

A SharePoint Developer Introduction

Hands-On Lab

Lab Manual

SPCHOL306 – Using Silverlight with the Client Object Model – C#

This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2010 Microsoft. All rights reserved.

**Contents**

[SPCHOL306 – Using Silverlight with the Client Object Model 2](#_Toc260142378)

[Lab Objective 2](#_Toc260142379)

[Additional Resources 2](#_Toc260142380)

[Getting Started 2](#_Toc260142381)

[Logging in to the Virtual Machine 2](#_Toc260142382)

[Locations 2](#_Toc260142383)

[Lab Pre-requisites 3](#_Toc260142384)

[Copying code samples from Word document 3](#_Toc260142385)

[Code Snippets 4](#_Toc260142386)

[Exercise 1 – Create a Silverlight Application for the SharePoint Client Object Model 5](#_Toc260142387)

[Task 1 – Create a Silverlight Application Project 5](#_Toc260142388)

[Task 2 – Write code to access and render SharePoint List Data 7](#_Toc260142389)

[Task 3 – Deploy and Test using the SharePoint Silverlight web part 11](#_Toc260142390)

[Exercise 2 – Creating a graph using the SharePoint Object Model and Silverlight Graphing controls 14](#_Toc260142391)

[Task 1 – Create a Silverlight Application Project 14](#_Toc260142392)

[Task 2 – Write code to access Employee SharePoint List Data and display it in a Silverlight Graph Control 18](#_Toc260142393)

[Task 3 – Deploy and Test using the SharePoint Silverlight Charting web part 21](#_Toc260142394)

[Lab Summary 24](#_Toc260142395)

# SPCHOL306 – Using Silverlight with the Client Object Model

Estimated time to complete this lab: **30 minutes**

Visual Studio 2010, SharePoint Foundation 2010, and the Silverlight 3 Toolkit are required for these exercises. These are installed on the Virtual Machine used in this lab.

## Lab Objective

The objective of this lab is to learn about how to create and host Silverlight Applications inside SharePoint 2010, and how to use the SharePoint Client Object model to access SharePoint data from within Silverlight.

* Create a basic Silverlight application that displays a SharePoint list inside a datagrid and deploy the Silverlight application to SharePoint.
* Create a Silverlight application that displays SharePoint list data in a graph using the Silverlight Graphing controls.

## Additional Resources

This lab includes the following additional resources:

|  |  |  |
| --- | --- | --- |
| This Lab Manual | SPCHOL306\_Manual\_CS.docx | This document |
| Source Code | \Supporting Files\SPCHOL306\Completed\CS\Ex1\ \Supporting Files\SPCHOL306\Completed\CS\Ex2\ | Completed lab source code in C#. |
| Resources | \Supporting Files\SPCHOL306\Resources\CS\ | Various resources used throughout this lab. |

## Getting Started

### Logging in to the Virtual Machine

Please log into the virtual machine as the following user:

**Username:** Administrator

**Password:** pass@word1

### Locations

This Hands-On Lab contains a number of additional resources in fixed locations. By default, it is assumed that the base HOL Resources directory is **C:\Content Packs\Packs\SharePoint 2010 Developer Labs 1.0\Supporting Files\SPCHOL306\Resources**.

The default working folder for this lab is C:\SPHOLS\SPCHOL306.

### Lab Pre-requisites

Browse to base HOL directory ***Supporting Files\SPCHOL306\Resources***and execute the **optimize.ps1** PowerShell script:

1. Right-click on **optimize.ps1** and select **Run with PowerShell**:

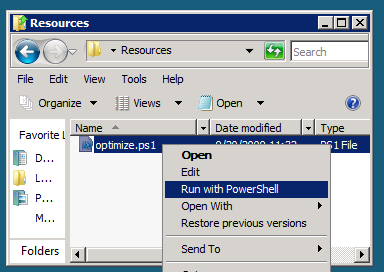


Figure 1 - Execute the PowerShell script

1. This will open a PowerShell window to execute the script. Please wait until the PowerShell script completes executing the script and closes the PowerShell window (this may take a few moments):

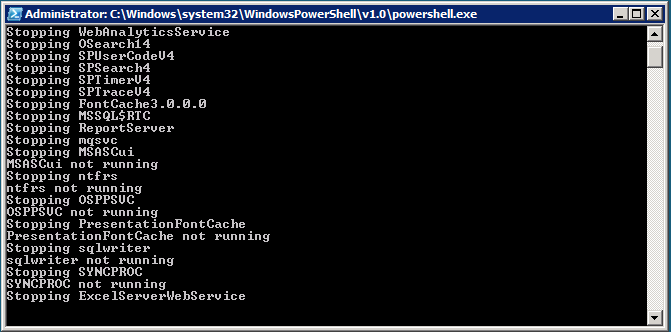


Figure 2 - PowerShell Window executing the script

### Copying code samples from Word document

Copying and pasting code from this Word document to Visual Studio is only safe for the sections of formatted code, e.g.:

Console.WriteLine("This is safe code!");

Code not in these sections may contain Unicode or other invisible characters that are not valid XML or C#/VB code, e.g.:

**Console.WriteLine(“This is NOT safe code !!”);**

### Code Snippets

You can also use Code Snippets to insert the appropriate code in the lab.

