Microsoft® Silverlight™

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Silverlight is a platform for building rich internet applications. This tutorial will explain the concepts behind Silverlight, and will show you how to build it into your web applications. After completing this tutorial, you will have a better understanding of Silverlight applications and how to develop them using XAML and C#.

Audience

This tutorial has been prepared for anyone who has a basic knowledge of XAML and C# and has an urge to develop websites. After completing this tutorial, you will find yourself at a moderate level of expertise in developing websites using Silverlight.

Prerequisites

Before you start proceeding with this tutorial, we are assuming that you are already aware about the basics of XAML and C#. If you are not well aware of these concepts, then we will suggest you to go through our short tutorials on XAML and C#.

Copyright & Disclaimer

Table of Contents

About the Tutorial i
Audience i
Prerequisites i
Copyright & Disclaimer i
Table of Contents ii

1. Silverlight – Overview 1
What is Silverlight 1
Platforms and Browsers 2

2. Silverlight – Environment Setup 4
Installation 4

3. Silverlight – Getting Started 9
Create a Web-page 9

4. Silverlight – XAML Overview 18
Basic Syntax 18
Why XAML in Silverlight 20
XAML & Code Behind 20

5. Silverlight – Project Types 23
Silverlight Web Applications 24
Silverlight Navigation Application 31

6. Silverlight – Fixed Layouts 37
Fixed Layout 37

7. Silverlight – Dynamic Layout 41
Stack Panel 41
Grid 44

8. Constrained vs. Unconstrained Layout 49
GridSplitter 50
ScrollViewer 56
Border 61
Full Screen Mode 67

9. Silverlight and CSS 71
Overlapping Content 74

10. Silverlight – Controls 80

11. Silverlight – Buttons 81
HyperlinkButton 86
The ToggleButton and RepeatButton 89
CheckBox 93
RadioButton 99

12. Silverlight – Content Model 109
RangeControl 110

13. Silverlight – ListBox 117
Calendar & DatePicker 124
TabControl 132
Popup 135
ToolTip 138

14. Silverlight – Templates 141

15. Silverlight – Visual State 143
State & State Group 143
Visual State Manager 144

16. Silverlight – Data Binding 149
One-way Data Binding 149
Two-way data binding 153

17. Silverlight – Browser Integration 157
Silverlight and HTML 157
Accessing DOM 157

18. Silverlight – Out-of-Browser Applications 164
Interaction 164
Offline 164
Elevated Trust 165
Enabling OOB 165
OOB Settings 173

19. Silverlight – Applications, Resources, and Deployment 176
Loading the Plug-in 176
Type Attibute 176
Data Attribute 177
<param> Tags 177
Fallback HTML Content 178
Silverlight.js 178
XAML Resources 179
App.xaml 180
Application Class 181

20. Silverlight – File Access 183
Open & Save File Dialogs 183

21. Silverlight – View Model 190
UI Development Challenges 190
Separated Presentation 191
Model / View / ViewModel 192
UI vs ViewModel 199

22. Silverlight – Input Handling 201
Input Types 201
Mouse Events 201
Keyboard 205

23. Silverlight – Isolated Storage 208
Using Isolated Storage 208
Increasing Your Quota 211

24. Silverlight – Text 216
TextBlock 216
Run 218
LineBreak 220
Built-in Fonts 221

25. Silverlight – Animation 223
Defining Animations 223
Repeating and Reversing 226
Key Frame Animation 226

26. Silverlight – Video and Audio 230
MediaElement as UI Element 230
Controlling 233

27. Silverlight – Printing 237
Steps for Printing 237
Printing Existing Elements 238
Custom UI Tree 242

1. Silverlight – Overview

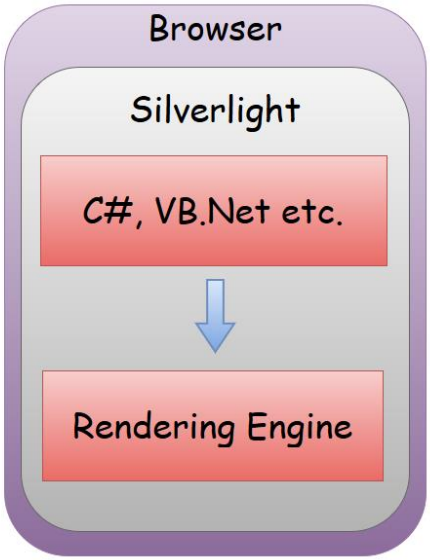
Welcome to Silverlight tutorials. Silverlight is a platform for building rich internet applications. This tutorial will explain the concepts behind Silverlight, and will show you how to build it into your web applications. After completing it, you will have a better understanding of Silverlight applications using XAML and C#.

What is Silverlight

Silverlight is a browser plug-in, designed for building rich internet applications; applications that run in the browser like normal web applications, but which try to advance the user interface beyond where HTML can go. For example,

- Silverlight is a framework for building rich, browser-hosted applications that run on a variety of operating systems.
- It can also co-exist with HTML. Therefore, Silverlight can enhance an existing web application.
- Silverlight works its magic through a browser plug-in. When you surf to a web page that includes Silverlight content, this browser plug-in runs, executes the code, and renders that content in a specifically designated region of the page.
- The important part is that the Silverlight plug-in provides a richer environment than the traditional blend of HTML and JavaScript that powers ordinary web pages.
- You can create Silverlight pages that play video, have hardware accelerated 3D graphics, and use vector animations.

From a developer's perspective, the most interesting feature of Silverlight is that it brings the .NET Framework programming model to the client side of your web applications.



- Silverlight is designed to run inside the web pages, so it can run as a browser plug-in. It provides graphical services for rendering bitmaps, vector graphics, high-definition video, and animations

- You can write in C#, or Visual Basic .NET, and use the .NET Framework class library features on the code that runs in the web browser.

- Silverlight user interfaces, themselves use a very similar model to Windows Presentation Foundation(WPF), which is the user interface framework in the full desktop .NET Framework.

- If you know WPF, Silverlight is easy to learn. Silverlight is a much smaller download than .NET. It is roughly a tenth of the size, so only a subset of the class library is present, and various implications have been made to WPF's model.

- Despite the reduced scale, experienced .NET developers will feel instantly at home in Silverlight.

Platforms and Browsers

The platforms and browsers supported by Silverlight are:

Windows

- Silverlight supports Windows, as you would expect of a Microsoft product. It requires Windows XP Service Pack 2 at least or recent versions of Windows.

- The older versions are not fully supported. For example, Silverlight will not run at all on Windows ME, and Windows 2000 has limited support.

- As for the browsers, Silverlight supports Microsoft's own Internet Explorer, of course, and it supports Firefox, and Google Chrome version 4.

- Broadly, Silverlight supports the common web browser plug-in API. It works in a wider range of browsers than the officially supported list.

Mac

- Silverlight supports Mac OS10, although Silverlight version 2 or later only runs on Intel-based Macs.

- On modern Macs, both Firefox and Safari are supported.

Linux

- Microsoft's own Silverlight plug-in does not run on Linux, but the Mono open source project has an offshoot called Moonlight, which is a Silverlight compatible plug-in that runs on Linux.

- Moonlight runs in Firefox, and interestingly has always been able to run in Standalone mode.

- One of the reasons the Mono project decided to build Moonlight in the first place is that they thought Silverlight would be a useful technology for building user interface widgets that run on the desktop.

