**Microsoft Office SharePoint Server (MOSS) for Developers**

**(Part 1)**

**Topic 5: Silverlight and Web Parts**



[**http://msdn.microsoft.com/rampup**](http://msdn.microsoft.com/rampup)

**Introduction**

Microsoft Silverlight is a cross-browser, cross-platform implementation of the .NET Framework for building and delivering the next generation of media experiences and rich interactive applications (RIA) for the Web. Silverlight unifies the capabilities of the server, the Web, and the desktop, of managed code and dynamic languages, of declarative and traditional programming, and the power of Windows Presentation Foundation (WPF).

Silverlight enables you to create a state-of-the-art application that has the following features:

* It is a cross-browser, cross-platform technology. It runs in all popular Web browsers, including Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari, and on Microsoft Windows and Apple Mac OS X.
* It provides a consistent experience no matter where it runs.
* It is supported by a very small download that installs in seconds.
* It streams video and audio. It scales video quality to everything from mobile devices to desktop browsers to 720p HDTV video modes.
* It includes compelling graphics that users can manipulate—drag, turn, zoom—directly in the browser.
* It reads data and updates the display, but it doesn't interrupt the user by refreshing the whole page.

Web developers and graphics designers can create Silverlight-based applications in a variety of ways. You can use Silverlight markup to create media and graphics, and manipulate them with dynamic languages and managed code. Silverlight also enables you to use professional-quality tools like Visual Studio for coding and Microsoft Expression Blend for layout and graphic design.

Silverlight combines multiple technologies into a single development platform that enables you to select the right tools and the right programming language for your needs. Silverlight offers you the following features:

* WPF and XAML. Silverlight includes Windows Presentation Foundation (WPF) technology, which greatly extends the elements in the browser for creating UI. WPF lets you create immersive graphics, animation, media, and other rich client features, extending browser-based UI beyond what is available with HTML alone. Extensible Application Markup Language (XAML) provides a declarative markup syntax for creating WPF elements. See [Creating User Interfaces with Silverlight](http://msdn.microsoft.com/en-us/library/cc189050(VS.95).aspx) for more information.
* Extensions to JavaScript. Silverlight provides extensions to the universal browser scripting language that provide powerful control over the browser UI, including the ability to work with WPF elements. See [Silverlight 1.0 - Development with JavaScript](http://msdn.microsoft.com/en-us/library/bb980125(VS.95).aspx) for more information.
* Cross-browser, cross-platform support. Silverlight runs the same on all popular browsers (on any platform). You can design and develop your application without having to worry about which browser or platform your users have. See [Creating and Deploying Silverlight Applications](http://msdn.microsoft.com/en-us/library/cc645070(VS.95).aspx) for more information.
* Integration with existing applications. Silverlight integrates seamlessly with your existing JavaScript and ASP.NET AJAX code to complement functionality you have already created. See [Integrating Silverlight with ASP.NET Web Pages](http://msdn.microsoft.com/en-us/library/cc221400(VS.95).aspx) for more information.
* Access to the .NET Framework programming model and to associated tools. You can create Silverlight-based applications using dynamic languages such as managed JScript and IronPython as well as languages such as C# and Visual Basic. You can use development tools such as Visual Studio to create Silverlight-based applications. See [Common Language Runtime and Base Class Library in Silverlight](http://msdn.microsoft.com/en-us/library/cc221412(VS.95).aspx) and [Dynamic Languages in Silverlight 2](http://msdn.microsoft.com/en-us/library/cc189024(VS.95).aspx) for more information.
* LINQ. Silverlight includes language-integrated query (LINQ), which enables you to program data access using intuitive native syntax and strongly typed objects in .NET Framework languages. See [Parsing XML Data in Silverlight](http://msdn.microsoft.com/en-us/library/cc188996(VS.95).aspx) for more information.
* If you already use ASP.NET, you can integrate Silverlight with the server and client capabilities of ASP.NET that you are familiar with. You can create server-based resources in ASP.NET and use the AJAX capabilities of ASP.NET to interact with server-based resources without interrupting the user.
* For more details, see [Silverlight Architecture](http://msdn.microsoft.com/en-us/library/bb404713(VS.95).aspx).

