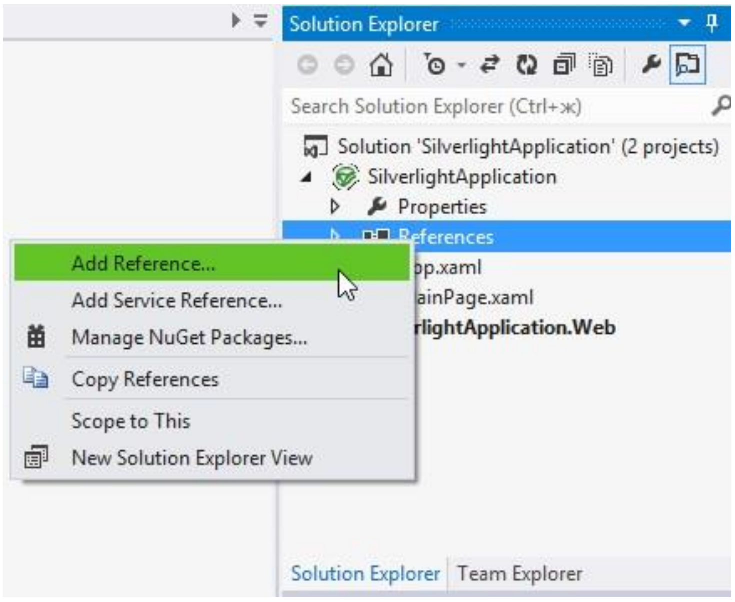
Appearance	
GraphicsSettings	PerpetuumSoft.Reportin
MasterReport	
ShowObjectBou	True
Behaviour	
DoublePass	False
InvalidRenderLe	ThrowException
Design	
Name	document1
Grid Settings	
GridStep	0,5 cm
ShowGrid	True
SnapToGrid	True
Grouping	
DefaultGroupNa	
Misc	
DataSources	(Collection)
DeleteEmptyPag	False

Now save the template and close the Designer.

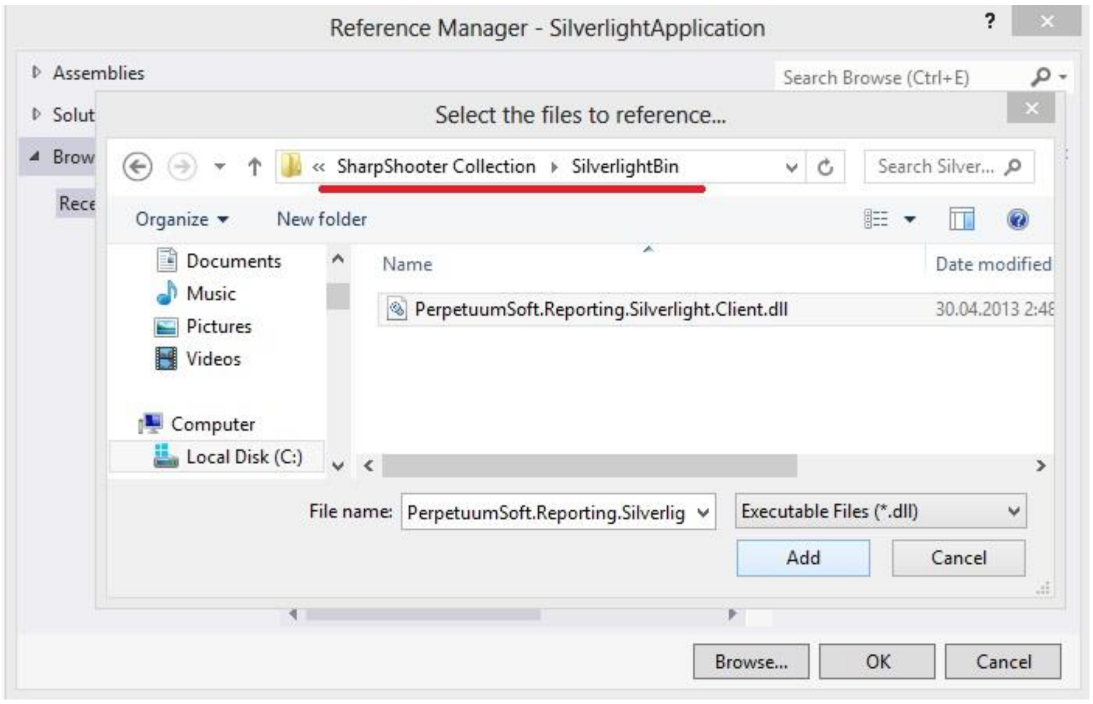


Step 6. Adding and Setting Up a Silverlight ReportViewer component.