To use the required code snippet for this lab:

* Right-click on the code file where you want to insert the code snippet.
* Select **Insert Snippet:**

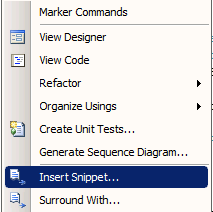


Figure 3 - Visual Studio Code Context Menu.

* Select code snippets from **My Code Snippets** gallery.

## Exercise 1 – Create a Silverlight Application for the SharePoint Client Object Model

In this exercise, we will create a basic Silverlight application that displays a SharePoint list inside a datagrid. The data for the list is retrieved from the server so it is available in Silverlight using the SharePoint Client Object Model. We will use a Visual Studio 2010 solution and deploy it to the local SharePoint server.

### Task 1 – Create a Silverlight Application Project

We will start by creating a standard Silverlight application project.

1. Open Visual Studio 2010 from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
2. From the menu, select **File | New | Project**.
3. In the New Project dialog box, expand the Installed Templates left hand menu to **Other Project Types | Visual Studio Solutions | Blank Solution**.
4. Select **.NET Framework 3.5**
5. Name the solution **Begin**.
6. Enter **C:\SPHOLS\SPCHOL306\CS\Ex1\** in the Location textbox.

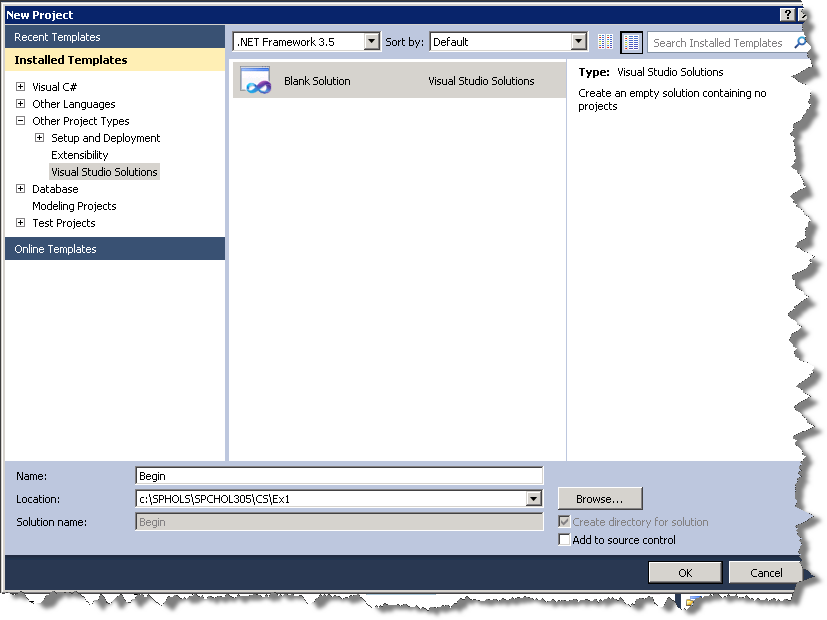
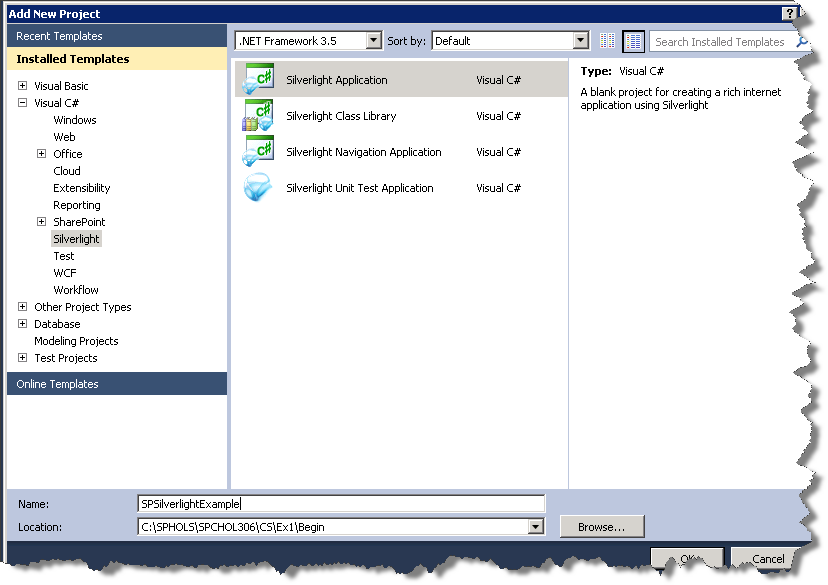


Figure 4 - New Project Dialog

1. Press **OK** to continue.
2. From the menu, select **File | Add | New Project**.
3. In the New Project dialog box, expand the Installed Templatesleft hand menu to **Visual C# | Silverlight**, and choose the **Silverlight Application** project type in the project type list in the middle section of the screen.
4. Select **.NET Framework 3.5**
5. Name the project **SPSilverlightExample**.
6. Leave the location unchanged.