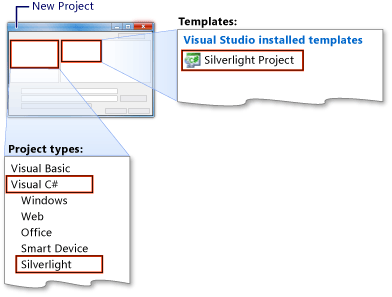
Silverlight-based applications run in the browser. Silverlight makes sure that you can run your applications in all modern browsers, without having to create browser-specific code.

To run a Silverlight-based application, users require a small plug-in in their browser. The plug-in is free. If users do not already have the plug-in, they are automatically prompted to install it. The download and installation take seconds and require no interaction from the user except permission to install.

You can create Silverlight-based applications using skills that you already have and tools that you are already familiar with. You can create Web pages that use both HTML and WPF elements with Silverlight. Like HTML, XAML enables you to create UI for your Web-based application with declarative syntax, with the difference that XAML provides significantly more powerful elements.

### Using Visual Studio

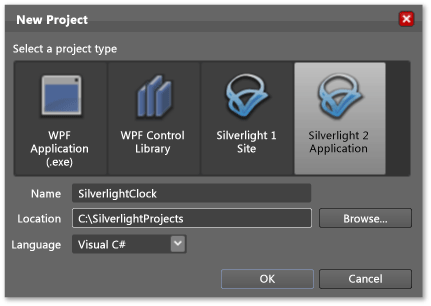
You can create Silverlight-based applications using [Visual Studio 2008](http://go.microsoft.com/fwlink/?LinkId=94863). (Silverlight support for Visual Studio 2008 is available as a plug-in for the IDE.) Visual Studio provides the same tools for Silverlight, such as powerful editors, project management, debugging, and so on, that it already provides for creating applications with the .NET Framework.



### Using Expression Blend

Designers can take advantage of the end-to-end capabilities of [Microsoft Expression Blend](http://go.microsoft.com/fwlink/?LinkId=94863) to design, build, and deliver Silverlight-based applications. Expression Blend has all the capabilities needed to create graphics and media assets, design interactivity, and produce experiences. Expression Blend can generate XAML output, meaning that programmers who use Visual Studio and designers who use Expression Blend can share the same files.

**Creating a Silverlight-based application with Microsoft Expression**



 Resources for Getting Started



Some of the resources in the following list provide information for both versions of Silverlight; other resources are specific to either 1.0 or 2. Note that the term "Silverlight" without a version number applies to both versions.

* [Getting Started with Silverlight](http://msdn.microsoft.com/en-us/library/bb404703(VS.95).aspx): Provides an introduction to developing applications with Silverlight 2.
* [Silverlight Architecture](http://msdn.microsoft.com/en-us/library/bb404713(VS.95).aspx): Provides an overview of the architecture of Silverlight.
* [Silverlight 1.0 - Development with JavaScript](http://msdn.microsoft.com/en-us/library/bb980125(VS.95).aspx): Provides information for developing with Silverlight 1.0, which supports the XAML markup language and JavaScript for creating applications. Most of the conceptual material contained in this section (animations, graphics, XAML, and so on) is also useful to Silverlight 2 developers.
* [Silverlight 2 Beta 2 - Development with the .NET Framework](http://msdn.microsoft.com/en-us/library/bb404700(VS.95).aspx): Provides information for developing with Silverlight 2, which includes .NET Framework features.
* [Silverlight Technical Articles](http://go.microsoft.com/fwlink/?LinkId=98682): Provides additional MSDN articles about Silverlight.

**Introduction to Creating a Silverlight Web Part**

This article describes how to create a very simple Microsoft Silverlight Web Part in ASP.NET that hosts a canvas (XAML), which can have events mapped to ECMAScript (JScript, JavaScript). We develop the Silverlight Web Part by using Microsoft Visual Studio 2005. The Silverlight Web Part uses Microsoft Silverlight only on the client side, and is based on the [Microsoft Silverlight 1.0 Runtime](http://www.microsoft.com/silverlight/resources/install.aspx). You can then use the Silverlight Web Part in Microsoft Office SharePoint Server 2007 and Windows SharePoint Services 3.0 to display a canvas (XAML).