2. Silverlight – Environment Setup

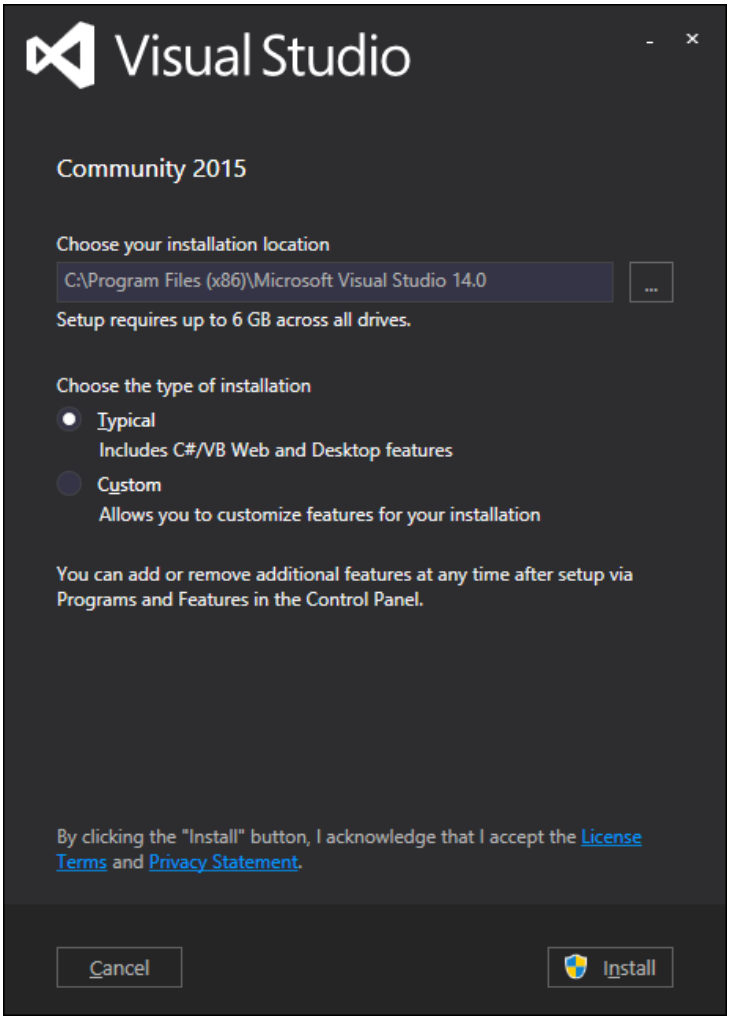
Microsoft provides two important tools for Silverlight application development. They are:

- Visual Studio
- Expression Blend

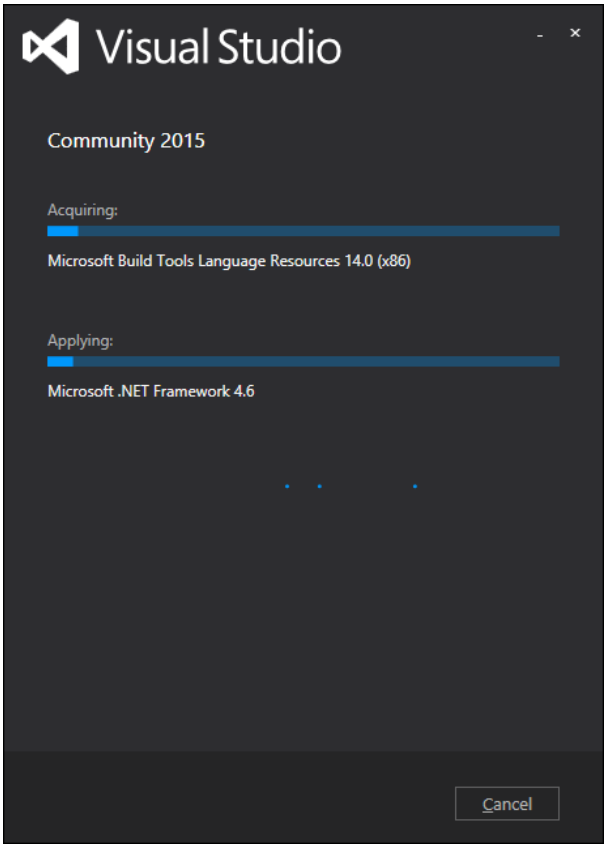
Currently, both tools can create Silverlight projects, but the fact is that Visual Studio is used more by developers while Blend is still used more often by designers. Microsoft provides a free version of visual studio, which can be downloaded from https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx. For this tutorial, we will be mostly using Visual Studio.

Installation

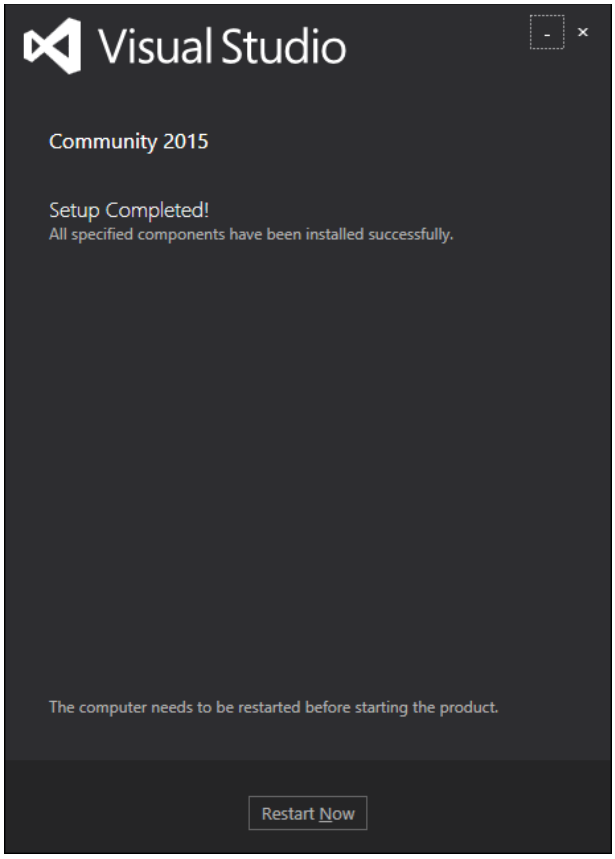
Step 1: Once Silverlight is downloaded, run the installer. The following dialog box will be displayed.



Step 2: Click the **Install** button and it will start the installation process.

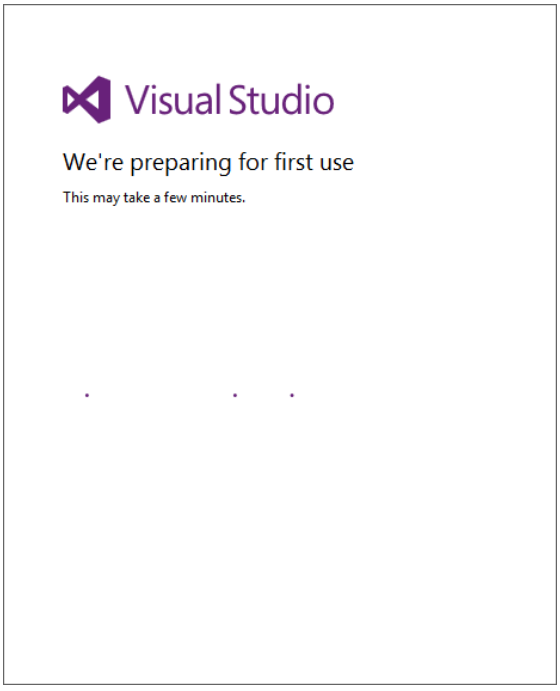


Step 3: Once Sivelight is installed successfully, you will see the following dialog box.