  
Figure 5 - Add New Project Dialog

1. Press **OK** to continue.

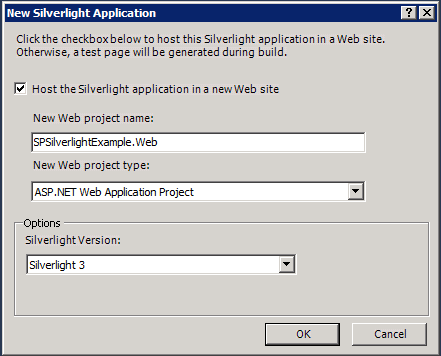


Figure 6 - Add Silverlight Application Dialog

1. Press **OK** to create the project.
2. Within the **SPSilverlightExample** project we will now add the reference assemblies to the SharePoint Silverlight Client Object Model. Right-click **References** in the **SPSilverlightExample** project and select **Add References**.
3. Browse to the “C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin” folder.
4. Select **Microsoft.SharePoint.ClientSilverlight.dll** and **Microsoft.SharePoint.Client.Silverlight.Runtime.dll** (hold CTRL to select multiple files)
5. Press **OK** to add the select dll references.

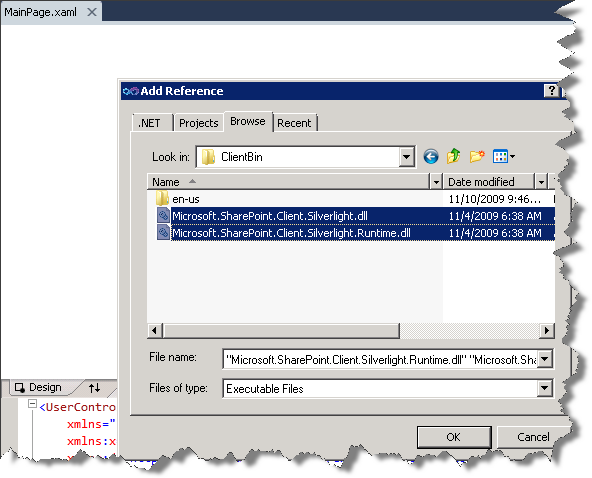


Figure 7 - Add References

### Task 2 – Write code to access and render SharePoint List Data

1. In Visual Studio open the **Toolbox** and expand **Silverlight Controls**.
2. Drag a **DataGrid** control onto the existing Grid in the Page.xaml Silverlight Designer.

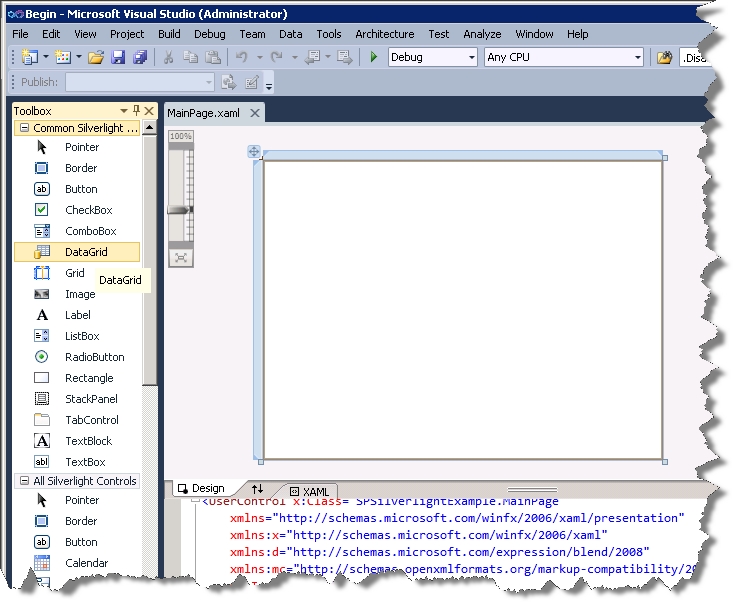


Figure 8 - Silverlight Controls Toolbox

1. Expand the DataGrid to take up the entire page by going to the properties grid and setting its **Width** and **Height** property to **Auto**, its **HoriztonalAlignment** and **VerticalAlignment** properties to **Stretch**, and its Margin to 0..

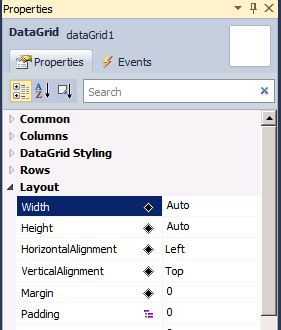


Figure 9 - DataGrid Properties

1. Also make sure to check **AutoGenerateColumns** (or set its value to true in the XAML).

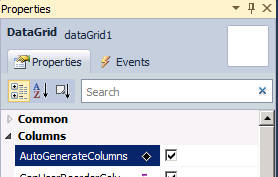


Figure 10 - AutoGenerate Columns Property

1. Open **App.xaml.cs** and add the following using statements to the top of the file:

using Microsoft.SharePoint.Client;

using System.Threading;

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_app\_namespaces*

1. Add the following code to the beginning of the **Application\_Startup** method.

ApplicationContext.Init(e.InitParams, SynchronizationContext.Current);

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_* *application\_startup*

1. Open **MainPage.xaml.cs** and add the following using statement to the top of the file:

using Microsoft.SharePoint.Client;

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_page\_namespaces*

1. Add the following class before the MainPage class:

public class Project

{

public string Title { get; set; }

public DateTime DueDate { get; set; }

public string Description { get; set; }

}

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_* *classes*

1. Add the following variable to the MainPage class:

private ListItemCollection \_projects;