1) Now it’s time to add a Report Viewer component to our Silverlight application to be able to display a report. In order to do that, we should add a reference to PerpetuumSoft.Reporting.Silverlight.Client assembly which contains ReportViewer. Right click ‘References’ of the SilverlightApplication project in the ‘Solution Explorer’ and choose’ Add Reference’ item in the popup menu.



The library is located in the \SilverlightBin folder of the installed product.

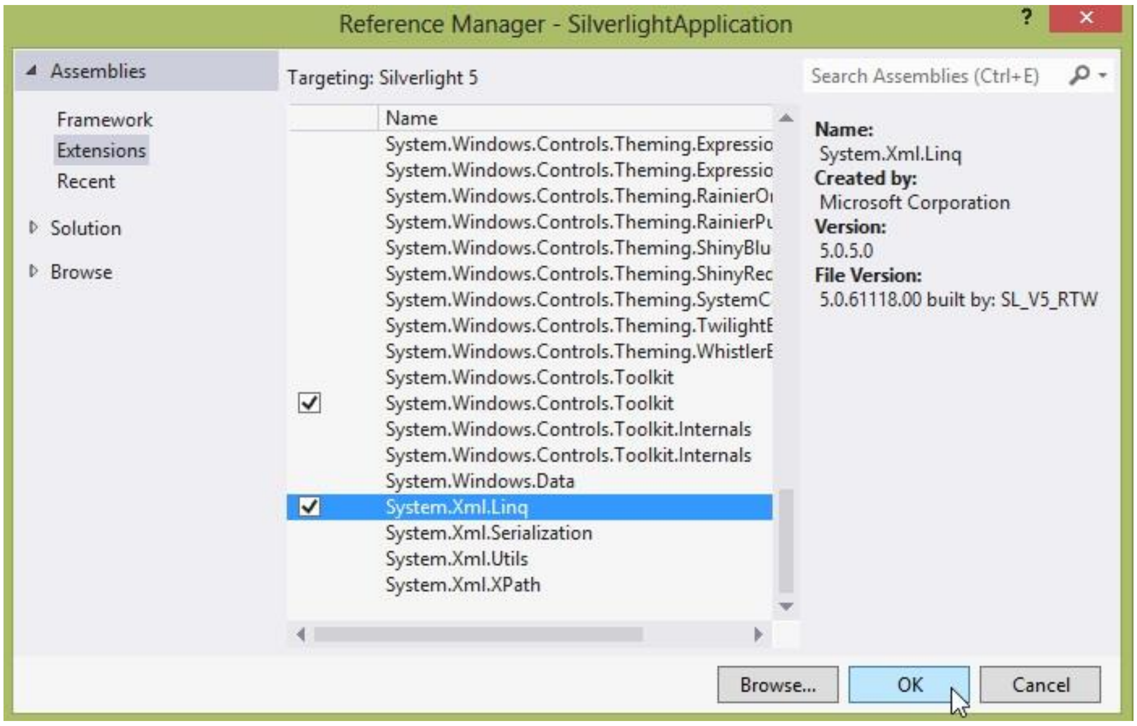


PerpetuumSoft.Reporting.Silverlight.Client.dll 30.04.2013 2:48

File name: PerpetuumSoft.Reporting.Silverlig

Executable Files (*.dll)

You should also add a reference to ‘System.Windows.Controls.Toolkit.dll’ and ‘System.Xml.Linq’ assemblies.