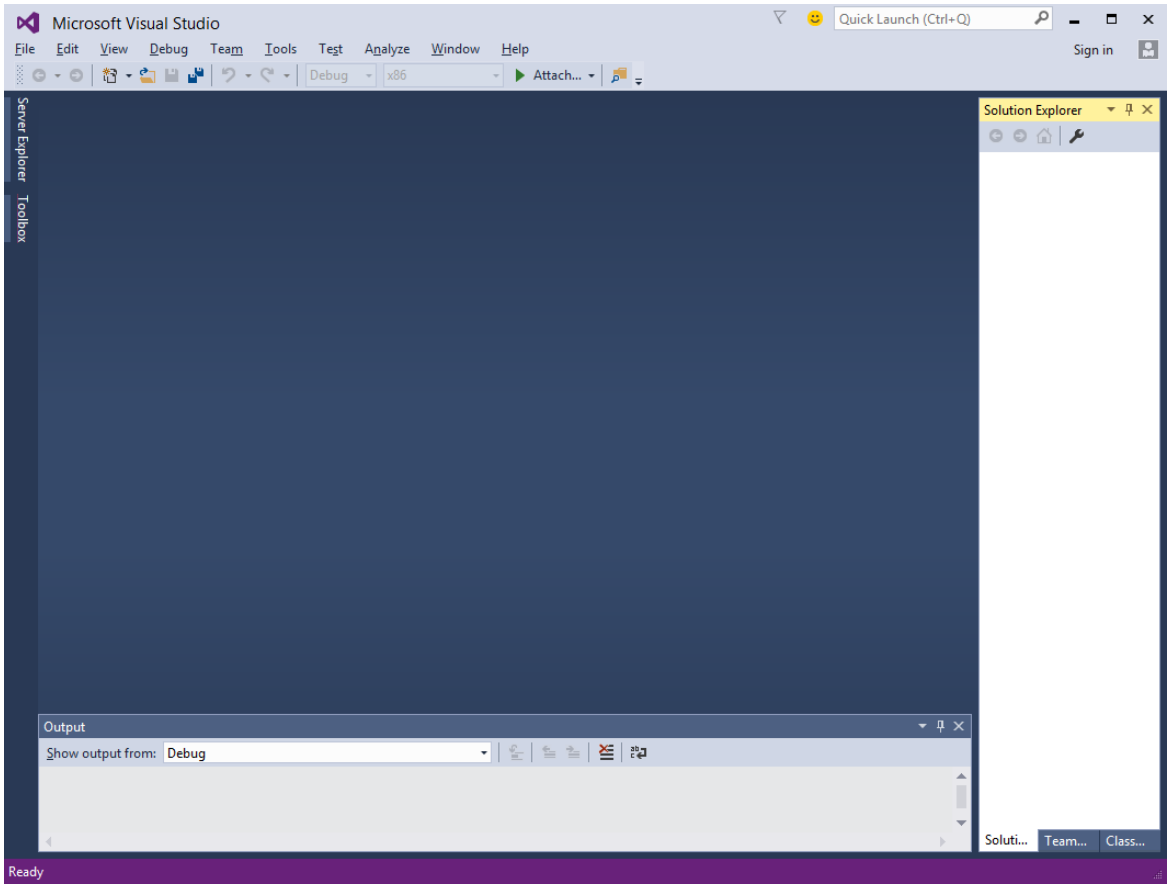


Step 4: Close this dialog box and restart your computer if required.

Step 5: Now open **Visual studio** from the **Start** menu, which will open the dialog box shown below. It will take some time for preparation, while staring for the first time.



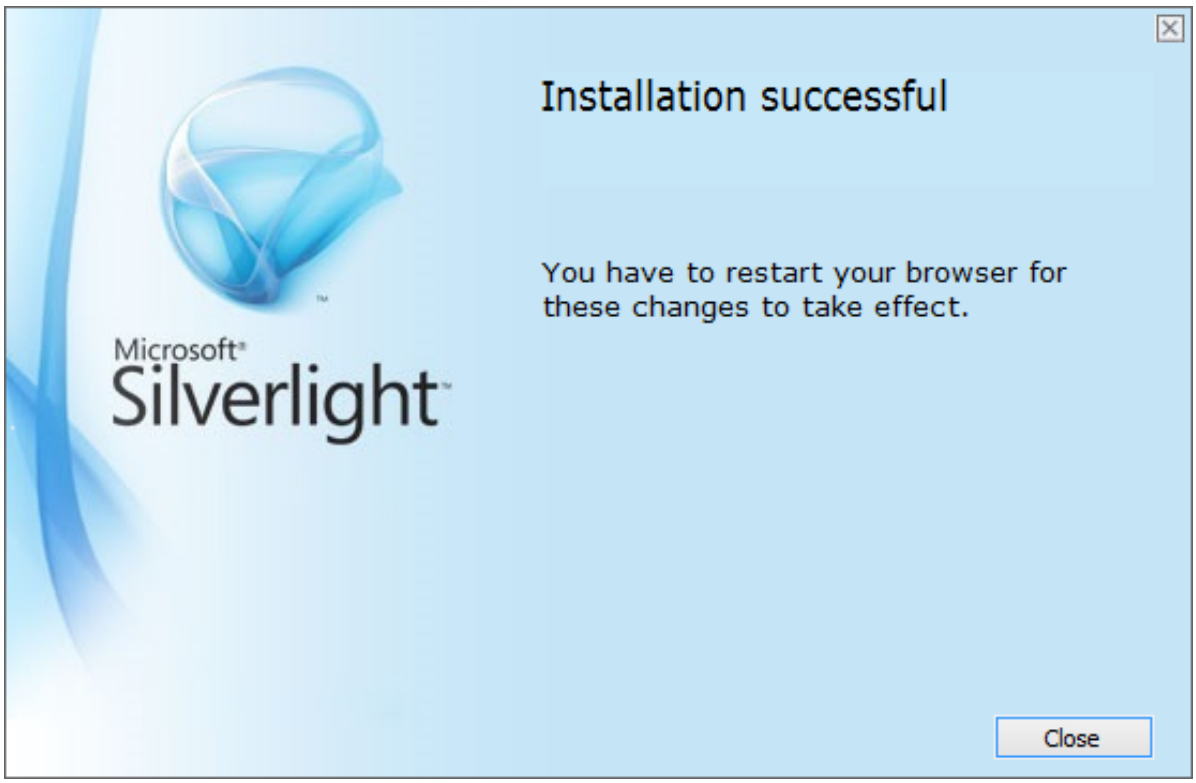
Step 6: Next, you will see the main window of Visual Studio.



Step 7: Now, to start with Silverlight application, you also need to install Silverlight Developer tool on your machine. Download and install the latest Silverlight Developer tool from http://silverlight.dlservice.microsoft.com/download/8/E/7/8E7D9B4B-2088-4AED-8356-20E65BE3EC91/40728.00/Silverlight_Developer_x64.exe

Step 8: Click **Install.** It will take some time for installation.

Step 9: Once the installation is complete, you will see the following message.



Step 10: Now you are ready to build your first Silverlight application. Click **Close**.

3. Silverlight – Getting Started

In this chapter, we will look at a working example of Silverlight. We need two things:

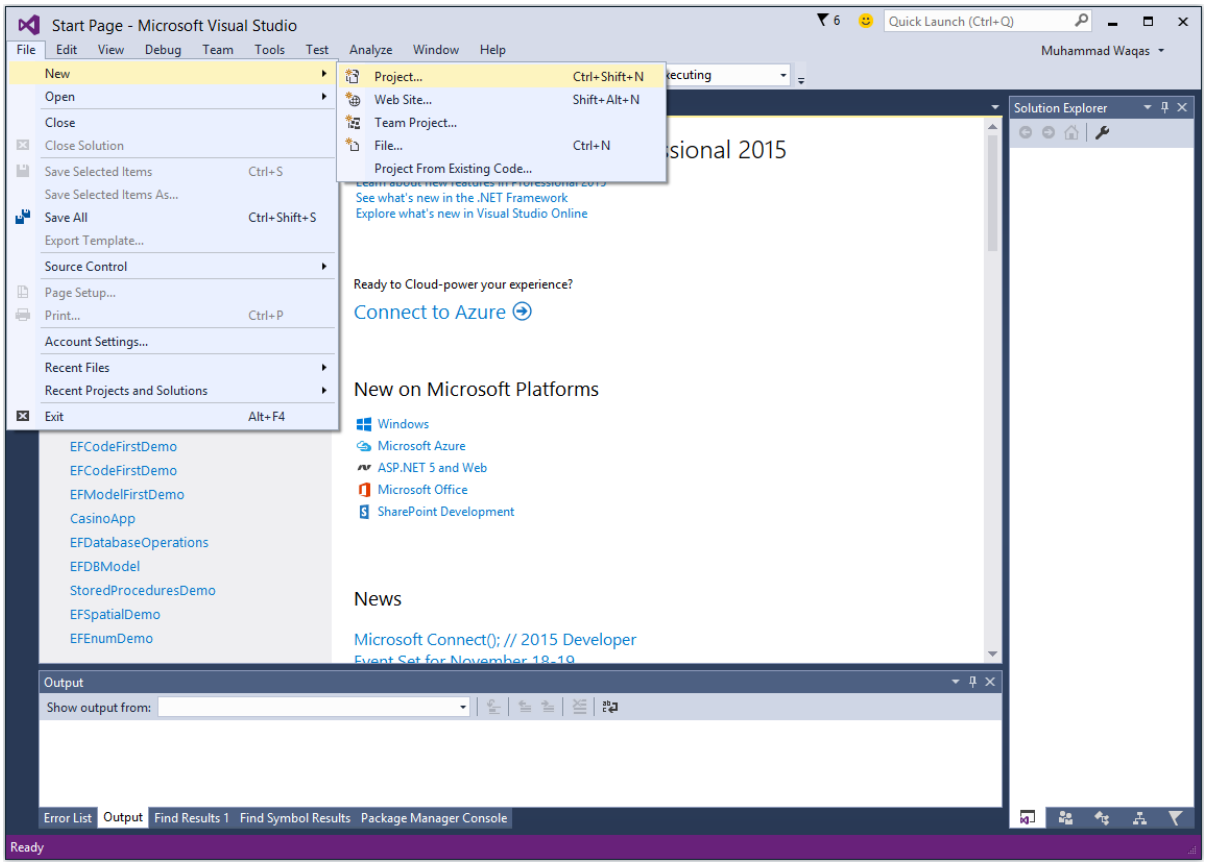
- First, we require a web page. Silverlight is intended for rich internet applications, It is designed to run inside of a web browser as part of a web page. The page needs to incorporate a suitable tag to load the Silverlight plug-in. It can also include the logic to detect whether Silverlight is installed, and can provide some fallback user interface, when it is absent.