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_* *property*

1. Add the following code to the Page constructor below the call to InitializeComponent:

ClientContext context = new ClientContext(ApplicationContext.Current.Url);

context.Load(context.Web);

List Projects = context.Web.Lists.GetByTitle("Projects");

context.Load(Projects);

CamlQuery query = new Microsoft.SharePoint.Client.CamlQuery();

string camlQueryXml = "<View><Query><Where><Gt>" +

"<FieldRef Name='Due\_x0020\_Date' />" +

"<Value Type='DateTime'>2008-01-1T00:00:00Z</Value>" +

"</Gt></Where></Query><ViewFields>" +

"<FieldRef Name=\"Title\" /><FieldRef Name=\"Description\" />" +

"<FieldRef Name=\"Due\_x0020\_Date\" />" +

"</ViewFields></View>";

query.ViewXml = camlQueryXml;

\_projects = Projects.GetItems(query);

context.Load(\_projects);

context.ExecuteQueryAsync(new ClientRequestSucceededEventHandler(OnRequestSucceeded), null);

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_initializecomponent*

1. Add the following code after the constructor:

private void OnRequestSucceeded(Object sender, ClientRequestSucceededEventArgs args)

{

// this is not called on the UI thread

Dispatcher.BeginInvoke(BindData);

}

private void BindData()

{

List<Project> projects = new List<Project>();

foreach (ListItem li in \_projects)

{

projects.Add(new Project()

{

Title = li["Title"].ToString(),

DueDate = Convert.ToDateTime(li["Due\_x0020\_Date"].ToString()),

Description = li["Description"].ToString()

});

}

dataGrid1.ItemsSource = projects; // must be on UI thread

}

**Code Snippet**: *My Code Snippets | spchol306\_ex1\_methods*

This code initializes the SharePoint Silverlight client object model context (ClientContext). It then gets a reference to the Projects list and runs a simple CAML query against the list to pull all projects with a due date greater than 1/1/2008. The results are converted into a list of Projects and bound to the Silverlight DataGrid control.

### Task 3 – Deploy and Test using the SharePoint Silverlight web part

To deploy the solution to SharePoint the resulting .xap file created by the Silverlight project needs to be in the **C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin** folder.

1. Right-click the **SPSilverlightExample** project, select properties and select the **Build** tab.
2. Change the output path to **C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin**.

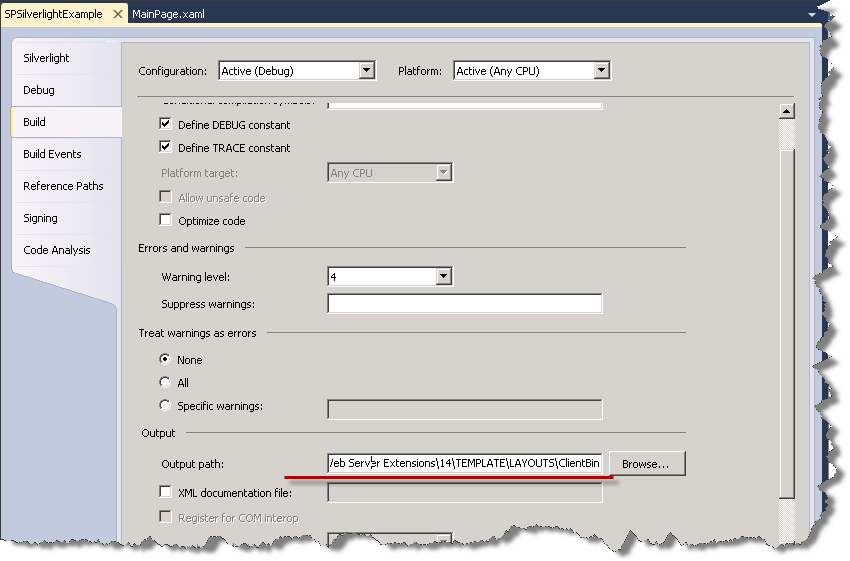


Figure 11 - Project Properties

1. Build the solution. The .xap file has been copied to the required SharePoint directory and you are ready to add the Silverlight web part to the SharePoint website.
2. Open Internet Explorer and browse to <http://intranet.contoso.com>.
3. Select the **Edit** icon at the top of the page.
4. Select the **Insert** tab and the click **Web Part**.
5. From the Category list select **Media and Content**, choose the **Silverlight Web Part** from the web part list, and click **Add**.
6. In the Silverlight Web Part dialog that pops up enter **/\_layouts/ClientBin/SPSilverlightExample.xap** as the URL.

