

Microsoft®
Silverlight™

Enabling Rich Internet Applications
for the Open Source World



contents

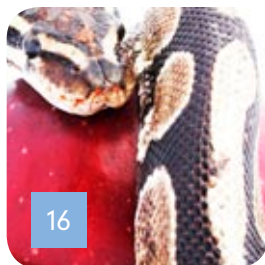
SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD



2 Letter from the Editor



3 Getting Started with Silverlight for Eclipse



16 RIA Applications for Python and Ruby Developers
– Made Possible with Silverlight 2



23 PHP Developers Don't Need Silverlight
– Or Do They?



35 Expression Web 2 for PHP Developers
– Simplifying Your PHP Applications

All the articles in this ebook were commissioned by and prepared for Microsoft Corporation. This document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Letter from the Editor...

By Michael Pastore

There are many exciting developments taking place in the world of Rich Internet Applications (RIAs) and many of them are taking place around Microsoft® Silverlight™. For some developers, Silverlight can be used to add eye-catching visual elements to a Web page or Internet application. For others, Silverlight lets them add the visual appeal and presentation of a client-side software application to their Web-based business application.

For developers who are accustomed to working with Microsoft technologies like Visual Studio 2008, the capabilities of Silverlight are well known. Microsoft has made a big push to ensure that .NET developers who work with Visual Studio and Expression Studio know all about the advantages of using Silverlight and what's coming as the Silverlight technology continues to mature.

But what about developers who work with open source development tools and languages? What about dynamic language programmers who aren't oriented toward Microsoft tools and technologies? Microsoft has plenty to offer them as well.

This eBook will explore how open source developers can take advantage of all that Microsoft has to offer developers of RIAs without forcing them to learn new tools.

We'll begin our eBook with Sheryl Canter showing us how developers who work in the Eclipse IDE can take advantage of a new Silverlight plug-in that offers many of the same features as Silverlight Tools for Visual Studio. As you follow along with the article you'll create a simple syndication feed reader with Silverlight that can be built with very little code.

Then we'll turn our attention to developers who work in dynamic languages. Thanks to support for Python and Ruby in Silverlight 2, dynamic language coders now have easy access to powerful tools for building RIAs. Silverlight 2 allows these developers to write client-side web programming code in their favorite language—instead of forcing them to switch to JavaScript.

Next, Justin Whitney will show PHP developers how they can use the Silverlight runtime, XAML, PHP code, and a touch of JavaScript to add some splash to their PHP apps. Justin will walk through the development of a simple app without using Microsoft development tools. The demo will focus on the Silverlight runtime's role as a XAML interpreter, and since it doesn't get into .NET development, you can follow along with either Silverlight 1.0 or 2.0.

Cal Schrottenboer then continues our look at PHP development by examining Microsoft's Expression Web 2 and its enhanced support for PHP. Thanks to its robust testing and development environment, support for PHP includes, IntelliSense and proper

handling of Byte Order Marks (BOMs), and more, Expression Web 2 simplifies the development of PHP applications.

Rich Internet Applications offer developers an avenue for creating visually appealing applications that can perform all the functions of a packaged business app, and now this functionality is available to developers regardless of their preferred tools and language. Enjoy and happy coding! ■





Getting Started with Silverlight for Eclipse

Microsoft® is taking interoperability seriously when it comes to Silverlight. Now, thanks to Microsoft and Soyatec, Eclipse developers will be able to create Microsoft Silverlight™ applications right in their favorite IDE. by Sheryl Canter

Fans of the open source and highly configurable Eclipse IDE got a present recently: a new beta of the Silverlight plug-in for Eclipse. It offers many of the same features as [Silverlight Tools for Visual Studio® 2008](#), including code-hinting and XAML preview. The plug-in is being developed by Soyatec with funding, architectural guidance, and project management from [Microsoft's Interoperability Team](#). The best way to tour the features of Silverlight for Eclipse is to actually try it. So in this article we'll create a simple Silverlight app using Eclipse--a syndication feed reader. The app lists feed items in the top of the window, and displays the selected feed item beneath the list. It's amazing how little code it takes to put this together. You can see the finished app in Figure 1.

Install the Silverlight Development Toolkit

The Silverlight module requires the following software, installed in the order shown:

- Microsoft Windows® XP SP2 or higher, or Windows Vista SP1
- [Java Development Kit \(JDK\) 1.5.0-11 or higher](#)
- [Eclipse Ganymede 3.4.1, 32-bit version](#)
- [Microsoft .NET® Framework 3.5](#)
- [Silverlight 2 Runtime 2.0.31005.0 or higher](#)
- [Silverlight 2 SDK](#)

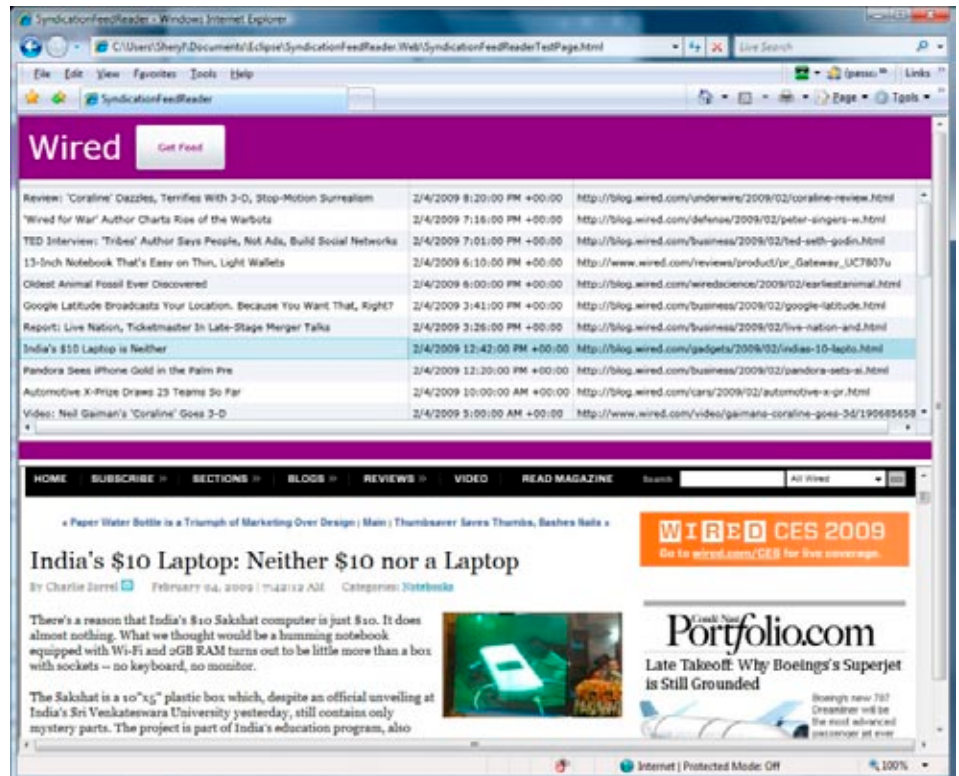


Figure 1. Syndication Feed Reader

SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD

A few things to note with this beta:

- There's no debugger yet, but one is planned by summer.
- The Silverlight plug-in for Eclipse doesn't yet work on 64-bit Windows Vista® systems, even if you're running the 32-bit version of Eclipse. A fix for this is also in the works.
- Finally, the basic version of Ganymede is fine to use because when you install the Silverlight module, it will pull in all the dependencies.

After you install Silverlight, you should update to the [December 2008 release of the DataGrid](#), which fixes nearly 30 bugs. We'll be using the DataGrid in our tutorial.

Once you've installed all the prerequisites, you are ready to install the Silverlight plug-in. Launch Eclipse, then select Help->Software Updates to open the Software Updates and Add-ons dialog. Select the Available Software tab and click the Add Site... button.

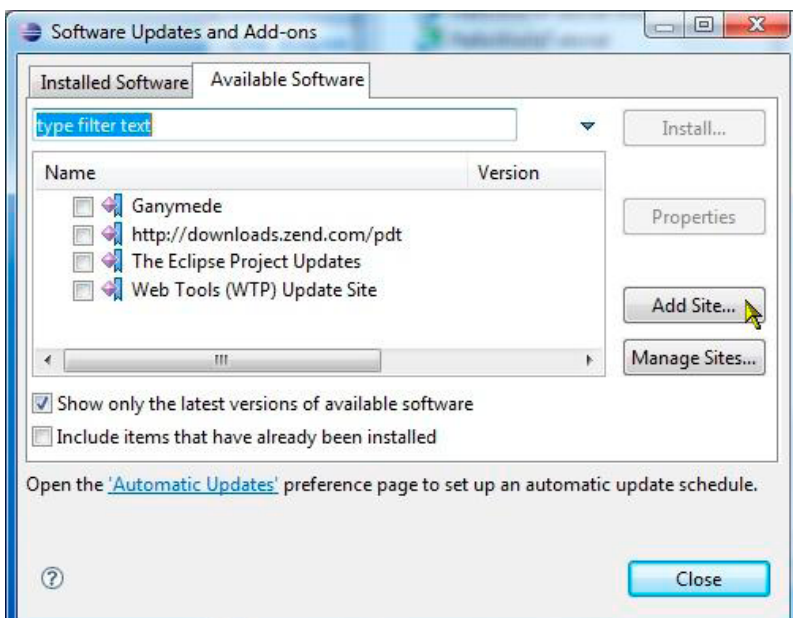


Figure 2. Select the Available Software tab and click the Add Site... button.

In the resulting dialog box, type `http://www.eclipse4sl.org/update/` and click OK. Don't leave off the "www". If you do, you'll get an annoying duplicate entry that doesn't hurt anything but looks ugly.

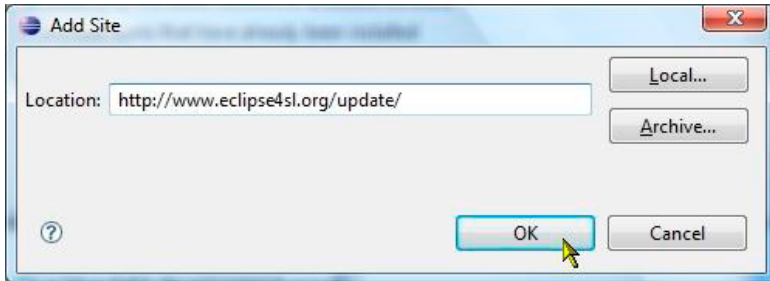


Figure 3. Type `http://www.eclipse4sl.org/update/` and click OK.

Now there will be a new entry on the Available Software tab for Silverlight. Select the Silverlight entry, then click the Install... button. Eclipse will install the Silverlight plug-in and all its dependencies.

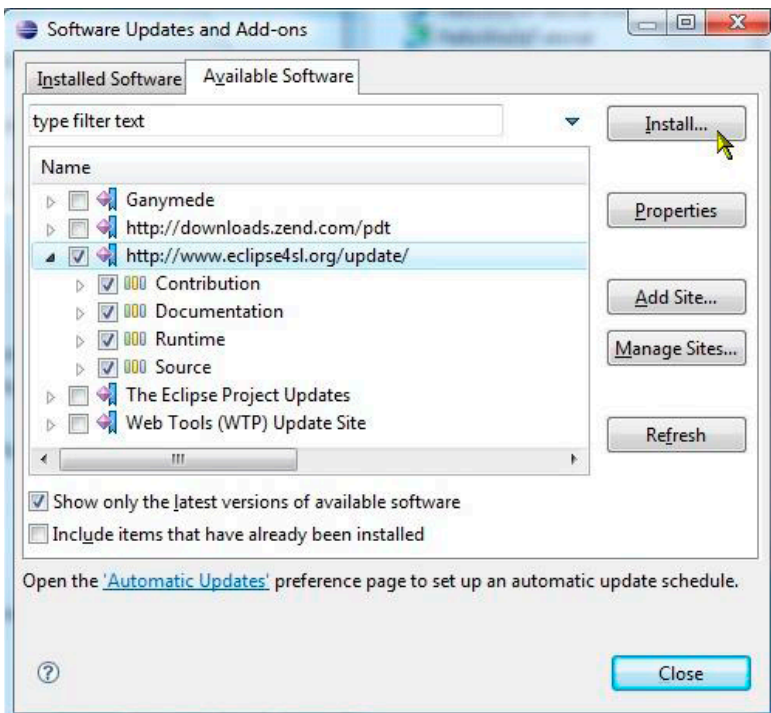


Figure 4. Select the Silverlight entry, then click the Install... button.

To see exactly what it installed, select Help | About. The plug-in button with the cube icon is for Silverlight.

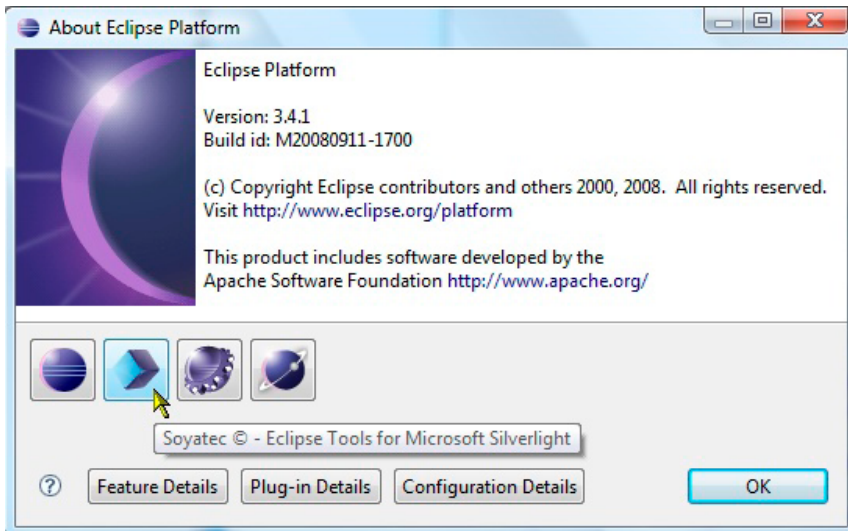


Figure 5. The plug-in button with the cube icon is for Silverlight.