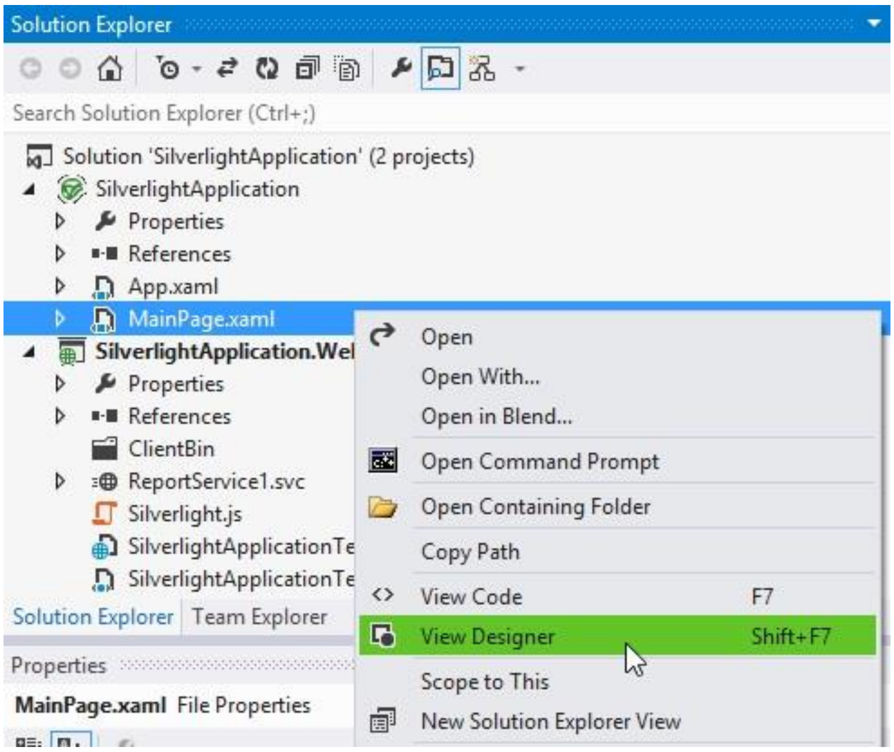


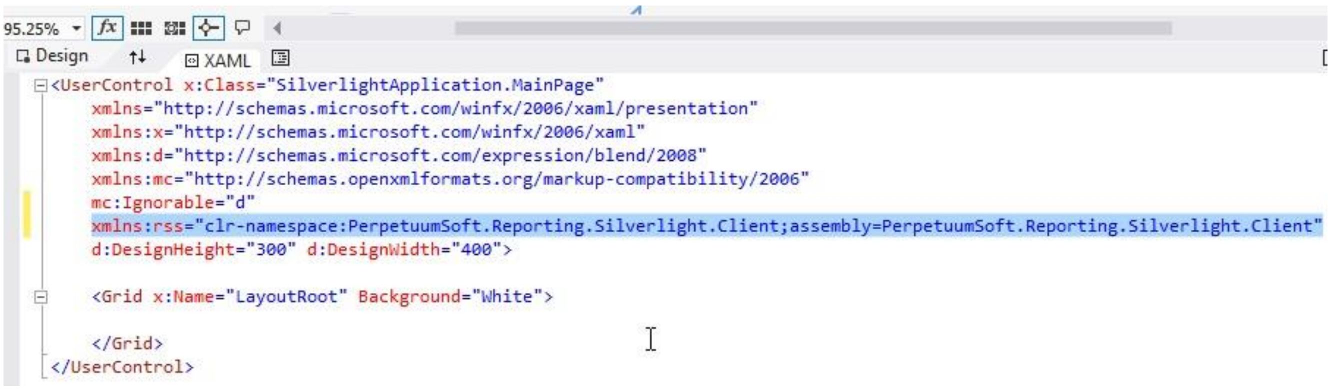
	Name
	System.Windows.Controls.Theming.Expressio
	System.Windows.Controls.Theming.Expressio
	System.Windows.Controls.Theming.RainierOr
	System.Windows.Controls.Theming.RainierPu
	System.Windows.Controls.Theming.ShinyBlu
	System.Windows.Controls.Theming.ShinyRec
	System.Windows.Controls.Theming.SystemC
	System.Windows.Controls.Theming.TwilightE
	System.Windows.Controls.Theming.WhistlerE
	System.Windows.Controls.Toolkit
✓	System.Windows.Controls.Toolkit
	System.Windows.Controls.Toolkit.Internals
	System.Windows.Controls.Toolkit.Internals
	System.Windows.Data
✓	System.Xml.Linq
	System.Xml.Serialization
	System.Xml.Utils
	System.Xml.XPath

Name: System.Xml.Linq
Created by: Microsoft Corporation
Version: 5.0.5.0
File Version: 5.0.61118.00 built by: SL_V5_RTW

After that, open MainPage.xaml in the Designer and add “rss” xml namespace for the PerpetuumSoft.Reporting.Silverlight.Client assembly as follows:

xmlns:rss="clr-namespace:PerpetuumSoft.Reporting.Silverlight.Client;assembly=PerpetuumSoft.Reporting.Silverlight.Client"