- The second thing we need is the Silverlight content itself. This tutorial will focus on the .NET programming model for Silverlight. We will create a compiled Silverlight application containing a mixture of XAML, the mockup language we use to define Silverlight user interfaces, and .NET code written in C#.

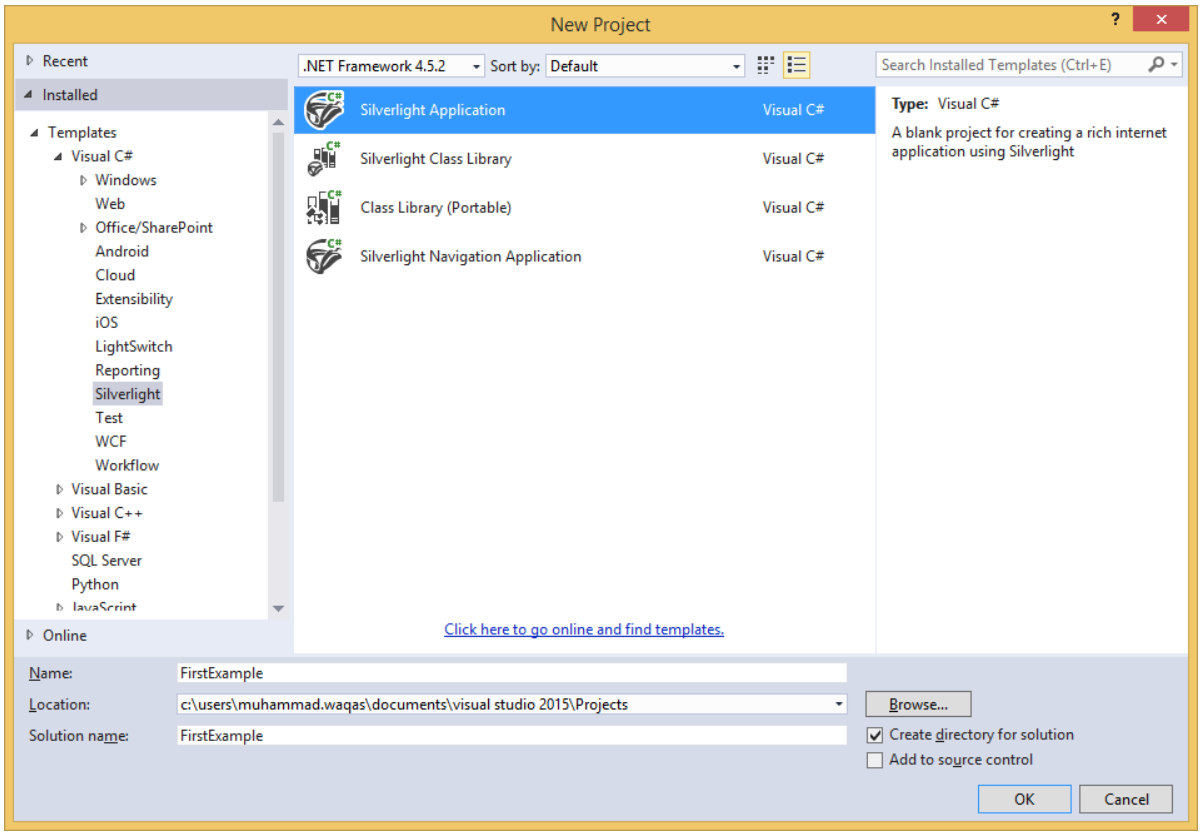
Create a Web-page

The easiest way to start using Silverlight is to create an ordinary website with HTML pages and no server side code. Let us look at a very simple example.

Step 1: Open **Visual Studio.** Click the **File** menu, point to **New** and then click **Project.**



Step 2: A **New Project** dialog box will open. Under **Templates,** select **Visual C#** and then **click Silverlight.** In the right pane, choose Silverlight Application.