Click the cube icon to bring up the Silverlight module information dialog:

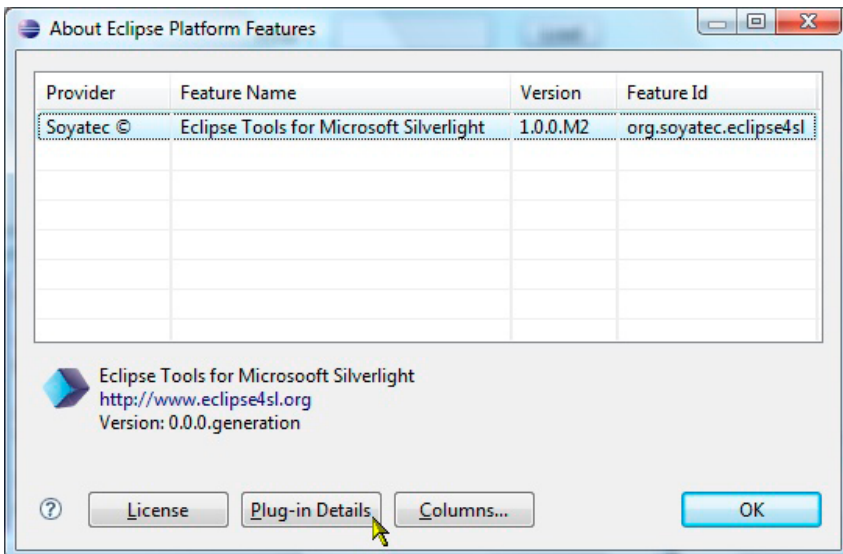


Figure 6. Click the cube icon to bring up the Silverlight module information dialog.

Then click the Plug-in Details button to see all the components:

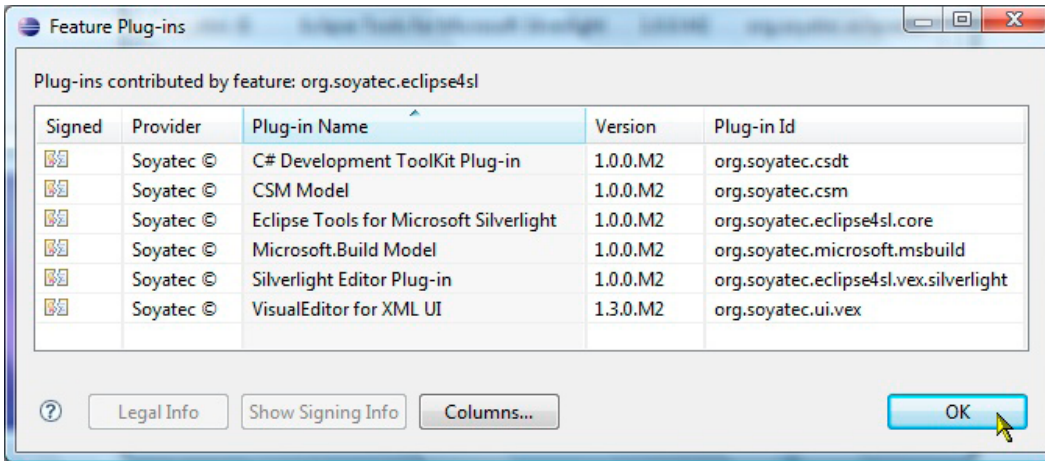


Figure 7. Click the Plug-in Details button to see all the components.

Create the Project Template

Now we're ready to build our Silverlight app. To start, select Windows->Open Perspective->Other... to open the perspective selection dialog:

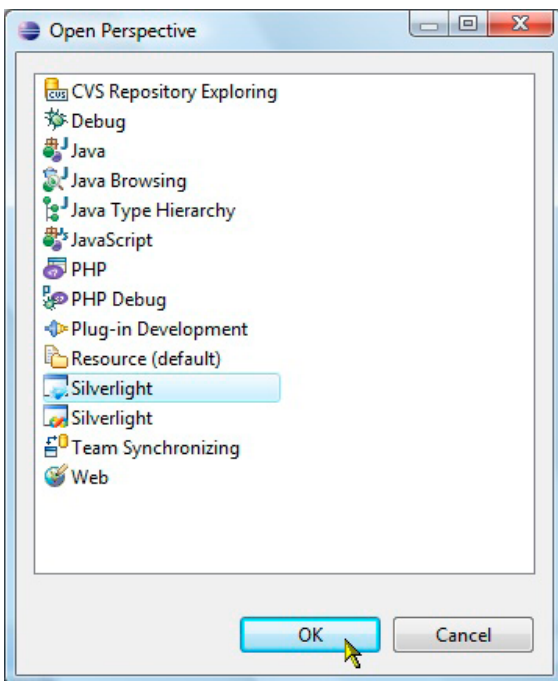


Figure 8. Select Windows->Open Perspective->Other... to open the perspective selection dialog.

Notice that Silverlight is listed twice. The first entry, with the little blue cube, is the Eclipse-style perspective with the Project Explorer docked on the left and the palette docked on the right. The second entry is the Visual Studio style perspective, which reverses these.

Now that the Silverlight perspective is set, select File->New. You'll see two types of Silverlight projects to choose from: Silverlight Project and Silverlight Web Project. Select Silverlight Web Project. This lets you embed your Silverlight app within a Web page that has other elements.

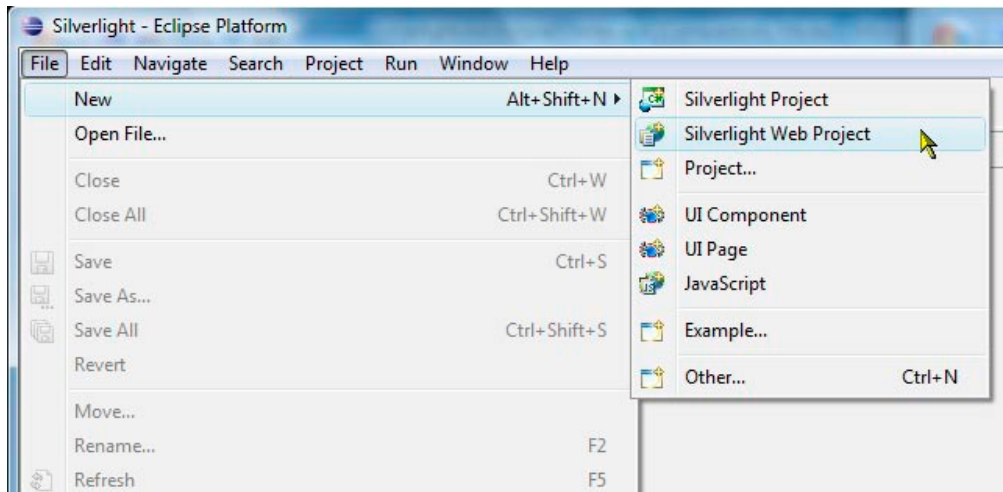
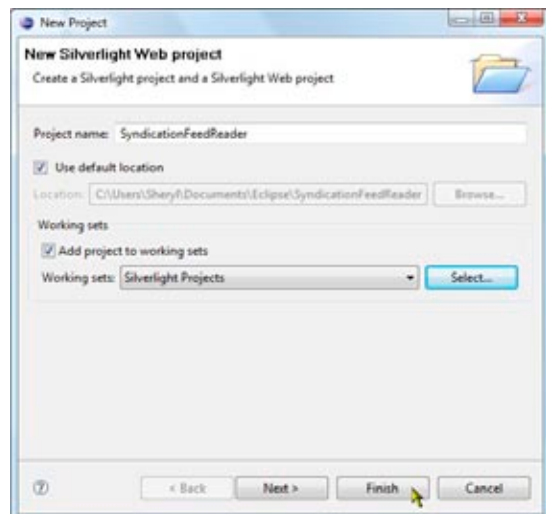


Figure 9. A Silverlight Web Project lets you embed your Silverlight app within a Web page that has other elements.

Next you'll see the New Project dialog. Enter SyndicationFeedReader as the project name. This project doesn't depend on any other projects so you don't have to click Next. Just click Finish:



Before we go any further, let's see what was created for us. Project Explorer shows two projects: SyndicationFeedReader and SyndicationFeedReader.Web:

Figure 10. In the New Project dialog, enter SyndicationFeedReader as the project name.

SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD

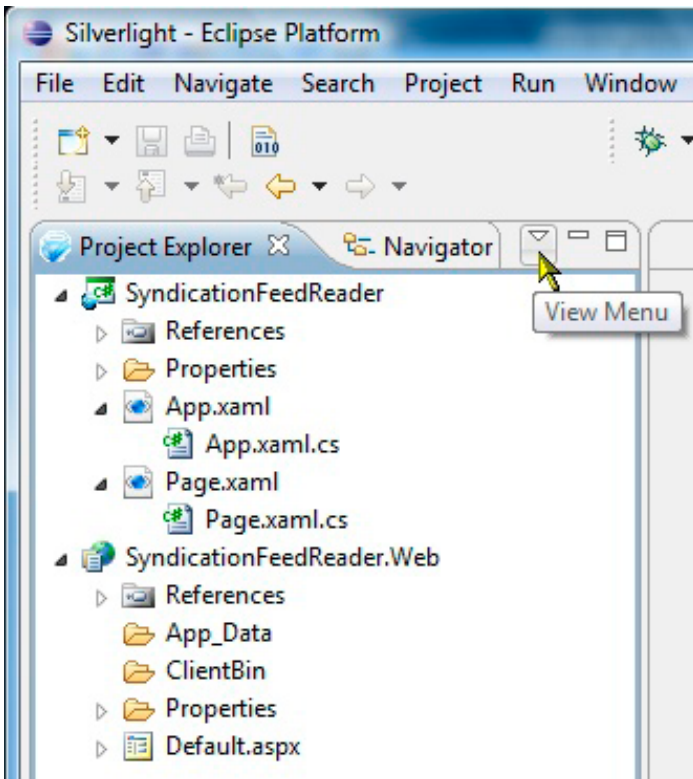


Figure 11. Project Explorer shows two projects: SyndicationFeedReader.Web and SyndicationFeedReader.

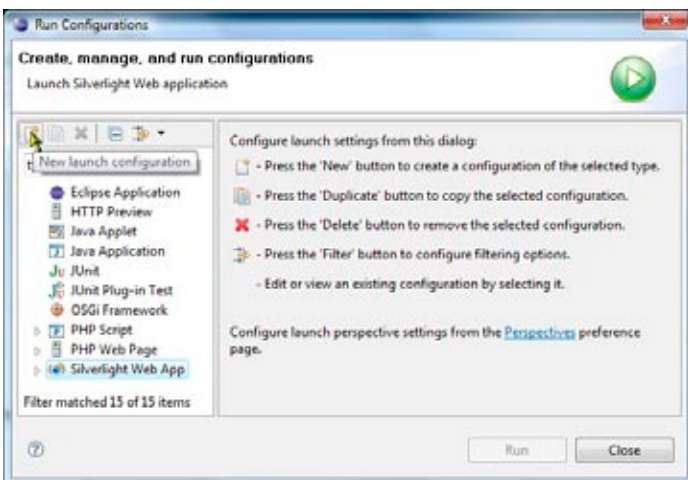


Figure 12. Select Run->Run configurations... to bring up the Run Configurations dialog.

In the screenshot to the left, the mouse cursor is hovering over the View Menu button. The View Menu doesn't display in the current version of the plug-in, but you can open and close your Silverlight projects from other Eclipse perspectives.

Let's build and run what we have so far. It will display only a blank page, but it gives us a chance to set things up. Select Run->Run configurations... to bring up the Run Configurations dialog.

Select Silverlight Web App from the list, and then click the New button:

Next you'll see the configuration dialog. In the Name field enter SyndicationFeedReader, and for Project to test select SyndicationFeedReader.Web. The start page, SyndicationFeedReaderTestPage.html will be selected. Click Apply and then Run:

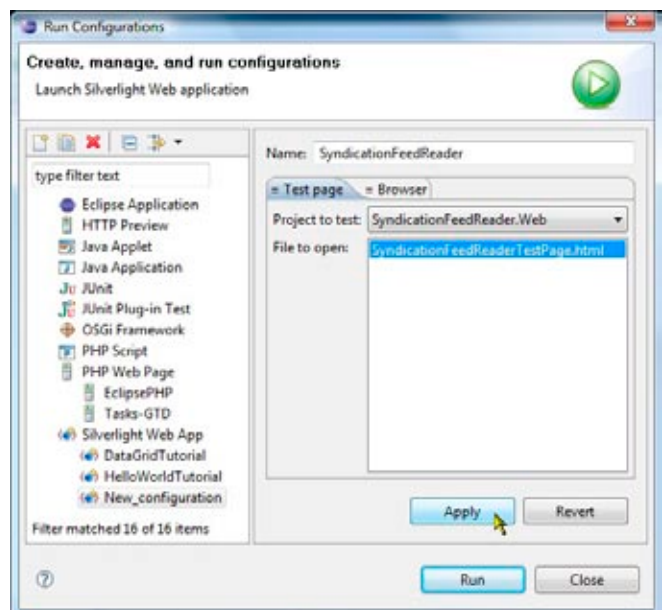


Figure 13. Enter the run configuration name and select the project start page.

This will launch what appears to be a blank page in your browser but if you right-click in the browser window, you'll see the Silverlight Configuration context menu. Silverlight is running – there's just nothing there yet.

