

# The Best of Both Worlds

## Introducing the ASP.NET Silverlight Controls

Silverlight 2 has been released. For you as an ASP.NET developer, is that ho-hum or hurrah? They both create visible programs that run in a browser, so are they competing or complementary technologies? Are they like oil and water — interfering with each other and best used separately? Or are they like peanut butter and jelly — both good, but better together?

It turns out that although some situations are best with one or the other, many scenarios benefit from both technologies used together. And as we'll see in this article, using them together couldn't be easier.

### Getting the ASP.NET Silverlight Controls

There are two new ASP.NET server controls in the ASP.NET toolbox that open up possibilities previously difficult or impossible to do on an ASP.NET page:

- The ASP.NET Silverlight control lets you embed compiled Silverlight content into an ASP.NET Web page.
- The ASP.NET MediaPlayer control provides an easy way to add audio or video playback capabilities to an ASP.NET Web page.

To get these controls, you need to download and install the Microsoft Silverlight Tools for Visual Studio 2008 Service Pack 1 ([go.microsoft.com/fwlink/?LinkId=128134](http://go.microsoft.com/fwlink/?LinkId=128134)). Be sure to uninstall any earlier CTP or beta versions before installing the Silverlight Tools for Visual Studio 2008 SP1.

This also installs the Silverlight 2 runtime you'll need to test your project, as well as a Silverlight 2 project template for Microsoft Expression Blend 2 Service Pack 1, although Blend is not required to create Silverlight content.

After you install the Silverlight tools, two new controls are available in the Toolbox in Visual Studio 2008, as shown in Figure 1.

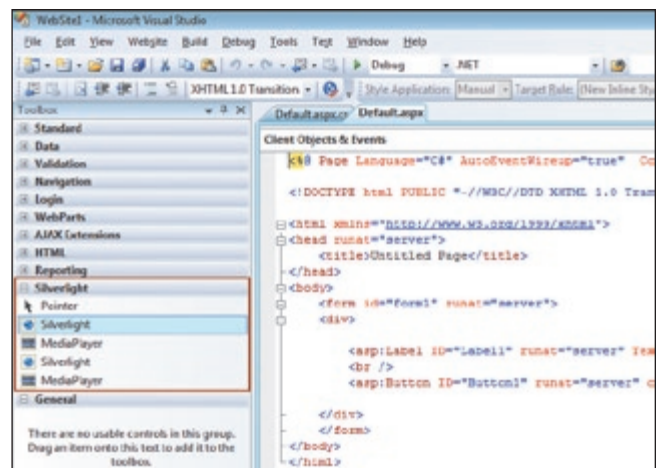


Figure 1: Toolbox with Silverlight controls.

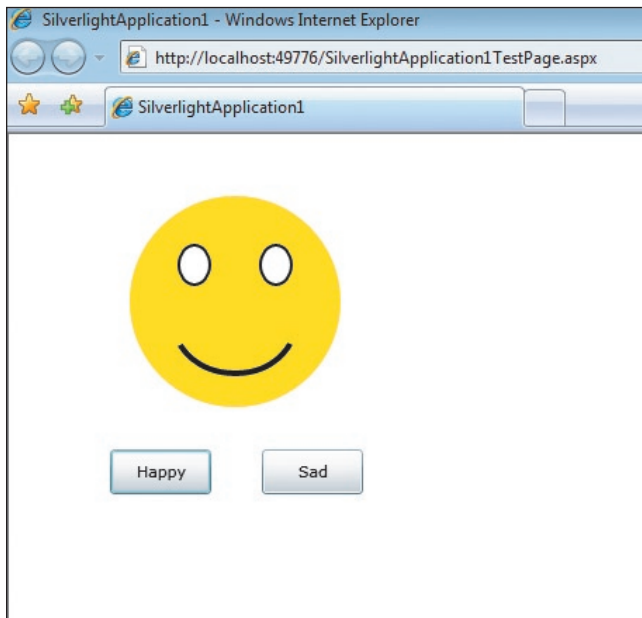
### Adding the Silverlight Control to a Web Page

With the ASP.NET Silverlight control, you can easily integrate Silverlight content into any ASP.NET Web page.

Let's add a Silverlight application to an existing ASP.NET page. To follow along on your computer, download the accompanying files for this article (see end of article for download details). All you need for this procedure is the MoodIndicator.xap file.

Compiling a Silverlight project produces a .xap file as the output. This file combines all the scripts and resources and assemblies into a single file. This file (in zip format) is what the Silverlight control uses to display Silverlight content on an ASP.NET page.