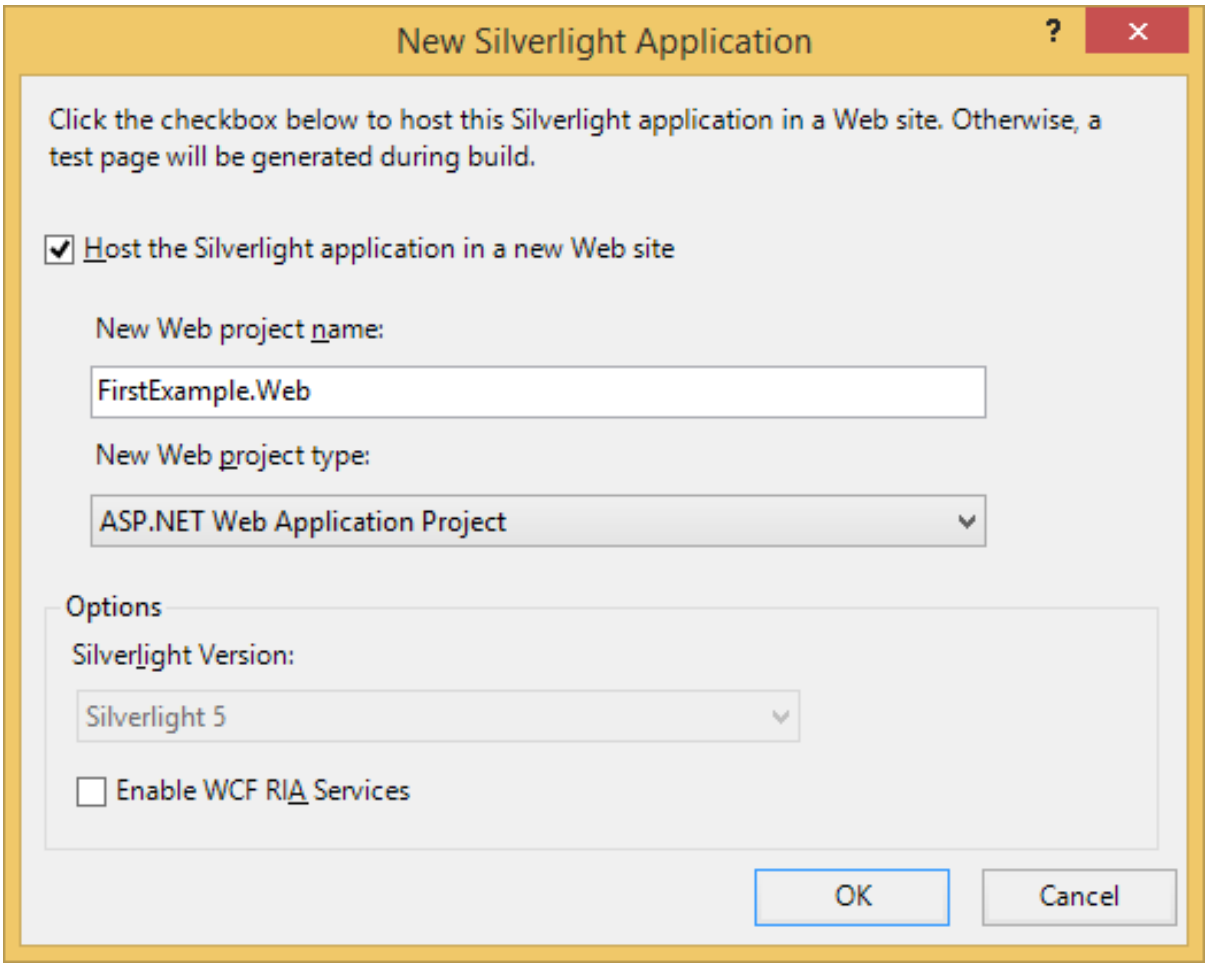


Enter a project name and a location on your hard drive to save your project and then click **OK** to create the project.

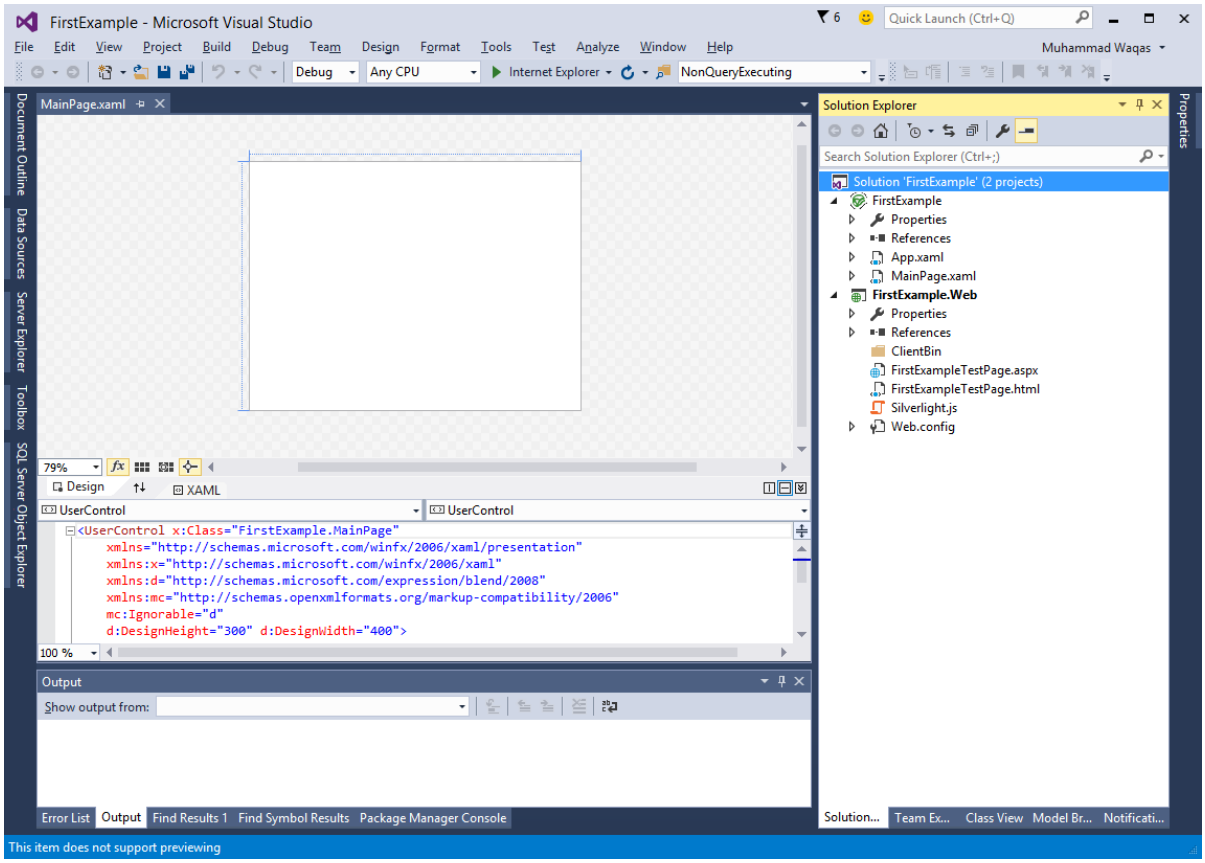
The Silverlight project itself is just going to build the Silverlight content, and that content is just one asset amongst many that are going to make up the whole web application.

Click **OK.**

Step 3: Check the **Host the Silverlight application checkbox.** The default is an ASP.NET Web Application Project.



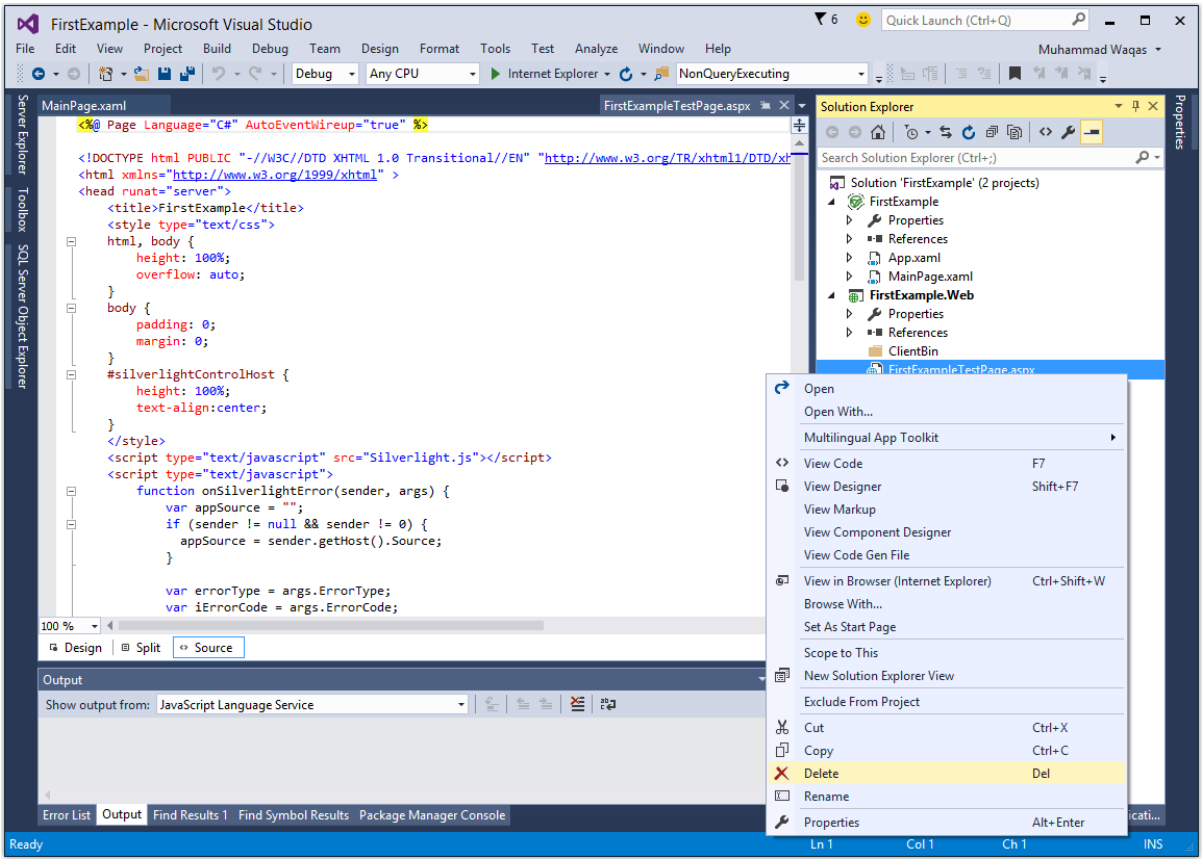
Step 4: MS-Visual Studio has created two projects, the Silverlight project and an ASP.NET web application. Now, we do need an ASP.NET web application. You can see this in the **Solution Explorer** window as shown below.