Note: We will need to make some small changes in `SyndicationFeedReaderTestPage.html`, but it may not be listed as part of the `SyndicationFeedReader.Web` project in Project Explorer. (Soyatec has fixed this bug, but the fix may not be released by the time you read this.) If it's missing, here's how to add it. Right-click on the project name `SyndicationFeedReader.Web` and select `New -> Add existing file`. Then navigate to the project directory, change the filter to `*.*`, and select the file. You'll be asked if you want to overwrite the existing file. Say no. The file will appear in your project list.

Customize the Code for Our App

Now let's create the user interface by adding code to `Page.xaml`. XAML (pronounced "zammel") stands for eXtensible Application Markup Language. XAML files are a type of XML. Double-click `Page.xaml` to open it in the editor. The lower pane holds your source code, and the top pane gives a preview of what it will look like.

```
<UserControl x:Class="SyndicationFeedReader.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  >

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

  </Grid>

</UserControl>
```

The Silverlight Grid control works much like an HTML table, though the syntax is different. Our Silverlight app will contain two rows. The top row will hold a label and a button for accessing the feed items. The bottom row will contain the list of feed items. The display of the selected feed item will be done in HTML, below the embedded Silverlight app.

To add row definitions, use this syntax:

```
<UserControl x:Class="SyndicationFeedReader.Page"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  >

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

    <Grid.RowDefinitions>
      <RowDefinition Height="70" />
      <RowDefinition Height="*" />
    </Grid.RowDefinitions>

  </Grid>

</UserControl>
```

Next we want to add a horizontal StackPanel to hold the label text and the button. The Palette on the right can help. Click on Containers, then drag the StackPanel item to the code window below the row definitions but still inside the grid. Insert your cursor at the end of the <StackPanel> tag, type a space, and then press Ctrl+Space to bring up code hinting. Scroll down until you reach Orientation (or type "O" to speed your search) and double-click to select it. The default value of "Horizontal" is what we want.

Our two StackPanel controls will be a TextBlock and a Button. The TextBlock will simply say "Wired" (the name of the feed we are using in this tutorial). Note that not all feeds will work. The domain must permit [cross-domain access](#) by hosting a small file on the server such as [this one on wired.com](#).

To add a TextBlock, click Controls in the Palette, then drag the TextBlock item to your source code window, between the StackPanel tags. Then click on Button and drag a Button control to the source window below the TextBlock. Configure the controls as shown below. We don't have to tell it to use the first line of the grid (row 0) because it does this by default. As with the StackPanel, you can get syntax help by pressing Ctrl+Space.

```
<StackPanel Orientation="Horizontal">

    <TextBlock Text="Wired"
        VerticalAlignment="Center"
        FontSize="36"
        Padding="10"
        Foreground="White"
    />

    <Button Content="Get Feed"
        Margin="5"
        Height="50"
        Width="100"
        Foreground="Purple"
    />

</StackPanel>
```

So far so good, but the button doesn't do anything yet. We need to add an event handler so that clicking on the button accesses the feed. Position your cursor before the button's closing bracket and press Ctrl+Space. One of the options in the code hint box is the Click event. Double-click on this to select, and an event handler dialog will appear.

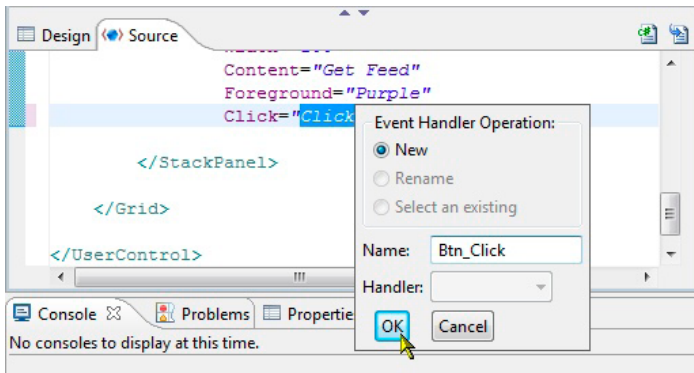


Figure 14. Select the Click event from the code hint box and an event handler dialog will appear.

Type in Btn_Click as the event name, and click OK: This will then create an event handler stub in the code-behind file, Page.xaml.cs, and open it for editing.

The event handler needs just three lines of code.

First we instantiate a WebClient object for accessing server data. Then, since we will access the feed asynchronously, we need to add an event handler for the OpenReadCompleted event. And finally, we use the OpenReadAsync method to submit the request for the feed:

```
private void Btn_Click(object sender, RoutedEventArgs arg1)
{
    WebClient client = new WebClient();
    client.OpenReadCompleted +=
        new OpenReadCompletedEventHandler(client_OpenReadCompleted);
    client.OpenReadAsync(
        new Uri("http://feeds.wired.com/wired/index?format=xml"));
}
```

The OpenReadCompleted event handler will bind the feed data to the control that displays it--in our case, the DataGrid control. So before we add the handler, let's go back to Page.xaml and add the DataGrid.

Click Containers in the Palette, drag the DataGrid item to your source window, and drop it beneath the StackPanel (between the two close tags </StackPanel> and </Grid>). This not only adds the DataGrid tags to your source code, it adds the additional namespace to the top of the file, and adds a reference to System.Windows.Controls.Data to your project.

We want to give our DataGrid a name so we can refer to it in the OpenReadCompleted event handler, where we will bind the feed data to it. Give it the name gridSyndicationItems, and then add two more attributes: Grid.Row="1" (since the list goes in the second row of the grid--the first is row 0) and AutoGenerateColumns="false" (more on this in a minute). Finally, we want our Web page to display the selected feed, so we need an event handler for SelectionChanged. Code hinting can help with all of these; just press Ctrl+Space.

If you don't set `AutoGenerateColumns` to false, the `DataGrid` will automatically generate a column for every field in the feed, and generally you're not going to want that. If you set it to false, you can then define which columns to display, which is what we want to do. This is what the `DataGrid` markup should look like when you're done:

```
<Controls:DataGrid x:Name="gridSyndicationItems" Grid.Row="1"
    AutoGenerateColumns="false" SelectionChanged="gridFeed_SelectionChanged" >
    <Controls:DataGrid.Columns>
        <Controls:DataGridTextColumn Binding="{Binding Title.Text}" />
        <Controls:DataGridTextColumn Binding="{Binding PublishDate}" />
        <Controls:DataGridTextColumn Binding="{Binding Id}" />
    </Controls:DataGrid.Columns>
</Controls:DataGrid>
```

The `Binding` statements specify what fields to display in each column. The field names are [properties of the Syndication Field object](#).

Now let's go back to our code-behind file, `Page.xaml.cs`, and add the event handler for when the feed reading has completed. Type the code for the `OpenReadCompleted` handler right below the `Btn_Click` handler:

```
private void client_OpenReadCompleted(object sender, OpenReadCompletedEventArgs e)
{
    if (e.Error == null)
    {
        XmlReader reader = XmlReader.Create(e.Result);
        SyndicationFeed feed = SyndicationFeed.Load(reader);
        gridSyndicationItems.ItemsSource = feed.Items;
        reader.Close();
    }
}
```

If there are no errors in getting the feed, we create a new instance of the `XmlReader` object and initialize it with our feed. Then we load the XML reader into a `SyndicationFeed` object called "feed" whose `Items` property contains the collection of feed items. To bind the feed data to the `DataGrid`, we assign `feed.Items` to the `ItemsSource` property of the `DataGrid`.

SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD

One thing, though. The namespace that contains the SyndicationFeed class is not yet a part of our project. In Project Explorer, expand the SyndicationFeedReader project, right-click on References, and select Add Reference:

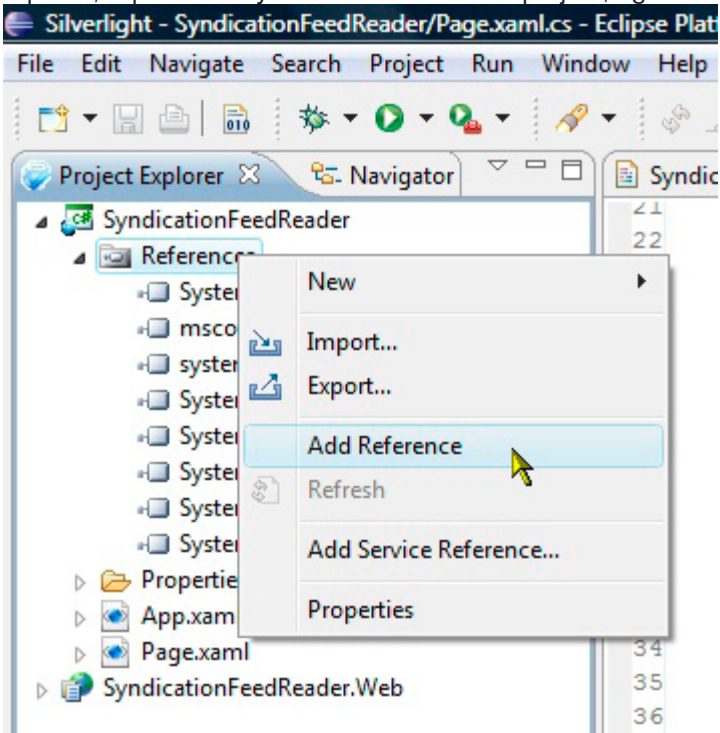


Figure 15. Right-click on References in the SyndicationFeedReader project and select Add Reference.

Now we need to add a few "using" directives to our code-behind file, Page.xaml.cs:

```
using System.ServiceModel.Syndication;  
using System.Xml;  
using System.Windows.Browser;
```

System.Xml and System.Windows.Browser are used in displaying the selected feed item. There are two parts to this:

- An event handler for when the selected feed item has changed.
- HTML to display the selected feed item in an iframe on the page.

This brings up a list of available assemblies, with those currently part of the project highlighted in grey.

Select System.ServiceModel.Syndication and click OK:

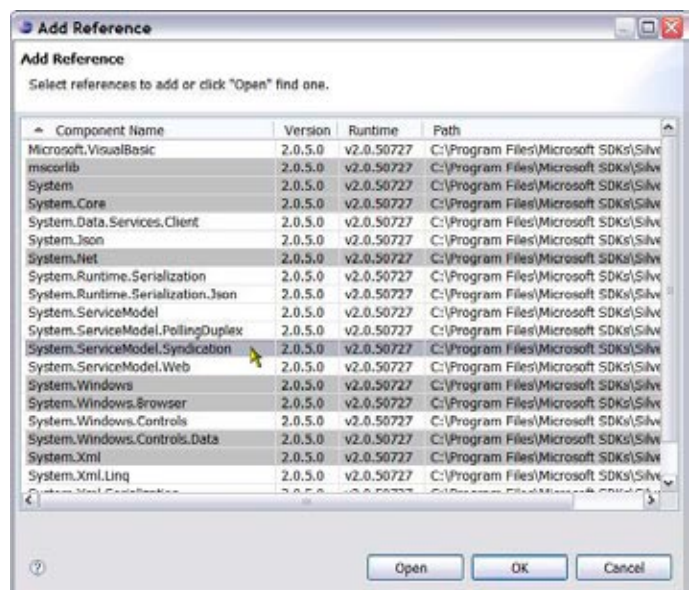


Figure 16. Select System.ServiceModel.Syndication and click OK.

Add the SelectionChanged event handler below the OpenReadCompleted event handler. This time the stub isn't created for us (code hinting will be enhanced in future versions):

```
private void gridFeed_SelectionChanged(object sender, EventArgs e)
{
    SyndicationItem selectedItem =
        (SyndicationItem)gridSyndicationItems.SelectedItem;
    HtmlElement element =
        HtmlPage.Document.GetElementById("rssFrame");
    element.SetAttribute("src", selectedItem.Links[0].Uri.ToString());
}
```

The first line updates the selected item, and the next two lines allow it to be displayed. Notice the reference to "rssFrame". This is name of the iframe that will hold the display.

The last step is to add the iframe to our HTML page. Right click on SyndicationFeedReaderTestPage.html in the SyndicationFeedReader.Web project, and select Open With->Text Editor. First, limit the embedded Silverlight app to 50 percent of the page by changing the height of the silverlightControlHost from 100 percent to 50 percent:

```
#silverlightControlHost {
    height: 50%;
}
```

Then add these two lines just before the </body> tag:

```
<hr style="color: Purple; height: 20px;" />
<iframe id="rssFrame" width="98.5%" height="50%" />
```

That's it! Save everything, and then go to the Run menu and select Run History->SyndicationFeedReader. When the browser window with the app is launched, click the Get Feed button to load the feed items. As you select different items, the associated page will load in the bottom half of the window. You can run the app outside of Eclipse by double-clicking SyndicationFeedReaderTestPage.html.

Interoperability with Java Web Services

We've built a very simple demonstration app here, but the Silverlight plug-in for Eclipse is not limited to creating standalone Silverlight applications. It also supports interoperability with Java Web services based on SOAP or REST protocols. You can find [details and sample code online](#).

The Silverlight project for Eclipse is in active development. Your [feature requests and feedback](#) are welcome. ■

* This article originally appeared in the RIA Development Center at <http://www.devx.com/RIA/Article/40806>

Sheryl Canter is a technophile and developer for Windows and the Web. She's written extensively about computers and technology for PC Magazine and other publications, and for 10 years was editor of PC Magazine's popular Utilities column. Her Web site and blog are at www.SherylCanter.com.



RIA Applications for Python and Ruby Developers – Made Possible with Silverlight 2

If you're a Python or Ruby fan, you may feel you're missing out on all the latest developments in Rich Internet Applications. With support for Python and Ruby in Microsoft® Silverlight™ 2, dynamic language developers now have easy access to the latest RIA technologies. by Cal Schrottenboer

For most Microsoft Visual C#® and Visual Basic.NET® programmers, the evolution of dynamic language support by Microsoft has probably progressed well below the radar screen. When I polled a number of my colleagues on this topic, I was met mostly with mere puzzlement and no knowledge of any support for dynamic languages by Microsoft. My research, however, has disclosed that a lot has actually been happening that neither I nor my friends have been aware of.