**Prerequisites**

* To successfully create and use the Silverlight Web Part, you must understand the following: Microsoft Silverlight, XAML, ECMAScript (JScript or JavaScript code), and ASP.NET control development.
* To be able to test the solution, you must have the latest version of the [Silverlight plug-in](http://www.microsoft.com/silverlight/resources/install.aspx) installed on the client computer.

To view the Page.xaml.js file and Default\_html.js file referred to in this article, see [Complete Page.xaml.js File and Default\_html.js File](http://msdn.microsoft.com/en-us/library/cc627341.aspx#MOSS2007SilverlightWebPart_PagexamljsandDefaulthtmljsFiles).

**Creating the Silverlight Web Part**

The first part of our task is to create a solution in Visual Studio 2005, and then create the Silverlight Web Part.

**To create the Silverlight Web Part**

1. In Microsoft Visual Studio 2005, open a Visual C# Class Library project.
2. Remove Class1.cs.
3. Add the basic SilverlightWebpart.cs to the project.
4. In the project references, add a reference to **System.Web**.
5. Add private properties, which set the path to .js files and the .xaml file. You can add properties to customize the Silverlight Web Part to suit your specific needs.
6. Register the following three EcmaScript (JScript, JavaScript) files in the Web Part; these files are part of any standard Silverlight project or control:
   * **silverlight.js**   A standard file that detects whether the Silverlight client plug-in is installed on the client computer. If not, the file shows an icon of a location from where you can install the plug-in.
   * **Default\_html.js**   The same as createsilverlight.js; this file contains the **createSilverlight** function that we need to modify with the code shown in code listing 7.
   * **Page.xaml.js**   Contains the event listeners.
7. Add the code that generates the HTML <div> tag where the Silverlight object will be created in your Web Part (for more information, see [Creating Custom Web Parts](http://quickstarts.asp.net/QuickStartv20/aspnet/doc/webparts/custom.aspx)).
8. Create a Default\_html.js file and copy the following code into it to modify the **createSilverlight** function.

**Deploying the Silverlight Web Part on Office SharePoint Server 2007**

You can deploy the Silverlight Web Part on your server running Office SharePoint Server 2007 by using the following steps.

**To deploy the Silverlight Web Part**

1. Ensure latest version of the [Silverlight plug-in](http://www.microsoft.com/silverlight/install.aspx) is installed on the client computer. In the case that the plug-in is not installed, when the page with the Web Part loads for the first time, it displays a link that will show a location from which to download and install the plug-in. (This will happen automatically and no code is required.)
2. Copy the *SomeName*.dll (in our sample it will be SilverlightWebPart.dll) into the \_app\_bin directory of your SharePoint application; for example, C:\Inetpub\wwwroot\wss\VirtualDirectories\80\\_app\_bin. The .dll file can be signed and copied to the global assembly cache also.
3. Copy the files default\_html.js, Page.xaml, Page.xaml.js and SilverLight.js into the top-level site (root) of the SharePoint Web application (for example, C:\Inetpub\wwwroot\wss\VirtualDirectories\80\\_app\_bin). We can copy these files at other locations also. Change the Silverlight Web Part property **JSPath** to the path where you have copied the .js files.
4. Add a <SafeControl/> element to the following entry to the web.config file with references to the SilverLight assembly, NameSpace, and mark it as safe.
5. For strong-named assemblies, the **version**, **culture**, and **publickeytoken** attributes will be added to the **SafeControl** element.
6. Go to the Web Part gallery from the site collection administration page. Click **New**, select **BasicSilverLightWebPart**, and then click **populate gallery**.
7. Go to any sample SharePoint site, and add the Silverlight Web Part to a page. Modify custom properties, such as the Silverlight control height and width.
8. Ensure that you supply the complete path to the Page.xaml file in the Silverlight Web Part’s custom properties (for example, /Page.xaml).
9. Save or publish the page.