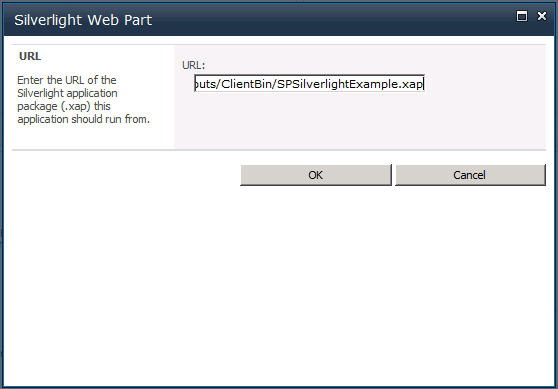


Figure 12 - Silverlight Web Part Url Dialog

1. Click **OK** to save the Silverlight Web Part.
2. Exit edit mode.
3. The final web part will look like this on the SharePoint page:

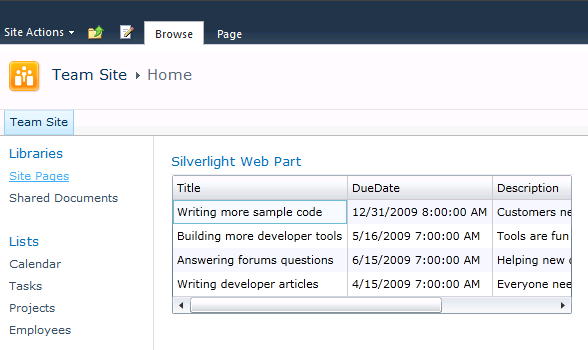


Figure 13 - Finished Silverlight Web Part

In this exercise you have created a basic Silverlight application that displays a SharePoint list inside a datagrid. The data for the list is retrieved from the server so it is available in Silverlight using the SharePoint Client Object Model.

## Exercise 2 – Creating a graph using the SharePoint Object Model and Silverlight Graphing controls

In exercise two we will again be using the SharePoint Object Model to access SharePoint list data but this time we will use LINQ and the Silverlight Charting controls to display the data in a graph.

### Task 1 – Create a Silverlight Application Project

1. Open Visual Studio 2010 from **Start | All Programs | Microsoft Visual Studio 2010 | Microsoft Visual Studio 2010**.
2. From the menu, select **File | New | Project**.
3. In the New Project dialog box, expand the Installed Templates left hand menu to **Other Project Types | Visual Studio Solutions | Blank Solution**.
4. Name the solution **Begin**.
5. Change the location to ***C:\SPHOLS\SPCHOL306\CS\Ex2\***

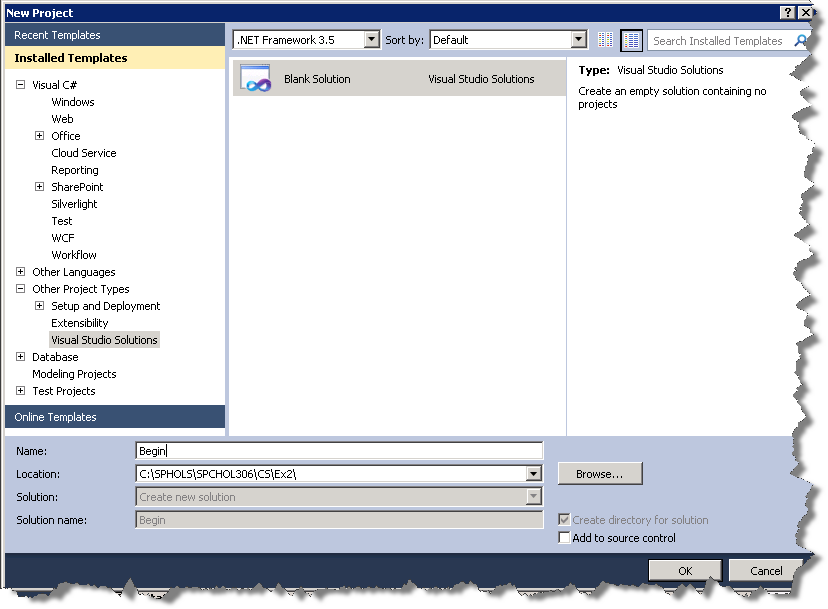


Figure 14 - New Project Dialog

1. Press **OK** to continue.
2. From the menu, select **File | Add | New Project**.
3. In the New Project dialog box, expand the **Installed Templates** left hand menu to **Visual C# | Silverlight**, and choose the **Silverlight Application** project type in the project type list in the middle section of the screen.
4. Name the project **SilverlightEmployeeContributionsGraph**.
5. Leave the location unchanged.