At the same time, dynamic language programmers, most of whom are likely not Microsoft oriented, may not have noticed that Silverlight 2 now allows them to write client-side web programming code in their favorite language--instead of forcing them to switch to JavaScript. This is a significant new development, something not previously possible in any other context, including Flash. When I pointed this out to a friend of mine who is a PHP developer, he was quite impressed.

A recent article entitled "[Developers Shift to Dynamic Programming Languages](#)" by Linda Dailey Paulson offers an explanation for the rise in popularity of dynamic languages. She writes: "Software developers are always looking for ways to boost their effectiveness and productivity and perform complex jobs more quickly and easily, particularly as projects have become increasingly large and complex.... With this in mind,

many developers are increasingly using dynamic languages such as JavaScript, Perl, Python, and Ruby." Moreover, in my conversations with dynamic language developers, the rapid development cycle and the ability to see changes immediately without the need to recompile are the most commonly stated reasons for their preference for dynamic languages.

This article will focus specifically on dynamic language support in the recently released Silverlight 2. This topic should be of potential interest to three categories of developers:

1. Python or Ruby developers who would like to be able to use Python or Ruby in the browser.
2. Classic C# and Visual Basic.NET developers who, based on my informal and non-scientific survey, most likely are not aware of the advantages that they might reap from using one of the dynamic languages in the right circumstances to supplement their normal development.
3. Developers of any stripe who want to offer scripting capabilities to end users via a web application.

Currently, dynamic language support by Microsoft extends to three languages: Python, Ruby and JavaScript. Of these three Microsoft supported dynamic languages, the most advanced at this point is IronPython. Version 2.0, RC1 was released in October 2008. By contrast, IronRuby is still in Alpha for Release 1.0. (I am advised that for open source languages--

which are continuously updated and not officially supported by Microsoft--there is not quite the same degree of significance associated with the terms Alpha and Beta.) The third language is Managed JScript, a Microsoft version of JavaScript.

Let's look briefly at each of Microsoft's three existing dynamic languages in turn, with a little bit more emphasis on IronPython, given that it is currently the most advanced.

IronPython

[IronPython](#) is a Microsoft-created version of Python. IronPython 2 is equivalent to Python 2.5 and includes all of its core features. Even though it is written in C#, IronPython supports most of the "standard library", a large collection of modules written in a combination of C and Python providing many pre-built tools to boost developer productivity.

One of the key advantages of IronPython over other versions of Python such as CPython or Jython is that IronPython is capable of using almost all of the classes in the .NET Framework. As most .NET programmers know, the .NET Framework contains thousands of classes designed to facilitate almost any programming task. The availability of both the standard library and the .NET Framework represents a major incentive for switching to IronPython from other flavors of the Python language when developing for the Microsoft Windows® platform.

The output of the IronPython engine can be set to be a compiled dll which is capable of running via the .NET Common Language Runtime (CLR). Now that Silverlight 2 has been released, the [CLR](#) is present not only on that portion of Windows computers which have the full .NET Framework installed, but also on any computer running the Silverlight 2 browser plug-in. Silverlight 2 works in Microsoft Internet Explorer® and Firefox on Windows, Firefox and Safari on Mac (OS X and higher and on the Intel platform only) and Firefox on Linux / Unix via the Moonlight project (when released). This means that applications written in Python can (eventually) run in a browser on all three of the major platforms.

The .NET CLR also provides IronPython applications with memory management, multi-threading capability and security features. Similarly, in Silverlight the CoreCLR provides comparable capabilities to IronPython applications. In both cases another component, the Dynamic Language Runtime

(DLR, discussed below) is required. Michael Foord, author of IronPython in Action, asserts that [IronPython running in Silverlight is a couple orders of magnitude faster than traditional JavaScript](#).

IronRuby

[IronRuby](#), also written in C#, is the Microsoft implementation of the Ruby dynamic programming language. IronRuby was [announced at MIX07](#) and although at about a year and a half in development is just now approaching its first full release.

Ruby also has a standard library. Work is currently underway to provide support in IronRuby for [most of the standard library](#). Microsoft is also working on [support for the Rails framework](#). And, unlike non-Microsoft implementations of Ruby, IronRuby can directly access almost all of the classes contained in the .NET Framework, thereby greatly expanding the number of pre-built tools available for any programming task.

As with all of the Microsoft dynamic languages, IronRuby runs on the DLR which makes it interoperable with each of the other Microsoft dynamic languages and with Silverlight.

Managed JScript/ Microsoft Dynamic Languages

JScript is Microsoft's version of JavaScript (technically ECMA Script 3.0) which runs in Internet Explorer. [Managed JScript](#) (as compared to JScript.NET) is the version of JScript available in Silverlight which is designed to take advantage of the DLR.

Dynamic languages have many devoted supporters, as do statically typed languages, and rail cars of ink have been spilled discussing the relative merits of each. The question of which one is better for creating your RIA is rendered moot because existing dynamic language supporters do not need to switch to C# or Visual Basic.NET in order to create a Silverlight application. As can be seen from the sample applications discussed in this article, it is possible to create Silverlight applications using exclusively one of the Microsoft dynamic languages.

Most interestingly, two of Microsoft's dynamic languages are Open Source. Both IronPython and IronRuby have been released under the [Microsoft Public License](#).

The source code for IronPython is available at [CodePlex](#). The source code for IronRuby is hosted on [Rubyforge](#). Since Managed JScript is not Open Source, only the binary files (Microsoft.JScript.Runtime.dll and Microsoft.JScript.Compiler.dll) are released. The location of the latest versions should always be available from the [Silverlight Dynamic Languages SDK site on CodePlex](#) (discussed below).

Silverlight Dynamic Languages SDK

To provide developers with the tools which they would need to use Microsoft dynamic languages for building Silverlight applications, Microsoft created the Silverlight Dynamic Languages SDK. Version 0.4.0 was released in mid-October 2008. It has four main components: the Dynamic Language Runtime (DLR), assemblies for IronPython, IronRuby and Managed JScript, Chiron (a tool for creating .xap files for dynamic language applications, discussed below) and Microsoft.Scripting.Silverlight.dll which provides the integration between Silverlight and the DLR. It also includes a number of sample applications written in each of the three supported dynamic languages.

Dynamic Language Runtime

The Dynamic Language Runtime (DLR) is a critical element for dynamic language applications running under Silverlight. The DLR is a common substrate for all of Microsoft's dynamic languages that provides them with code generation capabilities, efficient method dispatch, name lookup and debugging support. Most importantly, the DLR is what enables Microsoft's dynamic languages to interoperate with the .NET Framework and with each other.

The DLR is contained in two assemblies, Microsoft.Scripting.dll and Microsoft.Scripting.Core.dll. Since neither of these assemblies is included in the Silverlight browser plug-in, it is necessary for a developer to package them with any dynamic language application. This is one of the tasks of the Chiron tool (discussed below).

Chiron

[Chiron](#) is a special tool for creating Silverlight application packages (xap files) for projects which use one of the Microsoft dynamic programming languages. Chiron can be used either from a command line or indirectly from Microsoft Visual Studio® 2008 (see Figure 4). The syntax for calling Chiron from a command line is relatively simple (assuming

that the command line current directory is the folder containing index.html and the folders containing the Python/Ruby/JScript file(s) and other assets).

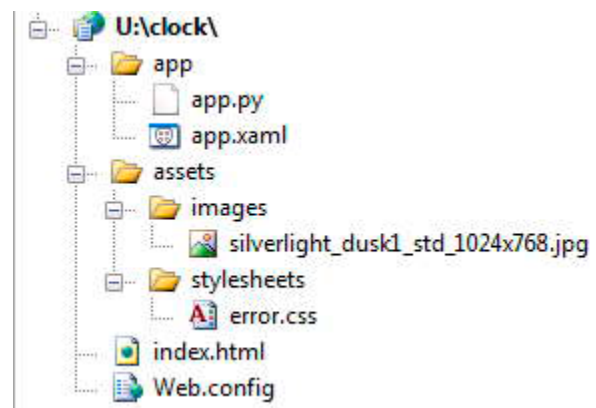


Figure 1. This screenshot illustrates the folder structure for a simple IronPython Silverlight application. In this case the required current directory from which to call Chiron would be U:\clock\.

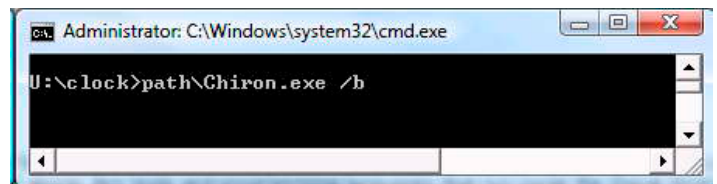


Figure 2. This screenshot illustrates how to call Chiron with the /b parameter which launches the user's default browser to view the Silverlight application. "Path" is simply an alias for the actual physical path to Chiron.exe.

Note that Chiron.exe is initially found in the bin folder in the Silverlight Dynamic Language SDK along with several other dll files. Generally, moving Chiron to another location is not possible without also moving all of these other .dll files, although a separate path to these support files can be specified in the Chiron config file. Of course, the location or locations of Chiron and its support files can also be added to the PATH environment variable.

SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD

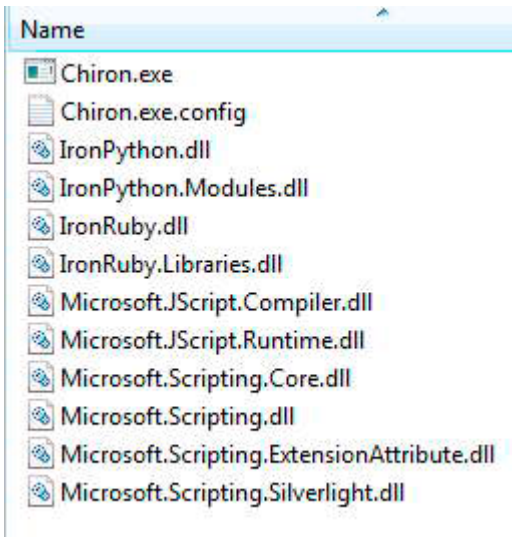


Figure 3. This screenshot shows the list of files contained in the bin folder of the Silverlight Dynamic Language SDK. Access to all of these files is required for the proper functioning of Chiron.

While Python and Ruby developers will likely run Chiron from the command line, most C# and Visual Basic.NET developers will probably find it easiest to [use Chiron from inside Visual Studio](#). (See also the section on IronPython Studio, below) Doing so requires setting three parameters in the Start Options of your dynamic language web application. These parameters can be found in your application's property pages on the Start Options page.

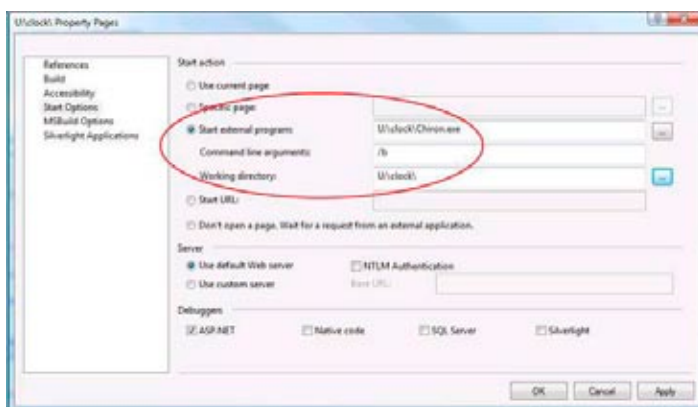


Figure 4. This screenshot points out the required Start Options which must be set in order to use Chiron from Visual Studio.

By default, the "Start action" is set to "Use current page". Unfortunately this will not work in a Dynamic Language project because there is no .xap file available. Creating the .xap file is the responsibility of Chiron, whose path must be specified in the first field as the external program to start when the project is run. The command line argument "/b" tells Chiron not only to start the built-in web server but also to launch the user's default browser to display the project's Start Page. The "Working directory" tells Chiron where to find the data files that it will need to build the required .xap file (i.e., the .py (or .rb) and .xaml (if any) files that make up your web application).

Running the sample projects provided with the Silverlight Dynamic Language SDK simply requires opening the folder containing the file "index.html" in Visual Studio (File / Open / Web Site). Right click the Project node in the Solution Explorer and open the Property Pages. Set the Start Options as described above and press F5 to run the project.

During development (using the /b parameter) the .xap file which is created by Chiron is written only to memory and not to disk. This is convenient because it can speed the performance of the application when testing modifications of the dynamic language code. However, for deployment it is preferable to save the .xap file to disk. This can be achieved by running Chiron from the command line and specifying both the source path (/d[irectory]) and the name to be given to the .xap file (/z[ipdlr]). Chiron will then build the project and include both the DLR assemblies and whatever specific language assemblies are required by your project. If necessary, Chiron will also generate an AppManifest.xaml file.

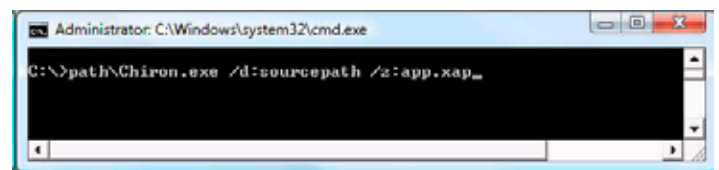


Figure 5. This screen shot illustrates calling Chiron to create the app.xap file where the project files are located in the sourcepath folder. Of course, both "path" and "sourcepath" are aliases for the actual physical paths which would be required in this case.

IronPython Studio

The [Extensibility group for Visual Studio](#) has released a product called [IronPython Studio](#) designed to facilitate Python development. IronPython Studio comes in two modes: Isolated and Integrated. This means that a licensed copy

SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD

of Visual Studio 2008 is not required in order for existing Python developers to use IronPython Studio. In cases of that sort, IronPython Studio can be installed on top of the royalty-free [Visual Studio 2008 Shell runtime](#). Existing users of Visual Studio 2008, on the other hand, can take advantage of the [Integrated installation mode](#).