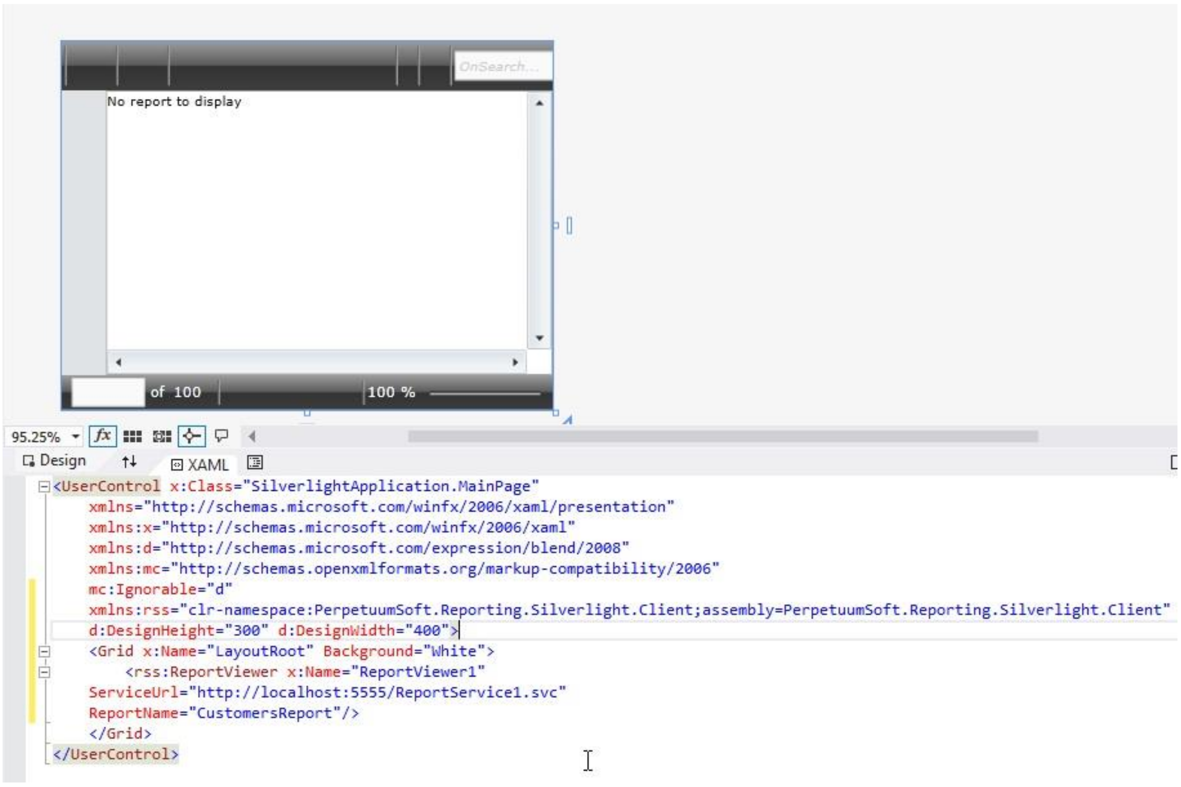


```
<UserControl x:Class="SilverlightApplication.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    xmlns:rss="clr-namespace:PerpetuumSoft.Reporting.Silverlight.Client;assembly=PerpetuumSoft.Reporting.Silverlight.Client"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">

    </Grid>
</UserControl>
```

Then add ReportViewer element to the Grid Section.