First, create a new Web site or Web project with Visual Studio 2008 SP1. You also can use Visual Web Developer 2008 Express Edition with SP1. On the Default.aspx page, add a ScriptManager control between the opening and closing tags for the form element. This control is necessary because the



**Figure 2:** Silverlight content in an ASP.NET Web page.

Silverlight server control creates an AJAX client control in the HTML sent to the browser. The ScriptManager control is located in the AJAX Extensions area of the Toolbox. Next, drag a Silverlight control from the Toolbox below the ScriptManager control and before the closing form tag. The Silverlight control is located in the Silverlight area of the Toolbox, as shown in Figure 1. This adds a Register directive to the page for the Silverlight assembly and also sets references to the Silverlight DLL added to the Bin folder. Then right-click on the Web site in Solution Explorer. Choose Add an Existing Item, then navigate to the MoodIndicator.xap file in the source files that accompany this article and add it to the project. Set the Source property of the Silverlight control to the MoodIndicator.xap file. Change the Width and Height properties of the Silverlight control to 400px (the default is 100 pixels). Press F5 to view the page in the browser. Interact with the buttons to indicate your current mood (see Figure 2).

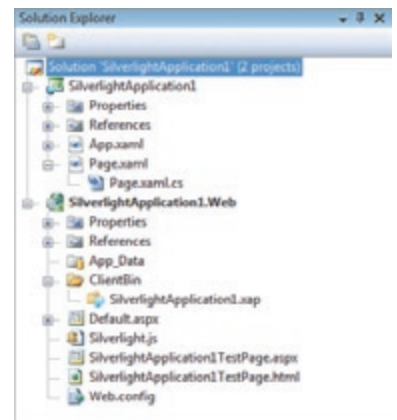
This scenario (where you have a compiled Silverlight application to place on an .aspx page) happens more often than you might think. If a designer or a different group is responsible for creating the Silverlight application, they will provide you with a single .xap file that you will add to an ASP.NET Web site using the steps described in the previous procedure. That's all you need to do.

And even if you are the person developing the Silverlight application, it's helpful that the only point of contact between Silverlight and ASP.NET is a single .xap file. This allows you to work on the Silverlight application or ASP.NET page independently without affecting the other.

The users viewing the page will need to have the Silverlight plug-in installed, but the control detects whether the Silverlight

plug-in is installed (and whether the required minimum version is present) and displays an image that prompts the user to download the Silverlight plug-in or update the plug-in to the required version.

That functionality is provided automatically, but you can customize the experience for users who do not have the Silverlight plug-in installed. You might provide a different graphic than the default image for users who don't have Silverlight installed, perhaps displaying an image showing what they'll see on the page if they download and install the plug-in.



**Figure 3:** Solution Explorer with two projects.

To customize the experience for users who don't have the plug-in installed, set the PluginNotInstalledTemplate property on the ASP.NET Silverlight control to indicate the HTML to display rather than display the standard Install Silverlight graphic.

## Creating Silverlight Content and Testing It on an ASP.NET Page

You may need to create the Silverlight application for use with ASP.NET. Visual Studio 2008 SP1 not only lets you create Silverlight content, but also can provide a test project to host the Silverlight application development.

Let's create a Silverlight project and test it on an ASP.NET page:

- 1) Create a Silverlight Application project with Visual Studio 2008 SP1. Choose either C# or Visual Basic as the language. Name the project MoodIndicator.
- 2) When prompted whether to create a Web project to host the Silverlight application, click OK. Visual Studio creates a solution with 2 projects (see Figure 3).
- 3) Open Page.xaml. Delete its existing content and replace it with the code in Figure 4. Notice that the Path element has a name assigned to it (Name = "mouth"). This is so we can access that element in code in the next step.
- 4) Open the Page.xaml.cs or Page.xaml.vb code-behind file (depending on your language choice). Add the code for the two button click event handlers for C# (see Figure 5) or Visual Basic (see Figure 6). These event handlers apply a transform to the "mouth" when someone clicks on one of the Silverlight application buttons.
- 5) Choose Rebuild Solution from the Build menu. This compiles the project and copies the compiled MoodIndicator.xap Silverlight application to the ClientBin folder of the MoodIndicator.Web project (the test project).

```

<UserControl x:Class="MoodIndicator.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Width="300" Height="280">
  <Grid x:Name="LayoutRoot" Background="White">
    <Ellipse HorizontalAlignment="Stretch" Margin="86,44,64,86"
      Fill="#FFFFFF00" VerticalAlignment="Stretch" />
    <Button Height="32" HorizontalAlignment="Left" Margin="72,0,0,24"
      VerticalAlignment="Bottom" Content="Happy" Name="btnHappy"
      Width="72" Click="btnHappy_Click" />
    <Button Height="32" HorizontalAlignment="Right" Margin="0,0,48,24"
      VerticalAlignment="Bottom" Content="Sad" Name="btnSad"
      Width="72" Click="btnSad_Click" />
    <Ellipse Height="30" HorizontalAlignment="Left" Margin="120,78,0,0"
      VerticalAlignment="Top" Width="24" Fill="#FFFFFF"
      Stroke="#FF000000" StrokeThickness="2" />
    <Ellipse Height="30" HorizontalAlignment="Right" Margin="0,78,98,0"
      VerticalAlignment="Top" Fill="#FFFFFF" Stroke="#FF000000"
      Width="24" StrokeThickness="2" />
    <Path Height="25" HorizontalAlignment="Stretch" Margin="120,0,98,108"
      Name="mouth" VerticalAlignment="Bottom" Stretch="Fill"
      Stroke="#FF000000" StrokeThickness="4"
      Data="M120,200 C120,200 130,223 160,223 C190,223 200,200 200,200" />
  </Grid>
</UserControl>