Sample Applications

Fractulator, one of the sample applications included with the Silverlight Dynamic Language SDK, nicely illustrates an application written entirely in IronPython running in a web browser. This application presents a user with an input box into which he is suggested to type a mathematical expression. Client side IronPython code immediately formats this expression into a user friendly layout, calculates the result of the expression and displays the IronPython syntax which could be used to represent this expression.

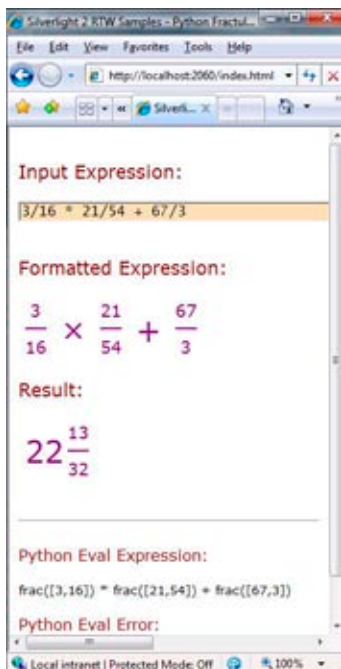


Figure 6. This Fractulator screenshot shows a user entered mathematical expression, its formatted equivalent, the formatted calculated result and the IronPython eval expression syntax.

Photoviewer is another SDK sample application built entirely without C# or Visual Basic.NET. Photoviewer performs a flickr search using Ruby and then displays the results using

Lightbox, an open source JavaScript project. This application very nicely illustrates the interoperability of Ruby, JavaScript and Silverlight.

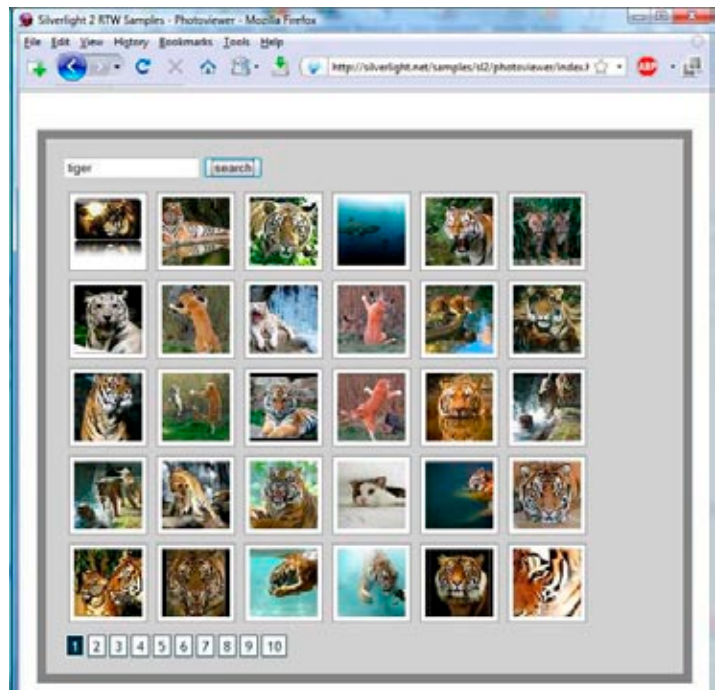


Figure 7. This screenshot shows the images returned from flickr from a search for "Tiger". Clicking on one of the images will display a larger size photo. Clicking on one of the numbered boxes at the bottom will display another set of 30 images.

Future Developments (C# 4.0)

Microsoft is currently working on some language improvements which will significantly simplify the syntax required to call dynamic language code from C#. Using a process similar to implicit typing, it will be possible to get a reference to a dynamic object by calling some method or function which returns a dynamic object.

```
Pseudo Code: dynamic d = Function-WhichReturnsADynamicObject();  
C# 4.0 Example: dynamic doc = Html-Page.Document.AsDynamic();
```

The significance of a dynamic object is that the C# 4.0 compiler will not check any method calls on a dynamic object at

compile time, thus deferring resolution until runtime. While this is standard procedure for dynamic languages, it is highly unusual for a static language like C#. In all other respects, however, C# will remain a static language. The purpose of this change is to facilitate C# interaction with these dynamic objects through syntax simplification which can be illustrated in the following code samples:

```
C# 3.x
ScriptObject map = win.CreateInstance("VEMap", "myMap");
map.Invoke("LoadMap");
map.Invoke("SetTitle", item.Title);
C# 4.0
dynamic map = win.New.VEMap("myMap");
map.LoadMap();
map.SetTitle(item.Title);
```

As you can see, methods belonging to a dynamic object can be called directly instead of through a call to `Invoke`, passing in the actual method as a string as is currently required. This deferral of method dispatch will extend to field and property accesses, indexer and operator calls as well as to delegate invocations. As we all know, cleaner simpler code is easier to write and less error prone. The mere existence of this project signals the importance which has been assigned to dynamic language development at Microsoft.

Summary

Dynamic language support in Silverlight turns out to be a major benefit for existing Python and Ruby developers because it allows them to write Python and Ruby code for web applications, which formerly required them to write JavaScript. As the Fractulator and Photoviewer sample applications illustrate, it is possible to create Silverlight applications without any C# or Visual Basic.NET code.

On the other hand, for the C# and Visual Basic.NET developer, dynamic language support creates the opportunity for a whole new set of developers to build Silverlight applications, many of which will be open source and available for inclusion in non-dynamic Silverlight applications. Additionally, where it is desirable to provide end users with built-in scripting capability, [this can best be achieved via the use of a dynamic language](#). Finally, since the demise of the HTML DOM and since the need for client side scripting does not appear to be on the visible horizon, further developments in scripting support promised for inclusion in the next release of the .NET Frame-

work looks to make the task of interoperability with scripting languages even simpler than it is today.

Special thanks to Michael Foord for making available to me a pre-publication copy of his excellent book, [IronPython in Action](#). In addition to a comprehensive explanation of IronPython in general--including how to take advantage of the .NET Framework, this book contains a very useful chapter outlining the use of IronPython in Silverlight. The book is now available. ■

Links

- [Python in your Browser with Silverlight 2](#) (Excellent introduction to this topic by Michael Foord)
- [Walk-through: Silverlight Flickr Client in IronRuby](#) (Jimmy Schementi Blog article)
- [Adding scripting to a C# Silverlight app](#) (Jimmy Schementi Blog article)
- [agdlr: Silverlight + DLR + Open Source](#) (Jimmy Schementi Blog article)
- [Silverline – Integrating Ruby on Rails and Silverlight](#) (A website maintained by Jimmy Schementi)
- [Silverlight Dynamic Languages in Visual Studio](#) (An explanation of how to use Chiron in Visual Studio by Laurence Moroney)
- [Dynamic Silverlight Part 1: Hello, World](#) (John Lam Blog article – MS IronRuby Dev Team Member. Caution – based on SL Beta)
- [Dynamic Silverlight Part 2: Managed JScript and flickr](#) (John Lam Blog article – MS IronRuby Dev Team Member. Caution--demo will not work because based on SL Beta)
- [Silverlight Minimal Examples](#) (Blog post illustrating using IronRuby to create simple Silverlight applications. Caution – based on SL Beta)
- [Natural Interop with Silverlight, Office and Python in Microsoft Visual C# and Microsoft Visual Basic](#) (Alex Turner PDC 2008 session)
- [Programming Silverlight with the CoreCLR](#) (MSDN article by Andrew Pardoe, Program Manager for the CLR)
- [A Great Set of IronRuby Tutorials](#) (self explanatory)
- [IronPython Cookbook](#) (a web site devoted to resources for IronPython)
- [New Features in C# 4.0](#) (a word document by Mads Torgersen, a C# Language Program Manager)
- [IronPython Studio](#) (explanation and download details)

SILVERLIGHT: ENABLING RICH INTERNET APPLICATIONS FOR THE OPEN SOURCE WORLD

* This article originally appeared in the RIA Development Center at <http://www.devx.com/RIA/Article/40465>

Cal Schrottenboer is a C# developer with experience in building Windows Forms application front ends for SQL Server databases. He also teaches programming classes at Foothill College in Los Altos Hills, California and Microsoft Network Administration (MCSE) classes at Mission College in Santa Clara. Cal maintains a WPF blog at www.WPFLearningExperience.com. His outside interests include travel and photography (www.travelswithcal.com).



PHP Developers Don't Need Silverlight – Or Do They?

Think PHP developers don't need Silverlight? Think again. By dynamically generating XAML (Extensible Application Markup Language), PHP developers can buff their skills with some very cool Microsoft® Silverlight™ tricks, adding a whole new responsive visual element to their PHP pages. This walkthrough goes through the details of building a simple Silverlight app just using XAML code, JavaScript, and PHP. by Justin Whitney

Silverlight gives a lot of love to .NET coders, especially those versed with Microsoft® development tools like Microsoft® Visual Studio® 2008 and the new Microsoft® Expression® Studio. But PHP developers have a lot to benefit from, as well. In fact, using the Silverlight runtime, XAML, PHP code, and a touch of JavaScript, you can add a lot of splash to your PHP apps.

To demonstrate the versatility of Silverlight as a browser plugin, I'm going to walk through the development of a simple app without using Microsoft development tools. This demo will focus on the Silverlight runtime's role as a XAML interpreter. Since it doesn't get into .NET development, you can do this with the Silverlight 1.0, as well as Silverlight 2.0 alpha.

The walkthrough will cover a few key principles that you can build on to create more robust PHP applications:

- The use of XAML in a Silverlight application.
- Delivering dynamic XAML using PHP.
- Adding mouse events to XAML code to create a highly responsive UI.

Note: the work here can be done in any text editor, including Notepad if that's your thing. Personally, I prefer [Editplus](#) –something nice and simple.

Step One: Create a Simple Page with a XAML Component
The first step will be the most complicated. Your goal here is just to get something up and running without using the Expression Studio.

First, make sure you have the latest Silverlight runtime installed. You can use the recent Silverlight 1.0 release, but since other articles on this site require Silverlight 2.0 alpha, you might as well install the latest [2.0 refresh](#).

Next, create a control panel in XAML format. Make a row of colored dots, such as Figure 1. (Confession #1: a variety of WYSIWYG XAML editors are in various stages of production and release. But for this demo I used [Microsoft® Expression® Design](#). After drawing a series of colored dots, I used the Export function, set it to Silverlight, and accepted all other defaults.)

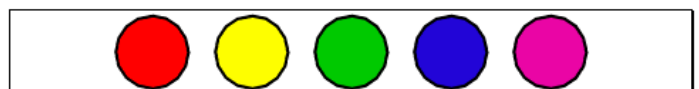


Figure 1. XAML Control Panel

Or, if you want, just copy and paste Listing 1 into a file called `controlPanel.xaml`.

Listing 1: controlPanel.xaml

```
<?xml version="1.0" encoding="utf-8"?>

<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="controlPanel_design">

  <Canvas x:Name="Layer_1" Width="640" Height="80" Canvas.Left="0" Canvas.Top="0">

    <Ellipse x:Name="Ellipse" Width="71" Height="71" Canvas.Left="99.5133"
  Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3" StrokeLineJoin="Round"
  Stroke="#FF000000" Fill="#FFFF0000"/>

    <Ellipse x:Name="Ellipse_0" Width="71" Height="71" Canvas.Left="192.86"
  Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3" StrokeLineJoin="Round"
  Stroke="#FF000000" Fill="#FFFFFF00"/>

    <Ellipse x:Name="Ellipse_1" Width="71" Height="71"
  Canvas.Left="286.207" Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3"
  StrokeLineJoin="Round" Stroke="#FF000000" Fill="#FF00C800"/>

    <Ellipse x:Name="Ellipse_2" Width="71" Height="71"
  Canvas.Left="379.553" Canvas.Top="5.5"
  Stretch="Fill" StrokeThickness="3" StrokeLineJoin="Round"
  Stroke="#FF000000" Fill="#FF230FD2"/>

    <Ellipse x:Name="Ellipse_3" Width="71" Height="71"
  Canvas.Left="472.9" Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3"
  StrokeLineJoin="Round" Stroke="#FF000000" Fill="#FFC80FA0"/>

  </Canvas>
</Canvas>
```

A couple of things to note about the XAML, because it'll come into play later:

- As you can see, it's just XML. As such, it can be delivered as any other XML document, as long as the document type is defined as such.
- Note how the image in Figure 1 is translated into nothing more than a series of Ellipse tags nested with a Canvas tag. As one might imagine, it shouldn't be too hard to generate something like this programmatically.

The key to using XAML on your page and making it a Silverlight application is the Silverlight.js file, which will be called on your page. Both Visual Studio 2008 and Expression Studio tools automatically create this file for inclusion in your web project. For this demo, you'll need to [download it here](#). You can read more about this file and some of the Silverlight plumbing that goes with it on MSDN's "[Instantiating a Silverlight Plug-in](#)."

Note: using this file independently like this is generally not advised because Microsoft frequently updates the JavaScript. In fact, Confession #2, to create this file, I started up a dummy Microsoft® Expression® Blend 2 project and copied its Silverlight.js into this demo.

You'll also need some additional JavaScript to instantiate the Silverlight object on your page. So create a new file called createSilverlight.js and include the code in Listing 2.

Listing 2: createSilverlight.js

```
function createControlPanel()
{
    Silverlight.createObjectEx({
        source: "controlPanel.xaml",
        parentElement: document.getElementById("dotControlPanel"),
        id: "slControlPanel",
        properties: {
            width: "100%",
            height: "100%",
            inplaceInstallPrompt:true,
            version: "1.0"
        }
    });
}

if (!window.Silverlight)
    window.Silverlight = {};

Silverlight.createDelegate = function(instance, method) {
    return function() {
        return method.apply(instance, arguments);
    }
}
```

Take a closer look at Listing 2 and you'll see a few things going on.