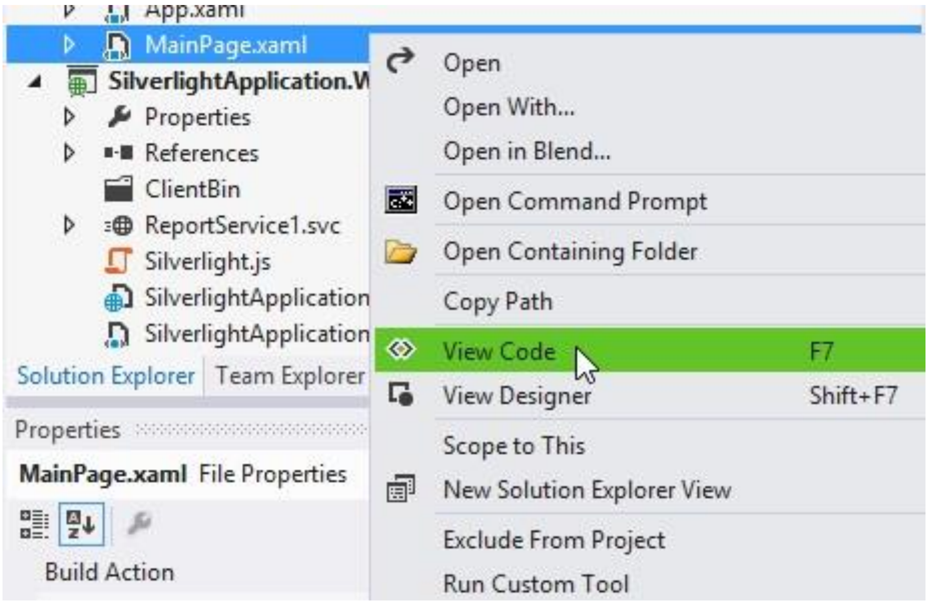
<rss:ReportViewer x:Name="ReportViewer1"
  ServiceUrl="http://localhost:5555/ReportService1.svc"
  ReportName="CustomersReport"/>

After you made changes the UserControl section should look like this:

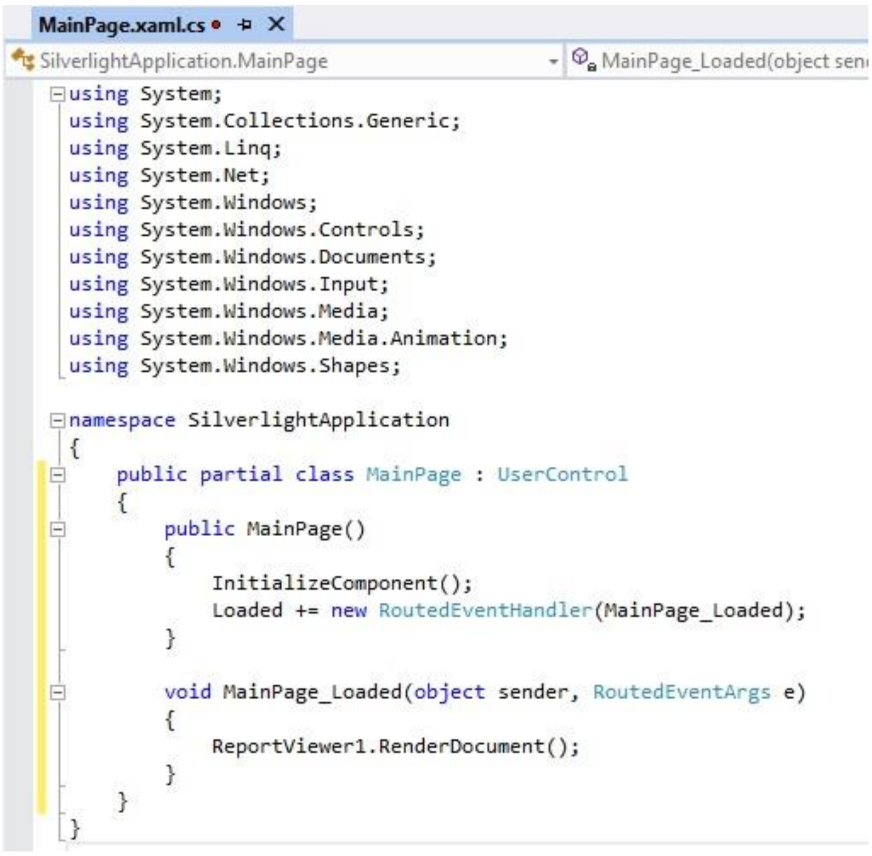


```
<UserControl x:Class="SilverlightApplication.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    xmlns:rss="clr-namespace:PerpetuumSoft.Reporting.Silverlight.Client;assembly=PerpetuumSoft.Reporting.Silverlight.Client"
    d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">
        <rss:ReportViewer x:Name="ReportViewer1"
    ServiceUrl="http://localhost:5555/ReportService1.svc"
    ReportName="CustomersReport"/>
    </Grid>
</UserControl>
```

After that, open MainPage source code and change/add the lines to make code look like this:



```
public partial class MainPage : UserControl
{
        public MainPage()
        {
            InitializeComponent();
            Loaded += new RoutedEventHandler(MainPage_Loaded);
        }

        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            ReportViewer1.RenderDocument();
        }
}
```