```

Figure 4: The Page.xaml file.

```

private void btnHappy_Click(object sender, RoutedEventArgs e)
{
    ScaleTransform txfr = new ScaleTransform();
    txfr.ScaleY = 1;
    this.mouth.RenderTransform = txfr;
}

private void btnSad_Click(object sender, RoutedEventArgs e)
{
    ScaleTransform txfr = new ScaleTransform();
    txfr.ScaleY = -1;
    this.mouth.RenderTransform = txfr;
}

```

Figure 5: Event handlers in Page.xaml.cs.

```

Protected Sub btnHappy_Click(ByVal sender As Object, _
  ByVal e As EventArgs) Handles btnHappy.Click

    Dim txfr as New ScaleTransform
    txfr.ScaleY = 1
    Me.mouth.RenderTransform = txfr

End Sub

Protected Sub btnSad_Click(ByVal sender As Object, _
  ByVal e As EventArgs) Handles btnSad.Click

    Dim txfr as New ScaleTransform
    txfr.ScaleY = -1
    Me.mouth.RenderTransform = txfr

End Sub

```

Figure 6: Event handlers in Page.xaml.vb.

- 6) Press F5 to run the test.aspx page of the MoodIndicator.Web project.
- 7) The MoodIndicatorTestPage.aspx file is displayed in the browser. Use the buttons to indicate your mood. Notice that Silverlight buttons display a hover effect when the mouse is over them.

Notice in Figure 3 there also is a MoodIndicator-TestPage.html page in the test project. You can right-click on that file and choose View in Browser to display the compiled Silverlight .xap file on an HTML page.

Each time you compile the Silverlight project, Visual Studio copies the compiled .xap file into the ClientBin folder of the test project.

## Microsoft Expression Blend 2

Although we've seen that you can author Silverlight content in Visual Studio 2008, you might choose to use Blend 2 SP1 to create Silverlight 2 content because you can work interactively on the design surface in Blend, which often is easier than working in the XAML code as currently required in Visual Studio 2008.

You can work directly in XAML with either tool, but only Visual Studio currently provides IntelliSense. I used Blend 2 SP1 to create the ellipses and paths for the Silverlight control in this article. I stripped out the Blend-only display attributes, which Visual Studio ignores, for readability.

## The MediaPlayer Control

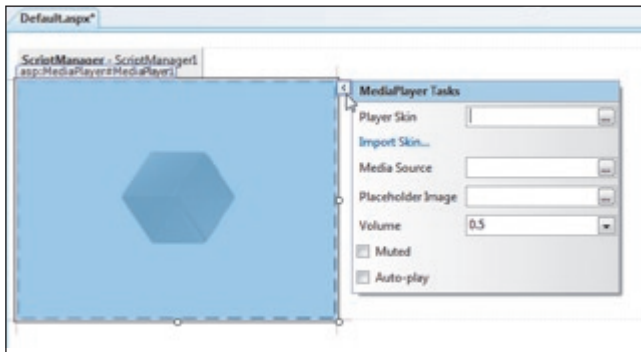
The ASP.NET MediaPlayer control offers a simple way to add audio or video to an ASP.NET Web page. You might think that only a site that specializes in delivering video would use media on a Web page, but consider how an ASP.NET Web site might use video or audio:

- video demonstration of a new product or feature
- installation, repair, or maintenance from tech support
- a message from the CEO to employees or Web site visitors
- testimonials from satisfied clients
- video messages to team members or sales personnel
- a video tour of homes for sale
- in-house training
- excerpts from training classes offered to the public
- demonstrations on preparing delicious foods from the Northwind products catalog
- audio clips for any of the above

## Adding Audio or Video to a Web Page

To play audio or video on a Web page, you must add the ASP.NET MediaPlayer control to the page and set the MediaSource property to the name and location of the audio or video file. This control is included in the Silverlight Tools you downloaded earlier.