- The createSilverlight.js page is the standard file name for the instantiation code. It also usually holds the createSilverlight function. But as you can see here, the naming is up to you.
- Look at the Silverlight.createObjectEx call. For now, the source is a XAML file--the file you just created. But that source can actually be anything that returns XAML code, including a compiled library or a PHP script.
- parentElement specifies the DOM object in which the results will appear. You'll get to that in just a moment.
- id is the ID for this Silverlight control. When creating multiple controls, be sure this is a unique name.
- Especially take a look at the properties attribute--you can specify actual dimensions or just use 100% like I am here. But you should be very particular about your choice for "inplaceInstallPrompt" and the version. The install prompt is covered in more detail in the above-referenced [MSDN article](#). As to the version, as stated earlier this will work with the Silverlight 1.0 runtime, so you can specify 1.0 here. But when you build some .NET code, which is where the real power of Silverlight comes unleashed, be sure to specify 2.0 here instead.

To pull it all together, make your test page. Create a new file called PHPdemo.htm and add the code in Listing 3.

Listing 3: PHPdemo.htm

```
<html>
<head>
  <title>Silverlight PHP Demo</title>
  <script type="text/javascript" src="Silverlight.js"></script>
  <script type="text/javascript" src="createSilverlight.js"></script>
  <style type="text/css">
    .controlPanel {
      height: 80px;
      width: 640px;
    }
    .XAMLCanvas {
      height: 480px;
      width: 640px;
    }
  </style>
</head>

<body>
  <div id="dotControlPanel" class="controlPanel">
    <script type="text/javascript">
      createControlPanel();
    </script>
  </div>
  <div id="dotDisplayCanvas" class="XAMLCanvas">
  </div>
</body>
</html>
```

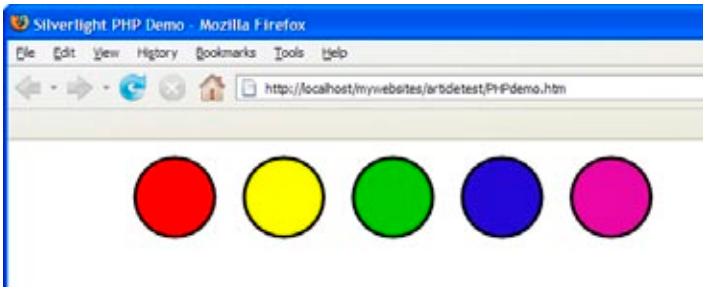


Figure 2. Simple Silverlight Page

Nothing too fancy here. You'll see a few things that aren't being used yet but for the most part, it's self-explanatory. You're pulling in the two JavaScript files, setting some dimensions via CSS, and then adding two divs, one for the control panel and one for the results. Within the div, you have a JavaScript call to the function that instantiates the Silverlight object.

Run this in a Web browser and you should see some Silverlight in action. Hopefully it'll look something like Figure 2.

Step Two: Add XAML Mouse Events

The bulk of the work is done. Now things get really interesting. The next few steps will be taken in pieces, each building on the last, so you can make sure everything's working before you get too far along. In this step, you'll add some mouse events to the XAML code from Listing 1.

Silverlight supports several events that you can raise in your XAML and handle via code, in this case JavaScript. For this demo, you'll play with one mouse event in particular: `MouseLeftButtonUp`, though it'd be good to review the others, such as `MouseMove`, `MouseEnter`, etc. These are described in more detail in "[Silverlight Mouse Support](#)."

Go back to the control panel XAML code in Listing 1. At the end of each ellipse tag, add the following:

```
MouseLeftButtonUp="dropDots"
```

In your `createSilverlight.js` file, add a new function:

```
function dropDots(sender, mouseEventArgs)
{
    alert('dots dropped!');
}
```

Reload the page. Now, when you click on one of the colored dots, you should see a popup.

Next, get a little fancier. Replace the alert statement you just wrote with the following:

```
sender.fill = "red";
```


This is just a quick demo to show how easy it is to interact with XAML. Reload the page and you should be able to turn each dot red by clicking on it. The "[Scripting and Mouse Events](#)" QuickStart on the Silverlight site walks you through a lot more of this type of thing.

Now replace the `sender.fill` assignment with a new statement:

```
alert(sender.Fill.Color);
```

This returns the fill color of the circle as a Long value. That's good enough for now.

Step Three: Add PHP as XAML Source

So where does PHP come into play? As mentioned earlier, you can use it as the source for a Silverlight object. In fact, for now, just to get a working example up and running, use the same XAML source as the control panel you've already created. Save the code in Listing 4 to "drawDots.php".

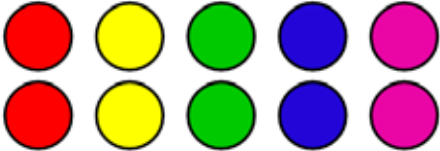
Listing 4: drawDots.php

```
<?
header('Content-type: text/xml');
?>
<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="controlPanel_design">
  <Canvas x:Name="Layer_1" Width="640" Height="80" Canvas.Left="0"
Canvas.Top="0">
    <Ellipse x:Name="Ellipse" Width="71" Height="71" Canvas.Left="99.5133"
Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3" StrokeLineJoin="Round"
Stroke="#FF000000" Fill="#FFFF0000"/>
    <Ellipse x:Name="Ellipse_0" Width="71" Height="71"
Canvas.Left="192.86" Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3"
StrokeLineJoin="Round" Stroke="#FF000000" Fill="#FFFFFFF0"/>
    <Ellipse x:Name="Ellipse_1" Width="71" Height="71"
Canvas.Left="286.207" Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3"
StrokeLineJoin="Round" Stroke="#FF000000" Fill="#FF00C800"/>
    <Ellipse x:Name="Ellipse_2" Width="71" Height="71"
Canvas.Left="379.553" Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3"
StrokeLineJoin="Round" Stroke="#FF000000" Fill="#FF230FD2"/>
    <Ellipse x:Name="Ellipse_3" Width="71" Height="71"
Canvas.Left="472.9" Canvas.Top="5.5" Stretch="Fill" StrokeThickness="3"
```

```
StrokeLineJoin="Round" Stroke="#FF000000" Fill="#FFC80FA0"/>  
</Canvas>  
</Canvas>
```

Next, add a new function to createSilverlight.js:

```
function createDisplayCanvas()  
{  
    Silverlight.createObjectEx({  
        source: "drawDots.php",  
        parentElement: document.getElementById("dotDisplayCanvas"),  
        id: "slDisplayCanvas",  
        properties: {  
            width: "100%",  
            height: "100%",  
            inplaceInstallPrompt:true,  
            version: "1.0"  
        }  
    });  
}
```



```
}
```

Figure 3. PHP as XAML Source

Last, in PHPDemo.htm, replace

```
<div id="dotDisplayCanvas" class="XAMLCanvas">  
</div>
```

Load the page and you should see a double row of colored dots, as in Figure 3. with

```
<div id="dotDisplayCanvas" class="XAMLCanvas">  
    <script type="text/javascript">  
        createDisplayCanvas();  
    </script>  
</div>
```

Not particularly inventive, but it gets the point across.

Step 4: Create Dynamic XAML with PHP

Now that all the pieces are in place, you can add the final code. In Step 2 you created some mouse events. In Step 3 you created some PHP that displays to your canvas area. You can probably guess what will happen next.

For this demo, your control panel will have one and only one function: add colored dots. You have many options for doing this, but the one I'll cover here is the use of a URL string to pass a concatenated JavaScript variable. In summary:

- Every time you click on a dot, the JavaScript will add a new value to a string.
- That string will then be passed to your PHP script.
- The PHP script explodes the string into an array.
- Based on the array length and values, it will return a set of dots.

To start, the createSilverlight.js file needs a bit of code. While colors can be passed as common names, I prefer working with hex values, which takes a little more work. As you saw earlier, the Fill.Color value returns a long. To convert that value to hex, add the following function (credit goes to Suyog Kale for [posting this routine](#)):

```
function getFillColor(color)
{
    var fill = color;
    if (fill < 0) {
        fill = (16777216 + parseInt(fill));
    }
    fill = fill.toString(16);
    if (fill.length > 6) {
        fill = fill.substr(fill.length - 6);
    }
    while (fill.length < 6) {
        fill = "0" + fill;
    }
    fill = "ff" + fill;
    return fill;
}
```

Note the last addition to the string: "ff". Silverlight works with 8-digit hex values for color, comprised of the 6-digit value preceded by "ff".

Replace the dropDots() function entirely with the following:

```
var dotColors="";

function dropDots(sender, mouseEventArgs)
{
    if (dotColors != "")
    {
        dotColors += ",";
    }
}
```

```
}
dotColors += getFillColor(sender.Fill.Color);
createDisplayCanvas();
}
```

Note the new global var "dotColors".
In the createDisplayCanvas() function, replace
source: "drawDots.php",
with

```
source: "drawDots.php?colors=" + dotColors,
```

At this point, createSilverlight.js should look something like Listing 5.

Listing 5: Completed createSilverlight.js

```
function createControlPanel()
{
    Silverlight.createObjectEx({
        source: "controlPanel.xaml",
        parentElement: document.getElementById("dotControlPanel"),
        id: "slControlPanel",
        properties: {
            width: "100%",
            height: "100%",
            inplaceInstallPrompt:true,
            version: "1.0"
        }
    });
}

if (!window.Silverlight)
    window.Silverlight = {};

Silverlight.createDelegate = function(instance, method) {
    return function() {
        return method.apply(instance, arguments);
    }
}

var dotColors="";

function dropDots(sender, mouseEventArgs)
{
    if (dotColors != "")
    {
```

```
        dotColors += ",";
    }
    dotColors += getFillColor(sender.Fill.Color);
    createDisplayCanvas();
}

function createDisplayCanvas()
{
    Silverlight.createObjectEx({
    source: "drawDots.php?colors=" + dotColors,
    parentElement: document.getElementById("dotDisplayCanvas"),
    id: "slDisplayCanvas",
    properties: {
        width: "100%",
        height: "100%",
        inplaceInstallPrompt:true,
        version: "1.0"
    }
    });
}

function getFillColor(color)
{
    var fill = color;
    if (fill < 0) {
        fill = (16777216 + parseInt(fill));
    }
    fill = fill.toString(16);
    if (fill.length > 6) {
        fill = fill.substr(fill.length - 6);
    }
    while (fill.length < 6) {
        fill = "0" + fill;
    }
    fill = "ff" + fill;
    return fill;
}
```

Replace the contents of drawDots.php entirely with Listing 6.

Listing 6: Completed drawDots.php

```
<?
header('Content-type: text/xml');
?>

<?
$arrColors = explode(",",$_GET["colors"]);
srand(time());
```



```
?>

<Canvas xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" x:Name="resultCanvas">

  <Canvas x:Name="Layer_1" Width="640" Height="80" Canvas.Left="0"

  Canvas.Top="0">

  <?
  $n = 0;
  foreach ($arrColors as $c) {
    $n++;
    $lft = (rand()%530);
    $top = (rand()%370);
    echo('<Ellipse x:Name="Ellipse_' . $n . '" Width="100" Height="100"

  Canvas.Left="' . $lft . '" Canvas.Top="' . $top . '" Stretch="Fill" Fill="#" . $c . '>');
  }
  ?>

  </Canvas>
</Canvas>
```

As you can see, mouse events in the XAML file trigger a JavaScript routine which concatenates a global variable with the fill color of the selected dot. It then redraws the canvas by calling the `createDisplayCanvas()` function, which passes this string via GET to the PHP page.

The new PHP script explodes the GET string into an array, which can then be looped through via `foreach`. As an added bonus, I've added a bit of randomness to the equation by also setting `Canvas.Left` and `Canvas.Top` to random values within the `Ellipse` XAML echo. This serves no practical function whatsoever other than, hopefully, to stimulate the imagination a bit and induce more creative and interesting demonstrations of the technique.

The final result can be viewed here: www.justinwhitney.com/slphpdemo/PHPdemo.htm. You can also download a zip of the [final source code here](#).

Where to Go From Here

This walkthrough barely scratches the surface of what you can do with Silverlight and PHP. Below are a few additional resources for investigating the topic further. If you come up with something cool, post it and send us a link. ■

- [Silverlight Downloads](#)
- [Delivering Parameterized Silverlight Content with PHP](#)
- [Silverlight with PHP](#)
- [Flickr Example](#)
- [Silverlight / PHP / MySQL / Linux / Firefox Demo](#)
- [Instantiating a Silverlight Plug-in \(Using CreateSilverlight.js and Silverlight.js\)](#)
- [Silverlight QuickStart: Scripting and Mouse Events](#)
- [Binding a XAML Application to DataBase Data](#)

* This article originally appeared in the RIA Development Center at <http://www.devx.com/RIA/Article/36022>

Justin Whitney is a regular contributor to DevX.com. He currently lives in San Francisco, where he consults for leading high-tech firms and writes about emerging technologies.



Expression Web 2 for PHP Developers – Simplifying Your PHP Applications.

Some of the most important new features in Microsoft's recently released Expression Web 2 involve enhanced support for PHP. Don't think this is just a half hearted effort to appeal to the Open Source web development crowd. Expression Web 2 supports PHP developers with a carefully constructed, feature full treatment which you should seriously consider for your PHP applications. by Cal Schrottenboer

Built-In Development Server with Integrated Browser Support
Speed and efficiency matter a lot with all forms of application development. Testing any PHP document during development obviously requires a web server, either local or remote. The disadvantage of using a remote web server is the inconvenience and time required to upload new and changed files to the remote server. For this reason, developers generally install a web server on their development computer. However, full fledged web servers such as Apache and IIS are complex applications and required configuration tasks can be rather challenging. Moreover, the use of a separate web server for testing development pages requires manual switching back and forth between multiple applications.

In steps Microsoft's Expression Web 2 (EW2).