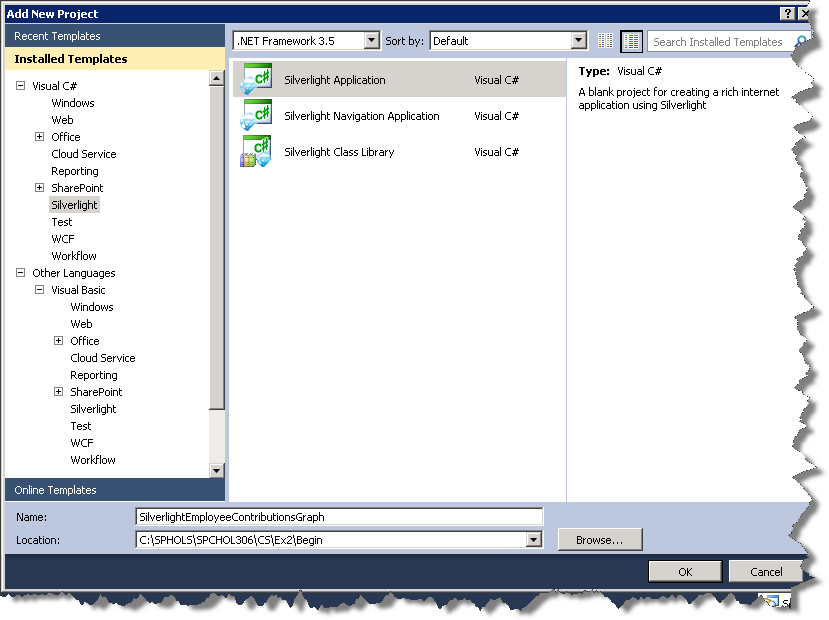


Figure 15 - Add New Project Dialog

1. Click **OK**.

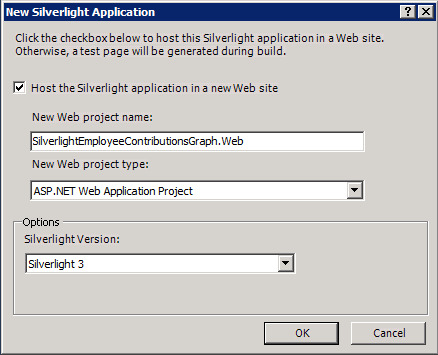


Figure 16 - Add Silverlight Application Dialog

1. Press **OK** to create the project.
2. Within the SPSilverlightExample project we will now add the reference assemblies to the SharePoint Silverlight Client Object Model. Right-click **References** in the **SilverlightEmployeeContributionsGraph** project and select **Add References**.
3. Browse to the **C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin** folder.
4. Select **Microsoft.SharePoint.ClientSilverlight.dll** and **Microsoft.SharePoint.Client.Silverlight.Runtime.dll** (hold CTRL to select multiple files)
5. Press **OK** to add the select dll references.

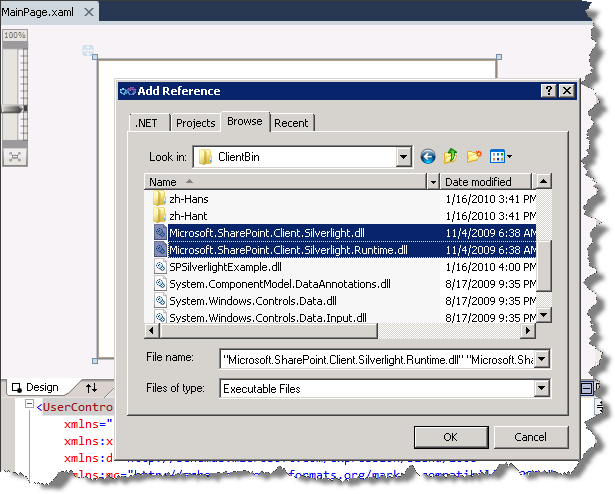


Figure 17 - Add References

1. Add a reference to the **Silverlight Charting Controls** assembly. It is available on the **.NET tab** and is called the **System.Windows.Controls.DataVisualization.Toolkit**.

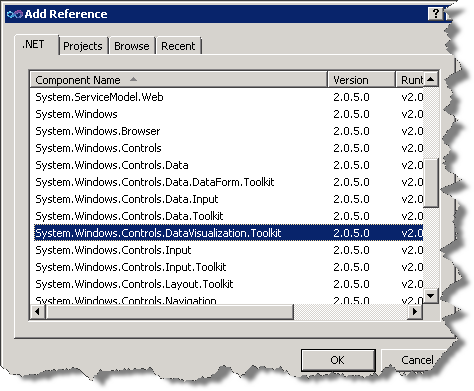


Figure 18 - Add DataVisualization Reference

### Task 2 – Write code to access Employee SharePoint List Data and display it in a Silverlight Graph Control

1. In the Solution Explorer **right-click** on the **App.xaml** file and select **View Code**. In the open **App.xaml.cs** and add the following using statements to the top of the file:

using Microsoft.SharePoint.Client;

using System.Threading;

**Code Snippet**: *My Code Snippets | spchol306\_ex2\_app\_namespaces*

1. Add the following code to the beginning of the **Application\_Startup** method.

ApplicationContext.Init(e.InitParams, SynchronizationContext.Current);

**Code Snippet**: *My Code Snippets | spchol306\_ex2\_application\_startup*

1. In the XAML view of the **MainPage.xaml** file and add the following XML namespace in the UserControl element:

xmlns:chartingToolkit="clr-namespace:System.Windows.Controls.DataVisualization.Charting;assembly=System.Windows.Controls.DataVisualization.Toolkit"

1. Add the following Silverlight Charting control inside the Grid element:

<chartingToolkit:Chart x:Name="chart" Width="350" Height="250" Title="Team Contributions">

<chartingToolkit:Chart.Series>

<chartingToolkit:ColumnSeries ItemsSource="{Binding}"

DependentValuePath="Contributions"

IndependentValuePath="Name"

AnimationSequence="FirstToLast"

Title="Contributions" IsSelectionEnabled="True" />

</chartingToolkit:Chart.Series>

</chartingToolkit:Chart>



Figure 19 - Silverlight Chart XAML

1. Open **MainPage.xaml.cs** and add the following using statement to the top of the file:

using Microsoft.SharePoint.Client;