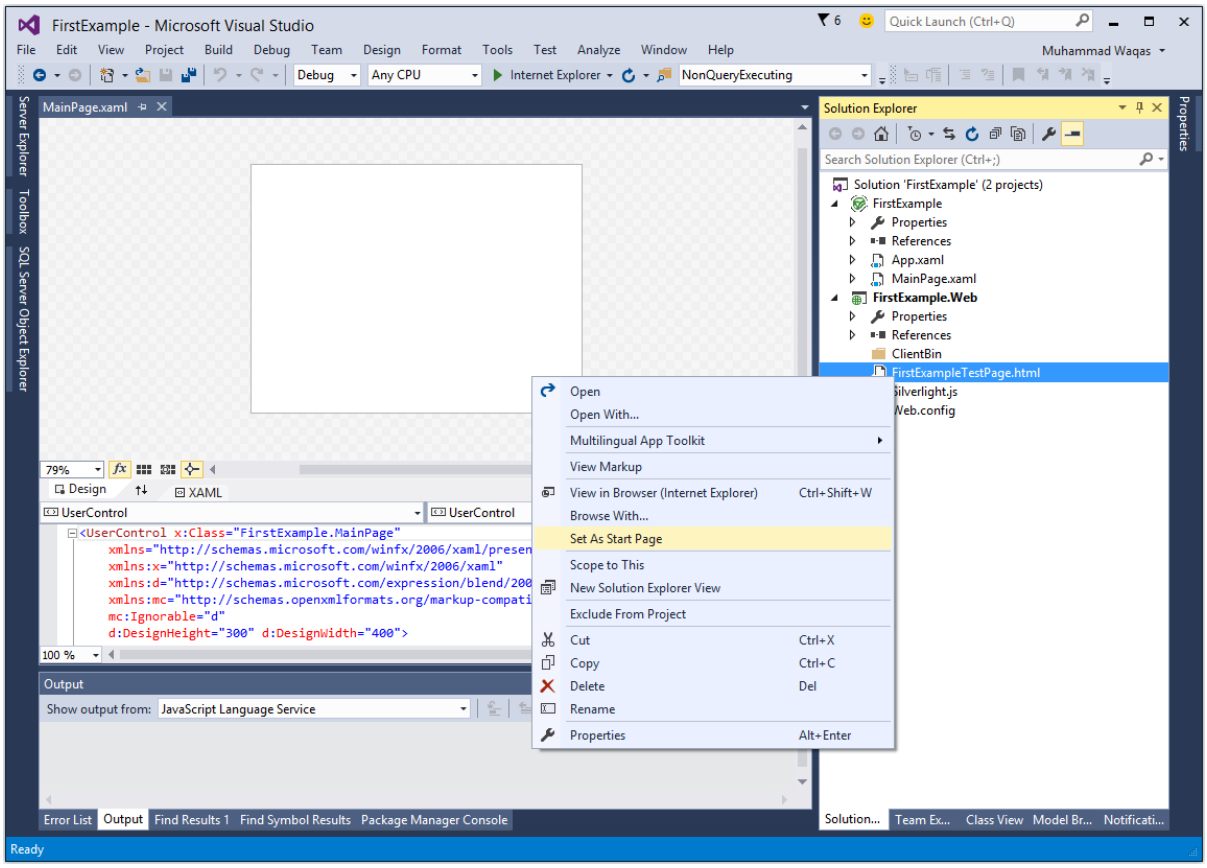
Anything that can serve up the content via HTTP will do but this is **Visual Studio**, and it understands the ASP.NET web technology, so that is what it gives us.

To demonstrate that Silverlight does not depend on any particular server-side technology, let us delete this **.aspx** file, leaving just the plain static HTML file.

Step 5: Right-click FirstExampleTestpage.aspx. From the list of options, click **Delete**.



Step 6: Set **FirstExampleTestPage.html** as the **Start** page.



The **MainPage.xaml** file defines the user interface for Silverlight content. Either you can write XAML code directly or you can also use **Toolbox** to drag and drop different UI elements.

Step 7: Given below is a simple code in **MainPage.xaml** in which a **Button** and a **TextBlock** are defined inside the **StackPanel**.

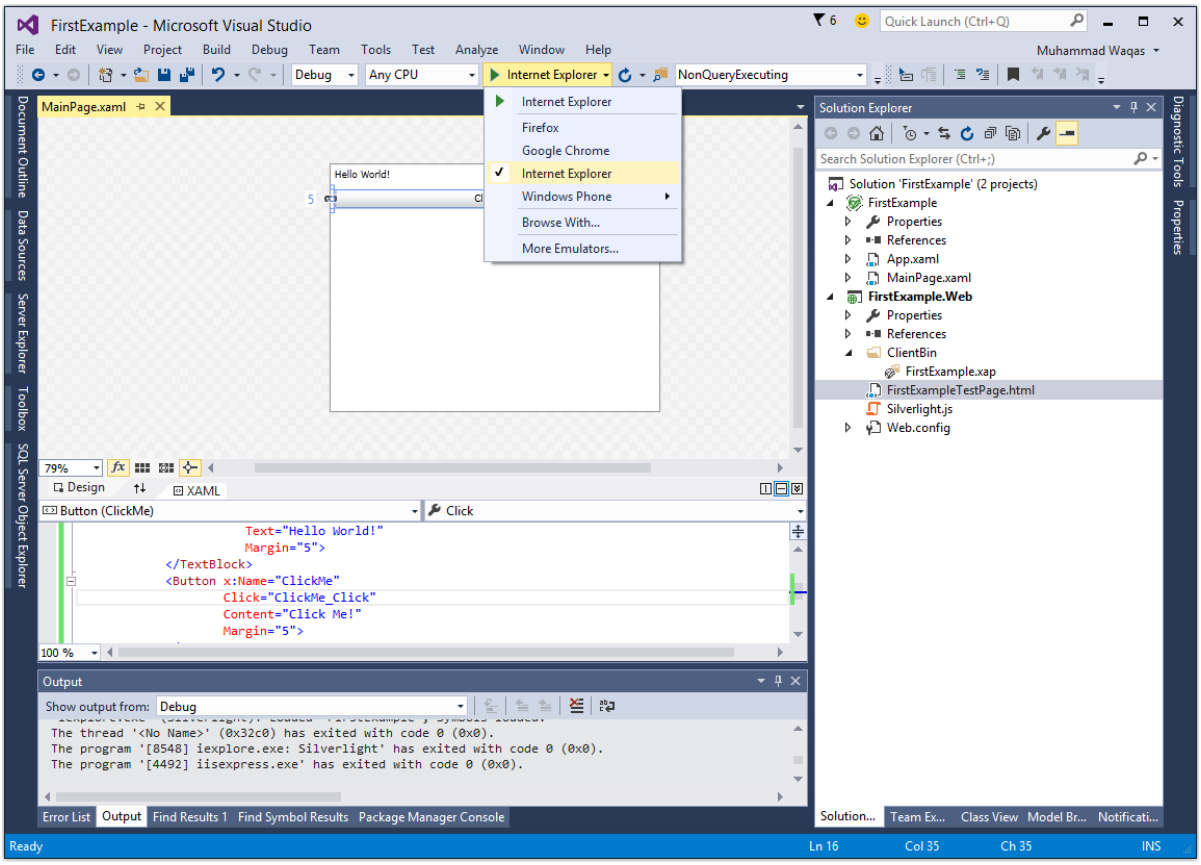
```
<UserControl x:Class="FirstExample.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel>
            <TextBlock x:Name="TextMessage"
                       Text="Hello World!"
                       Margin="5">
```

```
            </TextBlock>
            <Button x:Name="ClickMe"
                    Click="ClickMe_Click"
                    Content="Click Me!"
                    Margin="5">
            </Button>
        </StackPanel>
    </Grid>
</UserControl>
```