For starters, EW2 has a built-in web server that is capable of displaying PHP pages and is virtually configuration free. (Expression Web 1 also included a built-in web server but it wasn't capable of displaying PHP pages.)

To take advantage of this all you need to do is specify the location on your hard disk of the PHP runtime to use for processing PHP content in your web pages. But the real strength of EW2's built-in web server is its integration with different browsers. When you select Preview in Browser from the EW2 menu, you are presented with the choice of various browsers--including browsers preconfigured to different window sizes--in which to view your page. (Of course, this list of browsers is limited to those which you have actually installed on your computer.) You can even make a single selection to simultaneously launch your page in multiple browsers (see Figure 1).

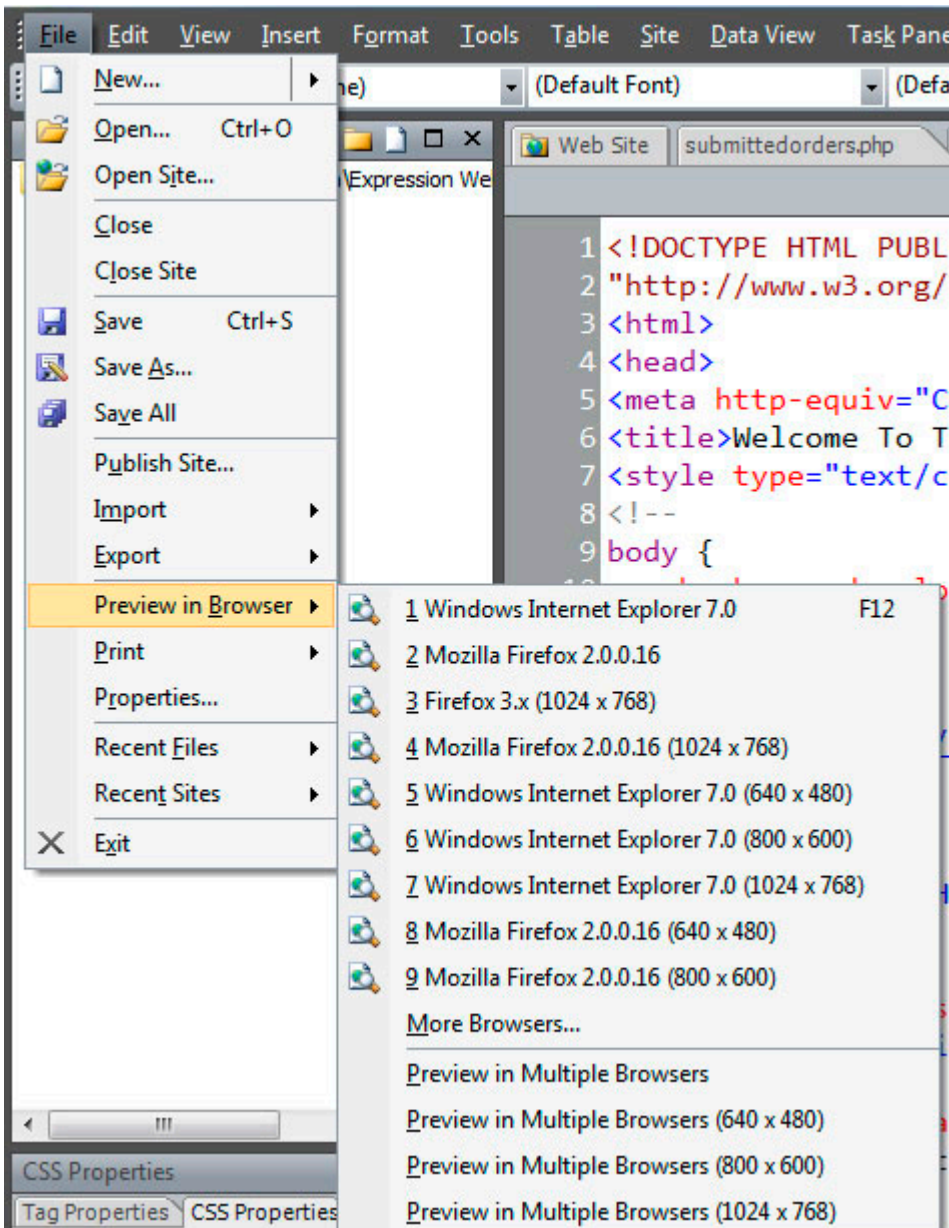


Figure 1. Expression Web 2 makes it easy for you to test your PHP application in multiple browsers and at multiple resolutions.

As is readily obvious, comparing the treatment of a given web page in different browsers and at different window sizes couldn't be easier. The F12 hotkey makes this even more convenient by previewing the selected page in the most recently used browser. A developer can add any number of additional browsers to the original list by selecting the Edit Browser List

entry. (Of course, you can only add a browser to this list once it has been installed on your computer.) This linked [screencast](#) illustrates the use of the EW2 built in web server.

PHP Includes and Design View Integration

Many PHP developers use Dreamweaver as their primary development tool. After the release of Expression Web 1 (EW1), various articles appeared comparing the strengths and weaknesses of Dreamweaver and Expression Web from the point of view of a PHP developer. One deficiency of EW1 frequently pointed out in comparisons of this type was the inability of EW1 to display PHP includes in design view. Part of my preparation for this article included the creation of a relatively [simple website](#) for a friend using PHP. I started out by creating a logo and a menu structure. Since I wanted every page in this website to include both of these items, I decided to use the technique of a PHP include. So I extracted the common elements from my home page design and saved it in a separate PHP file. Incorporating these common elements in each page simply required adding the following line of code: `<?php include('PSHeader.php'); ?>`. Even though these common elements are actually contained in a separate file, EW2's design view displays a composite view which is indistinguishable from that which would be generated if all the content were actually contained in a single file. Figure 2 shows what elements are used in the include and what is unique to the page.



Figure 2. Displaying a common header using a PHP include is easy with Expression Web 2.

This linked [screencast](#) illustrates how simple it is to use PHP includes.

Byte Order Marks (Problem Solved)

The initial release of Expression Web (January 2006) was targeted at ASP.NET developers and apparently the EW development team was unable to also manage to include optimizing the application for PHP developers in the first edition. Not surprisingly therefore, those PHP developers who tried to use Expression Web for their projects encountered a number of problems. For example, [some developers discovered](#) that EW 1 added a **Byte Order Mark (BOM)** to all of the HTML, PHP and CSS files which it creates using UTF-8 encoding. (The purpose of the **BOM** is to specify whether the least significant byte is placed first--as required by Intel and MIPS processors--or last--as required by Motorola processors.) The practical effect of these BOMs in PHP files was to frequently [prevent PHP includes from working properly when the referencing page was delivered from a web server](#).

The Byte Order Mark problem has now been eliminated in EW2 by the simple step of giving the user complete control over whether or not to include a BOM in any new document. The applicable settings allow this decision to be made on a file type by file type basis (Figure 3). Moreover, Microsoft set BOMs to OFF as the default for .PHP files. (The ON setting remains the default for all other common web project file types such as .HTM, .HTML, .CSS and .XML.)

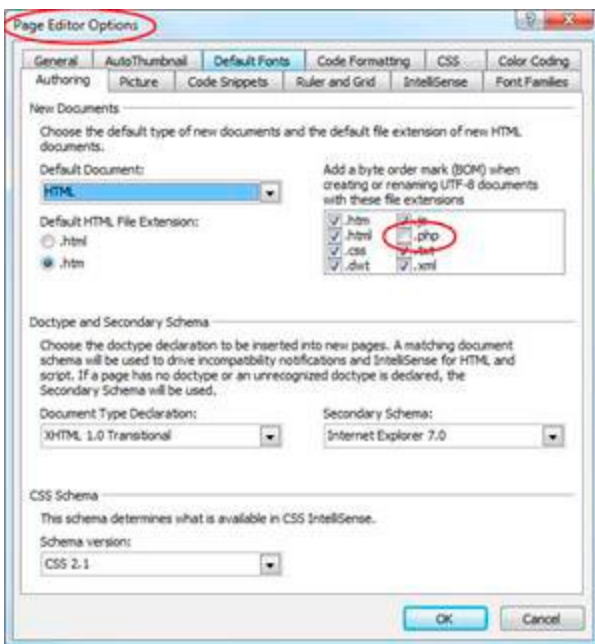


Figure 3. You can now set Byte Order Marks on each individual file type or document in Expression Web 2.

EW 2 even allows the BOM setting to be overridden on a document by document basis as the developer sees fit. You can access a document's properties by right clicking on the document in the editor window and selecting Encoding. HTML encoding settings can be found on the Language tab. As shown below, including a BOM is only allowed when UTF-8 encoding has been selected (Figure 4).

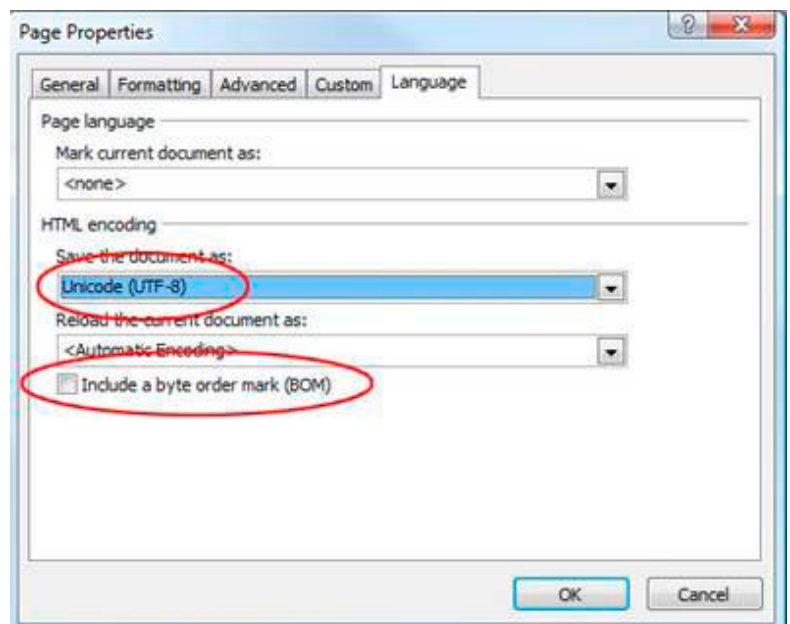
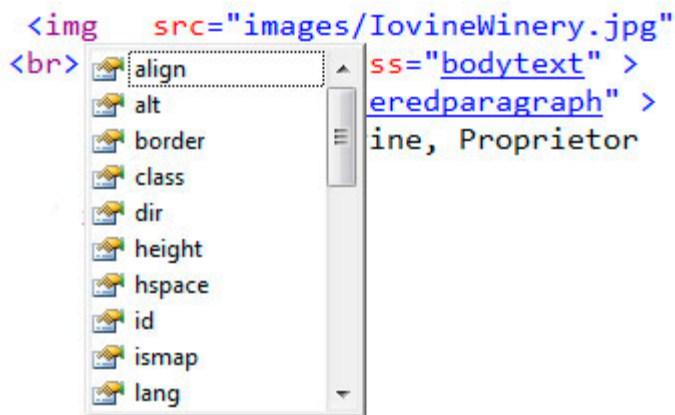


Figure 4. Byte order marks can only be set on UTF-8 documents.

This linked [screencast](#) illustrates both how BOMs in a PHP document can cause problems for PHP includes and also shows how to access the EW settings to control the inclusion of BOMs.

IntelliSense for PHP

IntelliSense (Microsoft's version of code completion) has long been one of the most appreciated features of Visual Studio and EW1 also featured IntelliSense--just not for PHP. With IntelliSense, whenever you are in a code block, IntelliSense will display all of your relevant choices, as for example a list of properties (or attributes), methods or events.



Not only does this save you typing and eliminate the possibility of spelling errors, it steers you away from options which are not valid in the current context and offers suggestions which you may not have considered. As this linked [screencast](#) shows, this feature is also extremely helpful in learning (or remembering) a new technology because it will also point out where a particular construction can be used but is not considered optimal. So if you are not an expert (or even just a little rusty) in XHTML or CSS, this feature can help guide you to a more maintainable, professional result.

IntelliSense is especially helpful for built in PHP functions since it will show the list of required parameters along with their appropriate data types.

```
<?php include('PSHeader.php ?>
include (string $filename)

if (mail ($to, $subject, $body, "From: Cal Schrottenboer PHP <$from> \r\n" ))
{
mail (string $to, string $subject, string $message, [string $additional_headers], [string $additional_parameters])
}
```

PHP Snippets and Built-in FTP Client

PHP snippets represent common PHP code items which can be inserted into your PHP document simply by selecting them from a menu list. For example, EW2 can write for you the entire PHP include (shown above) if you choose it from the Insert menu and then navigate to the file which you want to include. This linked [screencast](#) illustrates the various types of PHP snippets available in EW2.

EW2 also makes it easy to upload your final work product to a remote web server. Just specify the URL of the remote web site and provide the appropriate credentials and EW2 will open a standard FTP client connection to your remote web site. One interesting feature is that EW2 can compare the files in your development web site folder to the files at your remote web site to determine which ones are newer and/or different. Then should you choose, only new and modified files need be uploaded to the remote web server (Figure 5).