**Code Snippet**: *My Code Snippets | spchol306\_ex2\_page\_namespaces*

1. Add the following classes before the **MainPage** class:

public class EmployeeContributions

{

public string Name { get; set; }

public string TeamName { get; set; }

public decimal Contributions { get; set; }

}

public class TeamContributions

{

public string Name { get; set; }

public decimal Contributions { get; set; }

}

**Code Snippet**: *My Code Snippets | spchol306\_ex2\_classes*

1. Add the following variable to the **MainPage** class:

private ListItemCollection \_employees;

**Code Snippet**: *My Code Snippets | spchol306\_ex2\_property*

1. Add the following code to the Page constructor below the call to InitializeComponent:

ClientContext context = new ClientContext(ApplicationContext.Current.Url);

context.Load(context.Web);

List employees = context.Web.Lists.GetByTitle("Employees");

context.Load(employees);

CamlQuery query = new CamlQuery();

string camlQueryXml = null;

query.ViewXml = camlQueryXml;

\_employees = employees.GetItems(query);

context.Load(\_employees);

context.ExecuteQueryAsync(new ClientRequestSucceededEventHandler(OnRequestSucceeded), null);

**Code Snippet**: My Code Snippets | spchol306\_ex2\_initializecomponent

1. Add the following code after the constructor:

private void OnRequestSucceeded(Object sender, ClientRequestSucceededEventArgs args)

{

// this is not called on the UI thread

Dispatcher.BeginInvoke(BindData);

}

private void BindData()

{

List<EmployeeContributions> employees = new List<EmployeeContributions>();

// get list item values into a strongly typed class

foreach (ListItem li in \_employees)

{

employees.Add(new EmployeeContributions

{

Name = li["Title"].ToString(),

TeamName = li["Team"].ToString(),

Contributions = Convert.ToDecimal(li["Contribution\_x0020\_\_x0028\_in\_x00"])

});

}

// use linq to group employees on team name and then total team contributions

List<TeamContributions> teams = employees

.GroupBy(e => e.TeamName)

.Select(t => new TeamContributions

{

Name = t.Key,

Contributions = t.Sum(e => e.Contributions)

}).ToList();

chart.DataContext = teams; // must be on UI thread

}

**Code Snippet**: My Code Snippets | spchol306\_ex2\_methods

1. Like the previous exercise the SharePoint Silverlight client object model is used to retrieve data from a SharePoint list. Once the employee contribution items have been populated into a list, LINQ is then used to group employees into teams and their contributions summed together. Team contributions are then set as the chart’s data source.

### Task 3 – Deploy and Test using the SharePoint Silverlight Charting web part

To deploy the solution to SharePoint the resulting .xap file created by the Silverlight project needs to be in the **C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin** folder.

1. Right-click the **SilverlightEmployeeContributionsGraph** project, select properties and select the **Build** tab.
2. Change the output path to **C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\14\TEMPLATE\LAYOUTS\ClientBin**.

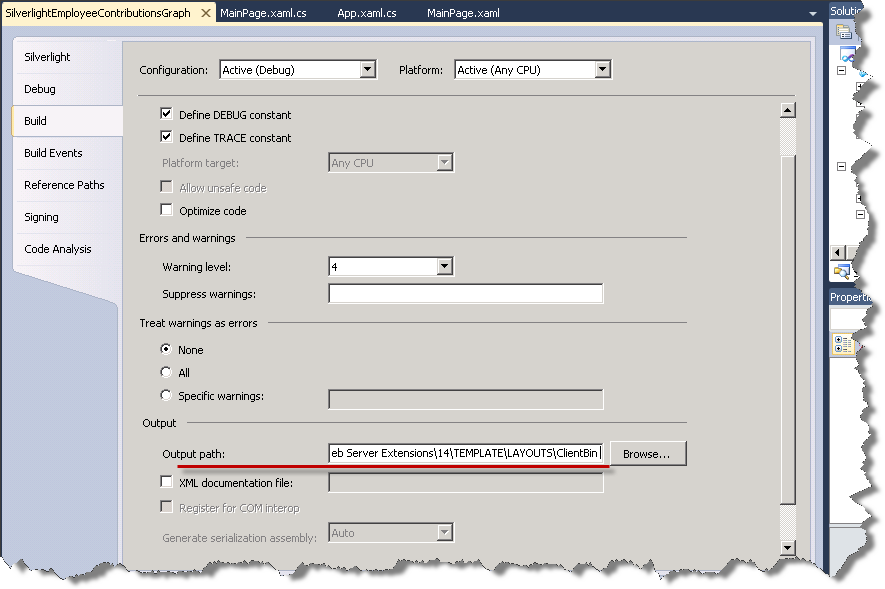


Figure 20 - Silverlight Project Properties

1. Build the solution. The .xap file has been copied to the required SharePoint directory and you are ready to add the Silverlight web part to the SharePoint website.
2. Open Internet Explorer and browse to <http://intranet.contoso.com>.
3. We will update the Silverlight web part added in the previous exercise to point toward the new Silverlight Charting control we have just made. Click the dropdown icon in the top right hand corner of the Silverlight web part and select **Edit Web Part**.