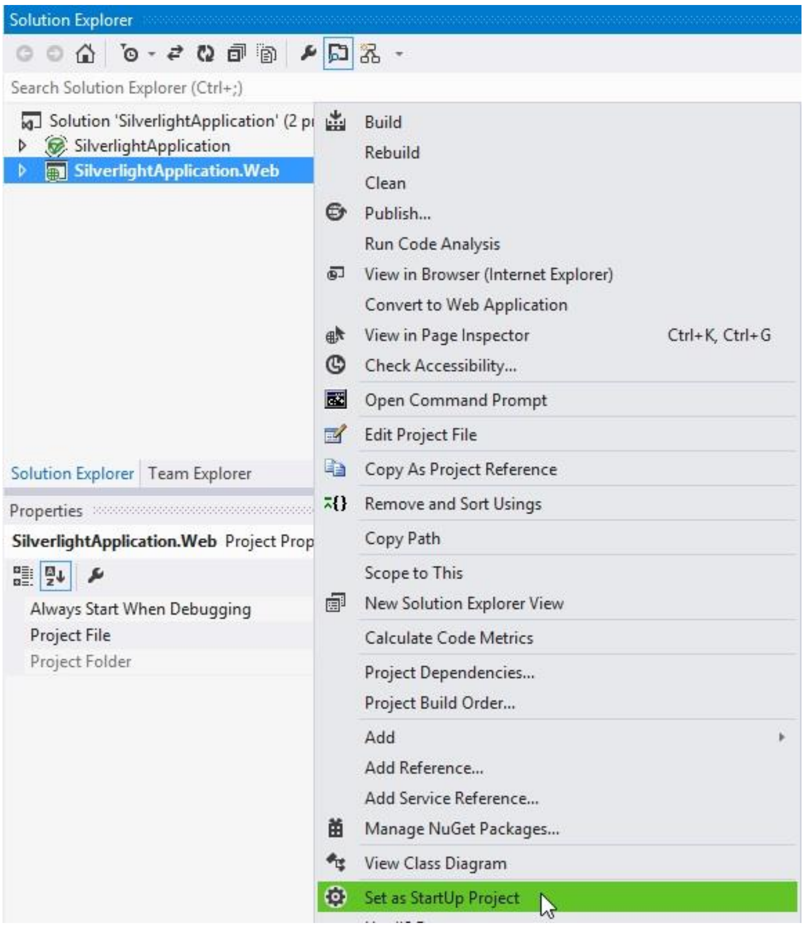


```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;

namespace SilverlightApplication
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
            Loaded += new RoutedEventHandler(MainPage_Loaded);
        }

        void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            ReportViewer1.RenderDocument();
        }
    }
}
```

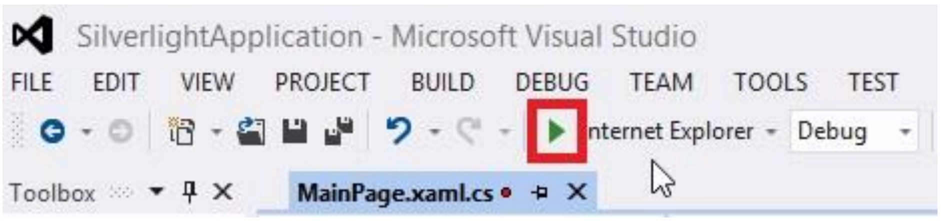
Note: RenderDocument() invocation leads to the rendering of the current report on the server and displaying it in the Report Viewer.

Step 7. Launching an Application.

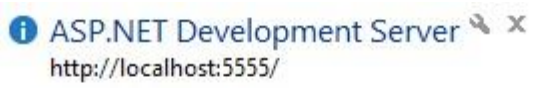
1) To launch our application we need to set the SilverlightApplication.Web project as a StartUp. In order to do that, right click SilverlightApplication.Web and select ‘Set as StartUp Project’ in the popup menu.



Then launch an application by clicking the “Start Debugging” button on the main Visual Studio toolbar.