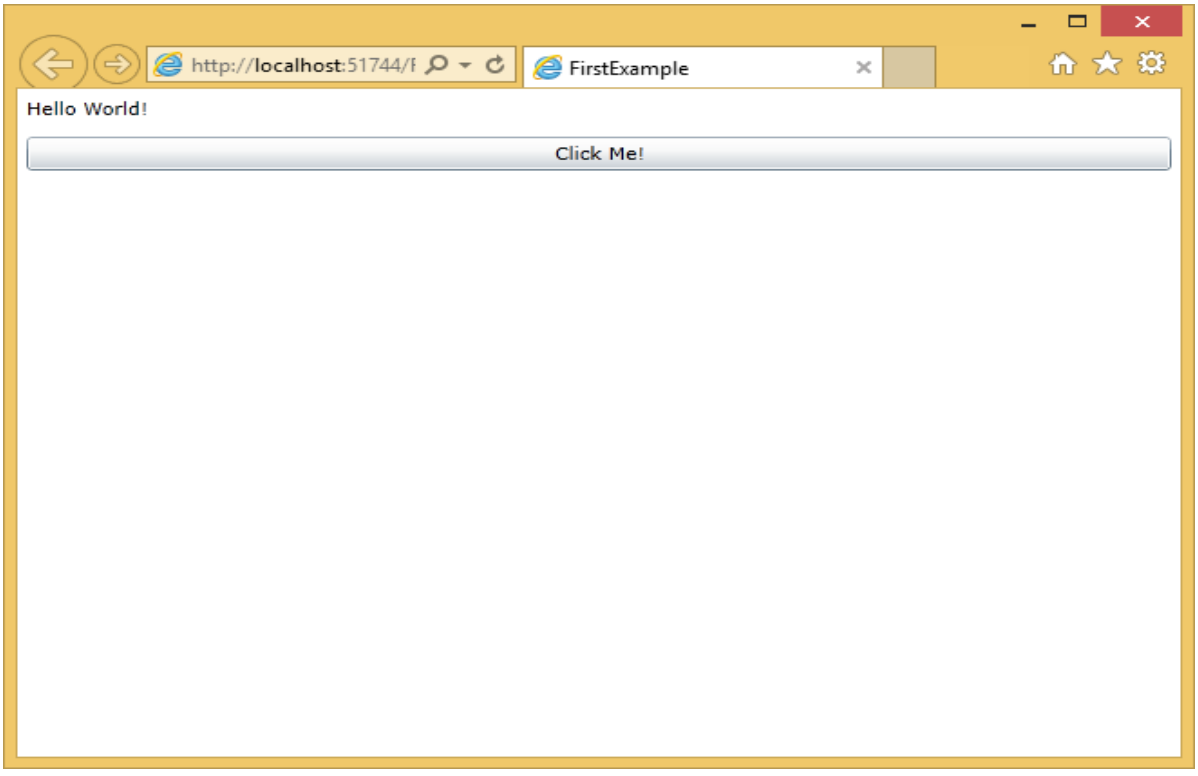
Step 8: This example assumes that you have created an event-handling method named **ClickMe_Click**. Here is what it looks like in the **MainPage.xaml.cs** file.

```
using System.Windows;
using System.Windows.Controls;

namespace FirstExample
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void ClickMe_Click(object sender, RoutedEventArgs e)
        {
            TextMessage.Text = "Congratulations! you have created your first
Silverlight Applicatoin";
        }
    }
}
```

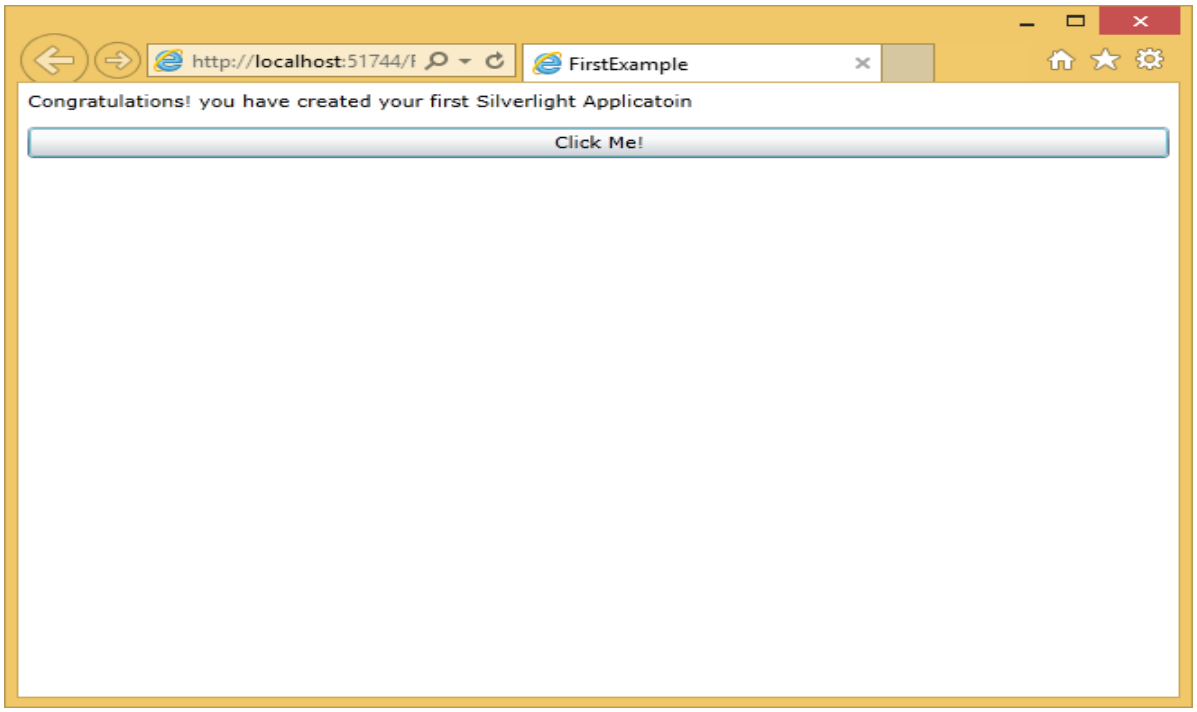
Step 9: A Silverlight application can be run on any installed browsers.



Step 10: When the above code is compiled and executed, you will see the following webpage.



Step 11: Now, when you click the **Click Me** button, it will update the text in the **TextBlock** as shown below.



We recommend you to execute the above example by adding some more UI elements.

4. Silverlight – XAML Overview

One of the first things you will encounter when working with Silverlight is XAML. XAML Stands for Extensible Application Markup Language. It is a simple and declarative language based on XML.

- In XAML, it is very easy to create, initialize, and set properties of an object with hierarchical relations.
- It is mainly used for designing GUI.
- It can be used for other purposes as well, for example, to declare workflow in a Workflow foundation.

Basic Syntax

When you create a new Silverlight project, you will see some of the XAML code by default in **MainPage.xaml** as shown below.

```
<UserControl x:Class="FirstExample.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"
    d:DesignHeight="300" d:DesignWidth="400">
    <Grid x:Name="LayoutRoot" Background="White">


    </Grid>
</UserControl>
```

You can see that the XAML file given above mentions different kinds of information; all of them are briefly described in the table given below.

Information	Description
`<UserControl`	Provides the base class for defining a new control that encapsulates the existing controls and provides its own logic.
`x:Class="FirstExample.MainPage"`	It is a partial class declaration, which connects the markup to that partial class code behind, defined in it.

`xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"`	Maps the default XAML namespace for Silverlight client/framework.
`xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"`	XAML namespace for XAML language, which maps it to x: prefix.
`xmlns:d="http://schemas.microsoft.com/expression/blend/2008"`	XAML namespace is intended for designer support, specifically designer support in the XAML design surfaces of Microsoft Visual Studio and Microsoft Expression Blend.
`xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"`	Indicates and supports a markup compatibility mode for reading XAML.
`>`	End of object element of the root.
`<Grid>` `</Grid>`	These are the starting and closing tags of an empty grid object.
`</UserControl>`	Closing the object element.