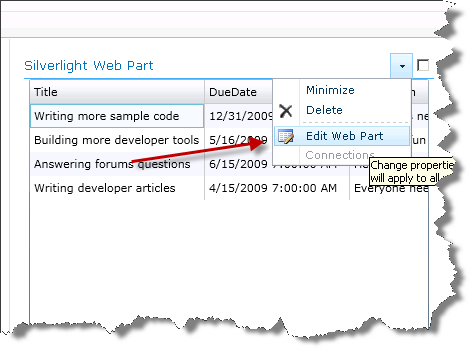


Figure 21 - Silverlight Web Part Properties

1. Click the **Configure** button (you may have to scroll the window to the right to see the Configure button) and then enter **/\_layouts/ClientBin/SilverlightEmployeeContributionsGraph.xap** in the Silverlight Web Part dialog.

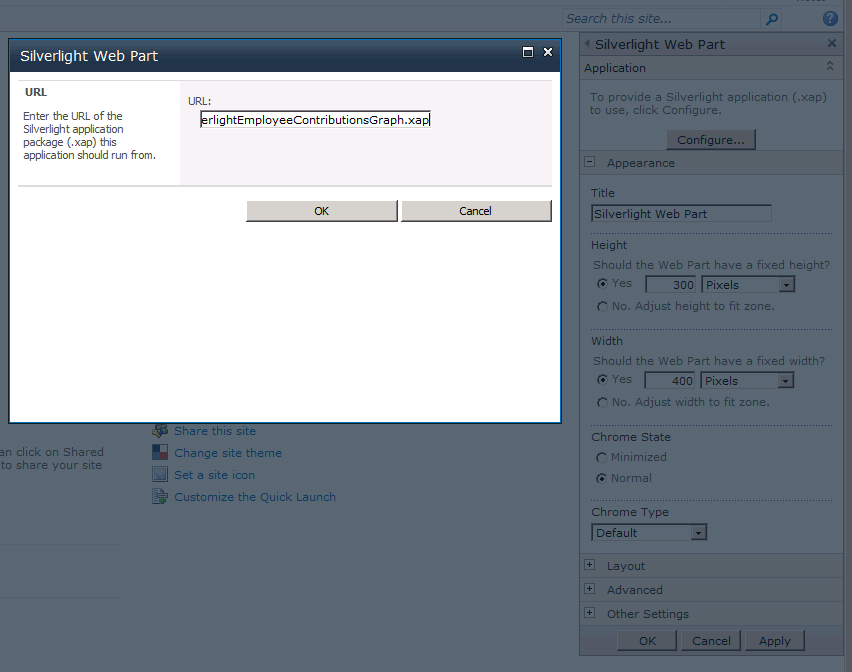


Figure 22 - Silverlight Web Part URL

1. Click **OK**.
2. Click **OK** at the bottom of the Silverlight Web Part sidebar.
3. The final web part will look like this:

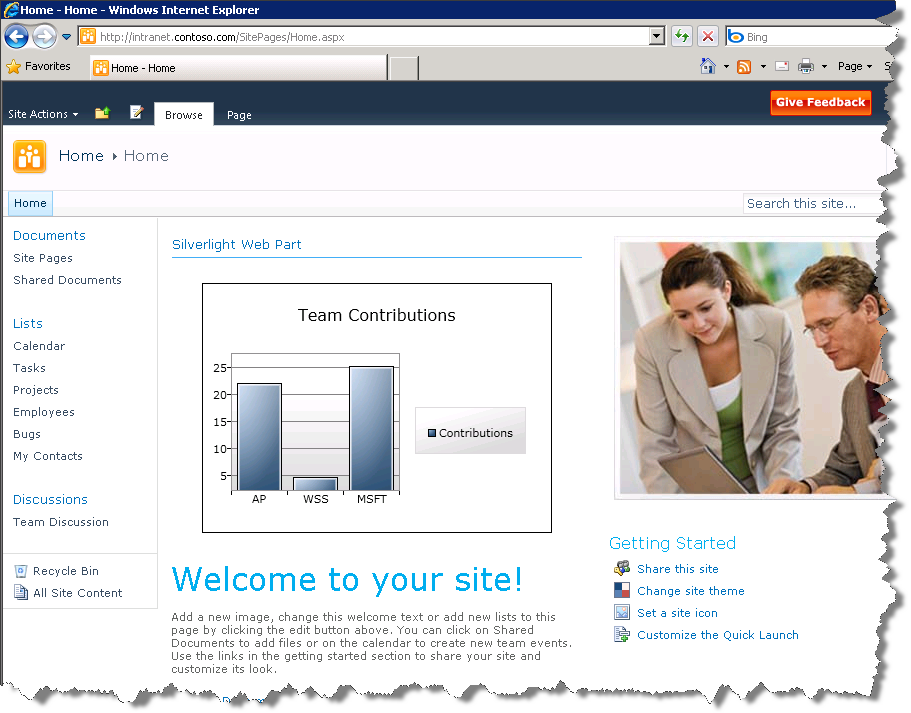


Figure 23 - Finished Silverlight Chart Web Part

## Lab Summary

In this lab you performed the following exercises.

Created new Silverlight Application project

Added references to the SharePoint Silverlight object model dlls.

Written code to get data from a SharePoint using the SharePoint object model

Deployed a Silverlight application to SharePoint and displayed it in the new SharePoint Silverlight web part.

Displayed SharePoint list data in a Silverlight datagrid.

Used LINQ to manipulate items returned from a SharePoint list.

Displayed SharePoint list data in a Silverlight Charting bar graph.