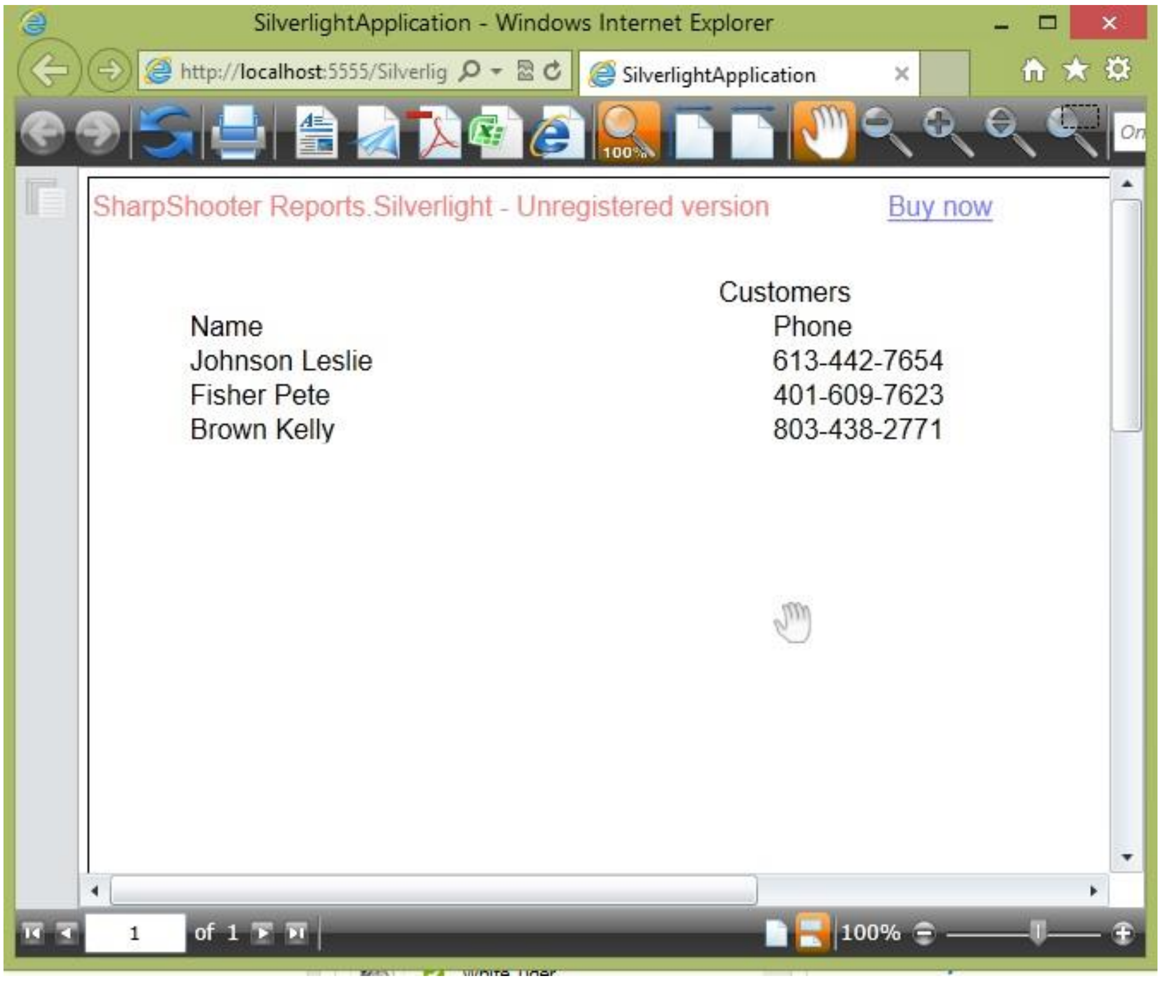


The popup message indicating that the Development server has been started on port 5555 will appear in the system tray.



And the report rendering progress bar is displayed.

As a result, we will have our sample report presented.


SilverlightApplication - Windows Internet Explorer
http://localhost:5555/Silverlig
SilverlightApplication
SharpShooter Reports.Silverlight - Unregistered version
Buy now
1
of 1
100%

Customers

Name	Phone
Johnson Leslie	613-442-7654
Fisher Pete	401-609-7623
Brown Kelly	803-438-2771

Conclusion

We have examined the basic steps and got a simple and quite operable application. In our case we did not have to write thousand lines of code – we just used a ready-made implementation. In most cases this will be enough. If you require a more complex behavior than the one provided in this sample, you can change a lot of aspects in SharpShooter Reports.Silverlight operation as well as the appearance of the Report Viewer itself.

If you have any questions regarding the SharpShooter Reports.Silverlight integration don't hesitate to contact us at support@perpetuumsoft.com