Syntax rules for XAML is almost similar to those of XML. If you look at an XAML document, you will notice that actually it is a valid XML file. Its vice versa is not true, because in XML, the value of the attributes must be a string while in XAML it can be a different object which is known as Property element syntax.

- Syntax of an Object element starts with a left angle bracket (<) followed by the name of an object, e.g. Button.
- The Properties and attributes of that object element are defined.
- The Object element must be closed by a forward slash (/) followed immediately by a right angle bracket (>).

Example of a simple object with no child element is shown below.

```
<Button/>
```

Example of an object element with some attributes:

```
<Button Content="Click Me"
        Height="30"
        Width="60"/>
```

Example of an alternate syntax to define the properties (Property element syntax):

```
<Button
        <Button.Content>Click Me</Button.Content>
        <Button.Height>30</Button.Height>
        <Button.Width>60</Button.Width>
</Button>
```

Example of an Object with Child Element: StackPanel contains Textblock as child element.

```
<StackPanel Orientation="Horizontal">
            <TextBlock Text="Hello"/>
</StackPanel>
```

Why XAML in Silverlight

XAML was not originally invented for Silverlight. It came from WPF, the Windows Presentation Foundation. Silverlight is often described as being a subset of WPF. This is not strictly true, as Silverlight can do some things that WPF cannot. Even where the functionality overlaps, the two are slightly different in the details.

- It is more accurate to say that WPF and Silverlight are very similar in many respects. Despite the differences, it is still informative to look at the XAML feature Silverlight has borrowed from WPF. For example, Silverlight offers graphics primitives for bitmaps and scalable shapes.
- It also provides elements for rendering video and audio.
- It has simple formatted text support, and you can animate any element. If you know WPF, this feature set will be familiar to you.
- One important point, you cannot take WPF XAML and use it in Silverlight.
- Although there are similarities, you will also find numerous small differences.

XAML & Code Behind

XAML defines the appearance and structure of a user interface. However, if you want your application to do anything useful when the user interacts with it, you will need some code.

- Each XAML file is usually associated with a source code file, which we refer to as the code behind. Various Microsoft Frameworks use this term.

- The code behind will usually need to use elements defined in the XAML, either to retrieve information about user input, or to show information to the user.

- In the XAML code given below, **TextBlock** and **Button** are defined. By default, when the application is run, it will show a text **"Hello World!"** on the web page and a button.

```
<UserControl x:Class="FirstExample.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"
    d:DesignHeight="300" d:DesignWidth="400">

    <Grid x:Name="LayoutRoot" Background="White">
        <StackPanel>
            <TextBlock x:Name="TextMessage"
                       Text="Hello World!"
                       Margin="5">
            </TextBlock>
            <Button x:Name="ClickMe"
                    Click="ClickMe_Click"
                    Content="Click Me!"
                    Margin="5">
            </Button>
        </StackPanel>
    </Grid>
</UserControl>
```

- The code behind can access any element that is named with the **x:Name** directive.

- Named elements become available through fields in the code behind, allowing the code to access these objects and their members in the usual way.

- The **x:Prefix** signifies that the name is not a normal property.

- **x:Name** is a special signal to the XAML compiler that we want to have access to this object in the code behind.

Given below is the button-click event implementation in which the **TextBlock** text is updated.

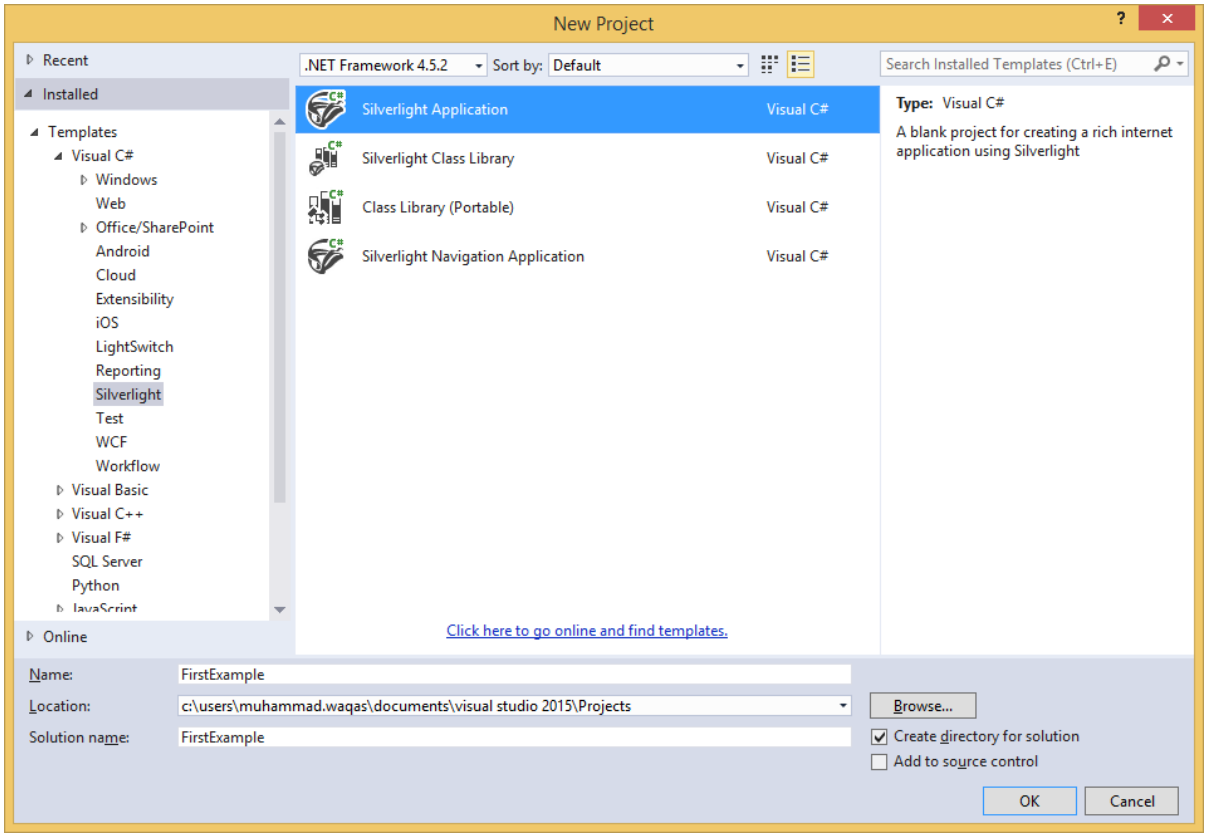
```
using System.Windows;
using System.Windows.Controls;

namespace FirstExample
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            InitializeComponent();
        }

        private void ClickMe_Click(object sender, RoutedEventArgs e)
        {
            TextMessage.Text = "Congratulations! you have created your first
Silverlight Applicatoin";
        }
    }
}
```

- XAML is not the only way to design the UI elements. It is upto you to either declare objects in XAML or declare/write in a code.
- XAML is optional, but despite this, it is the heart of **Silverlight** design.
- The goal with XAML coding is to enable the visual designers to create the user interface elements directly. Therefore, **Silverlight** aims to make it possible to control all the visual aspects of the user interface from mark-up.

5. Silverlight – Project Types

If you create a new project in Visual Studio, you will see four types of project in the right pane of the dialog box. They are:

- Silverlight Application
- Silverlight Class Library
- Class Library (Portable)
- Silverlight Navigation Application



- The first two, **Silverlight Application** and **Silverlight Class Library**, are straightforward enough. These are analogous to executables in DLLs in the world of classic Windows applications. Both build DLLs because of how Silverlight applications are deployed.

- Conceptually, a Silverlight Application project builds a program, which can be run, while the Class Library project builds a library designed to be incorporated into other applications.

- You can build a class library if you are planning to build multiple applications, and want to reuse the common code. If you are planning to sell the controls that other people will use in their applications, again a library is the thing to build.

- The other project types are a little less obvious, so we will look at those in detail later in this chapter.

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**