The MediaPlayer control inherits from the Silverlight control, but instead of allowing you to have any type of Silverlight content on a page, it is limited in functionality to provide a customizable skin that offers users the ability to stop and start playback, adjust the volume, skip to a specific chapter, etc.



**Figure 7:** MediaPlayer Tasks window.

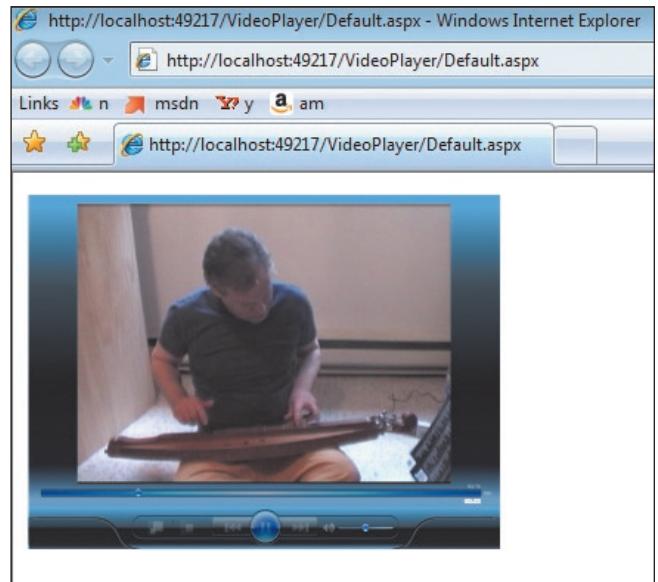
Users viewing the page need to have the Silverlight plug-in installed, but this control inherits from the ASP.NET Silverlight server control and so will detect whether the plug-in is already installed and display a prompt if the plug-in is not installed. To add video or audio to a Web page:

- 1) Create a new Web site named VideoPlayer.
- 2) In the Default.aspx page add a ScriptManager control below the opening tag for the form tag and before its closing tag.
- 3) Right-click on the Web site in Solution Explorer and choose Add an Existing Item. Select any .wmv video file on your system. You can use the Dulcimer.wmv video file that you can download with the source files for this article.
- 4) Drag a MediaPlayer control from the Toolbox to be inside the form tag and below the ScriptManager control. The MediaPlayer control is in the Silverlight section of the Toolbox (again, Figure 1).
- 5) In Design View, the MediaPlayer control displays a smart tag window, as shown in Figure 7, where you can set some properties.
- 6) Set the MediaSource property to the video file you added to the project.
- 7) You can click Import Skin if you want to use a different MediaPlayer skin (Professional is the default). You can select the Auto-play checkbox if you want playback to begin when the page loads. If the Auto-play checkbox is not selected, users will need to click the start button on the MediaPlayer to begin playback. Using the Property Window you can set the Height and Width of the control to any value appropriate for your video file.
- 8) Press F5 to display the default Web page in a browser. You can play or pause the video, as shown in Figure 8.

Most MediaPlayer skins have built-in full-screen playback capability, so someone watching the video can switch to or from full-screen mode by double-clicking on the video or clicking the full-screen button, if the skin provides one.

## How It Works

The MediaPlayer control wraps the functionality of a Silverlight MediaElement object and allows you to simply place the control on a Web page and determine which media file to play. Because the MediaPlayer is an ASP.NET server control,



**Figure 8:** Video on an ASP.NET page.

you can interact with the control programmatically to change any of its properties. A postback is required to change properties currently in the browser. For example, you could allow users to select which video to play from a DropDown control that, when the selection changes, sets the MediaSource property on a MediaPlayer control and causes a postback to occur.

When an .aspx page with a MediaPlayer server control is rendered as HTML sent to the browser, a MediaPlayer client control is generated in the HTML. You can interact with the MediaPlayer client control's properties, methods, and events at run time using scripting. See the documentation for the client-side MediaPlayer control (in the Sys.UI.Silverlight.MediaPlayer namespace). Wei-Meng Lee's article "Media Guide" in the April 2008 issue of *asp.netPRO* has many examples of how to customize the appearance and add client-side scripting functionality to the MediaPlayer control.

## Conclusion

To add Silverlight content or media content to your ASP.NET Web sites, all you need is the free Silverlight Tools download from Microsoft.

As you can imagine, there's much more you can do with these controls. Think about situations where you might want to add rich Silverlight functionality or a splash of video or audio and you'll find you are limited only by your imagination. </>

Source code accompanying this article is available for download to *asp.netPRO* subscribers at [www.aspnetPRO.com/download](http://www.aspnetPRO.com/download).

*Austin Avrashow* is a contract technical writer at Microsoft, most recently with ASP.NET User Education.