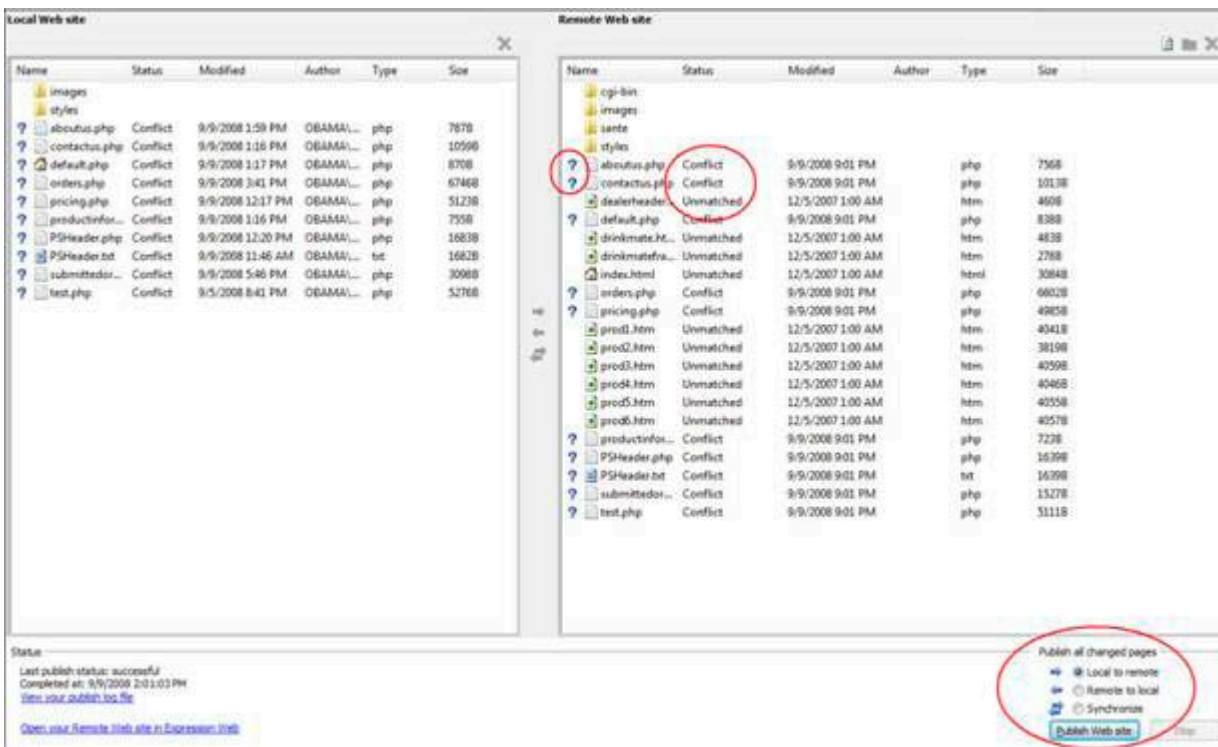


Figure 5. Expression Web's FTP client makes it easy to spot and resolve file differences.

CSS Support

CSS has long been established as the appropriate technique for separating content from formatting. However, the application of CSS can be extremely complex because of the potential interaction of one or more external style sheets, one or more embedded style sheets as well as inline styles (some direct and some inherited from parent elements). EW2's method of displaying the hierarchy of applicable styles and settings is capable of making sense of all these overlapping and potentially conflicting settings no matter how complex the applicable structure. The following screenshot shows the hierarchy of applicable styles in the section entitled Applied Rules (Figure 6). Settings from the selected style (.standardparagraph in this case) are shown in the section entitled CSS Properties. Those settings which remain at their defaults are shown as blank in CSS Properties.

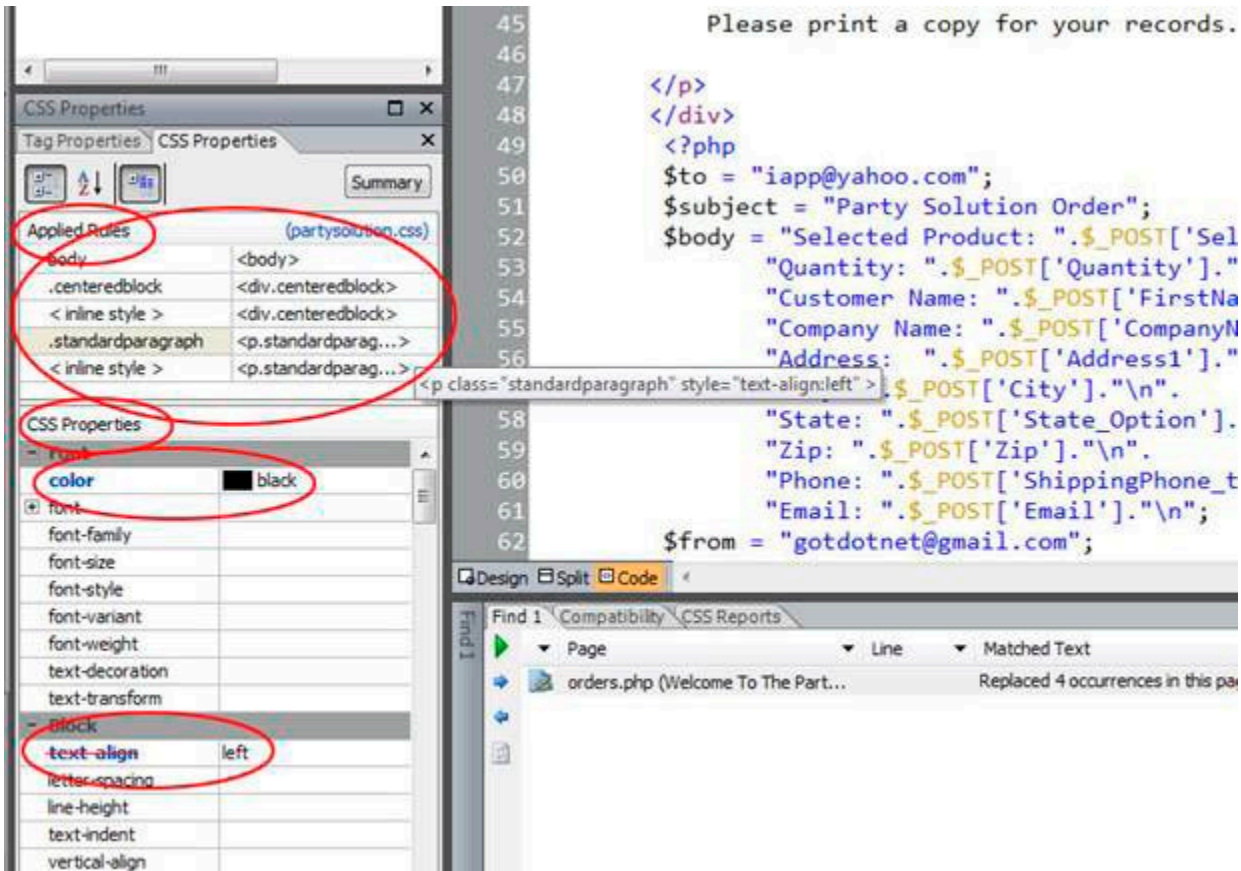
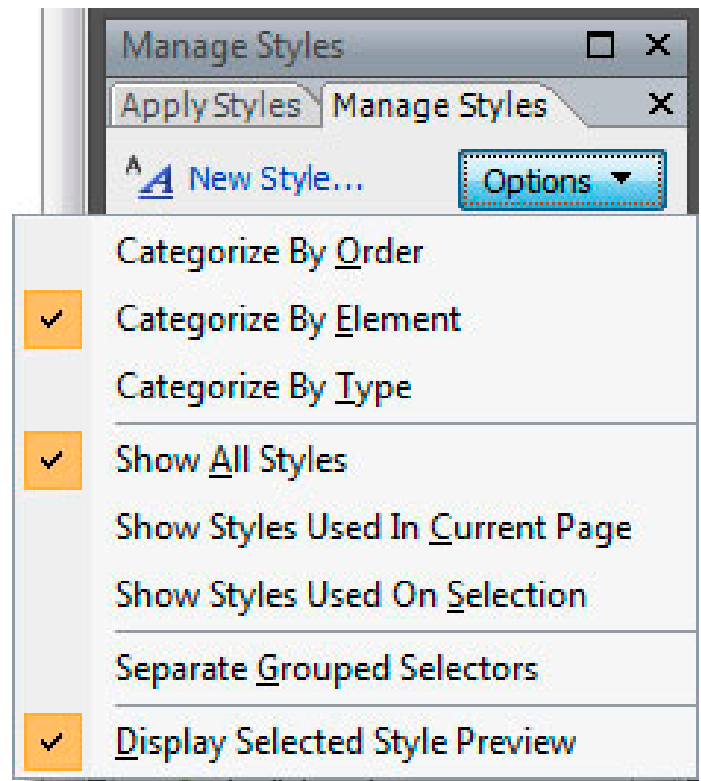
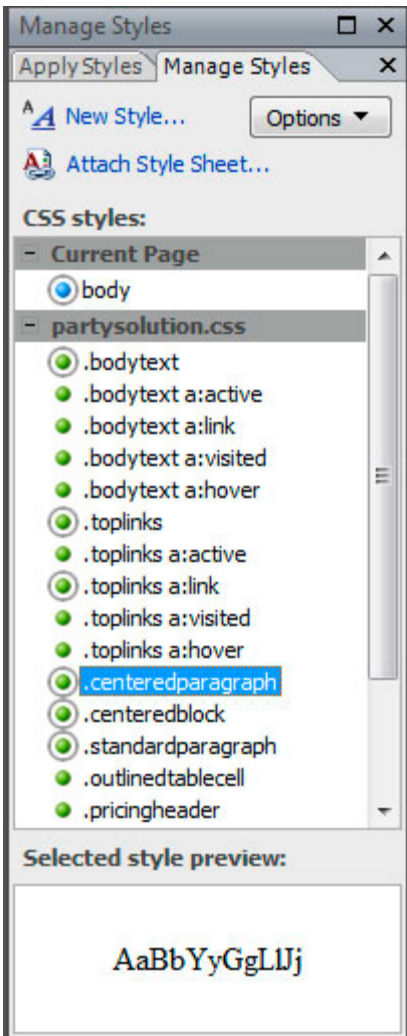


Figure 6. Managing and selecting the appropriate style sheet is simplified with Expression Web 2.

Locating a style which you want to apply to a particular element is facilitated by the Manage Styles task pane. Here you can find a list of all available style elements, arranged either by the order in which they appear within their style, by element or by type.

A number of filtering options are also supported.



Add to this the ability for a developer to easily create new style sheets and to automatically link new or existing style sheets and EW2's treatment of CSS becomes extremely rich.

XHTML Syntax Support and Code Coloring

Occasional web developers and anyone who has not yet memorized every detail of the latest XHTML standard (currently 4.01 Transitional) are certain to appreciate that whenever you enter a construct which does not comply with the selected standard, an error message is displayed warning of the standards violation.

```
<div class="centeredblock" >
  
</div>
<br>
  <div class="bodytext" >
    <p class="centeredparagraph" >
      Michael Iovine, Proprietor
```

The World Wide Web Consortium now regards the attribute 'align' as outdated. Newer constructs are recommended.

These messages also include an alert if a required setting is missing.

```

</div> All <img> tags must have an alt attribute.
<br> <div class="bodytext" >
```

Colored code has long been recognized as a great aide to reading, understanding and debugging code. EW2 now extends the color coding previously available in EW1 to PHP code as well. The following screenshots illustrate how much more difficult it is to interpret non-color coded content (Figure 7).

```
</style>
<link href="styles/partysolution.css" rel="stylesheet" type="text/css">
</head>
<body>

<?php include('PSHeader.php'); ?>
<br><br>
<div class="bodytext" >
  <p class="centeredparagraph" >
    Thank you <?php echo $_POST['FirstName'] ?> <?php echo $_POST['LastName'] ?> for your order!
  </p>
</div>
<div class="centeredblock" style="width:600px;font-size: 14px; font-family: Verdana, Helvetica, sans-serif, Copperplate; line-height: 16px;" >
<p class="standardparagraph" style="text-align:left" >
  Your Order: <br><br>
  Selected Product: <?php echo $_POST['SelectedProduct_Option'] ?>
  <br>
  Product Color: <?php echo $_POST['ProductColor_Option'] ?>
  <br>
  Quantity: <?php echo $_POST['Quantity'] ?>
  <br>
  Company Name: <?php echo $_POST['CompanyName'] ?>
  <br>
  Address: <?php echo $_POST['Address1'] ?> <?php echo $_POST['Address2'] ?>
  <br>
  City, State, Zip: <?php echo $_POST['City'] ?>, <?php echo $_POST['State_Option'] ?>, <?php echo $_POST['Zip'] ?>
  <br>
  Phone: <?php echo $_POST['ShippingPhone_tel1'] ?>-<?php echo $_POST['ShippingPhone_tel2'] ?>-<?php echo $_POST['ShippingPhone_tel3'] ?>
  <br>
  Email: <?php echo $_POST['Email'] ?>
  <br><br>
  Please print a copy for your records. You will be contacted shortly.
</p>
</div>
```

```
1 </style>
2 <link href="styles/partysolution.css" rel="stylesheet" type="text/css">
3 </head>
4 <body>
5
6 <?php include('PSHeader.php'); ?>
7 <br><br>
8 <div class="bodytext" >
9 <p class="centeredparagraph" >
10 Thank you <?php echo $_POST['FirstName'] ?> <?php echo $_POST['LastName'] ?> for your order!
11 </p>
12
13 </div>
14 <div class="centeredblock" style="width:600px;font-size: 14px; font-family: Verdana, Helvetica, sans-serif, Copperplate; line-height: 16px;" >
15 <p class="standardparagraph" style="text-align:left" >
16 Your Order: <br><br>
17 Selected Product: <?php echo $_POST['SelectedProduct_Option'] ?>
18 <br>
19 Product Color: <?php echo $_POST['ProductColor_Option'] ?>
20 <br>
21 Quantity: <?php echo $_POST['Quantity'] ?>
22 <br>
23 Company Name: <?php echo $_POST['CompanyName'] ?>
24 <br>
25 Address: <?php echo $_POST['Address1'] ?> <?php echo $_POST['Address2'] ?>
26 <br>
27 City, State, Zip: <?php echo $_POST['City'] ?>, <?php echo $_POST['State_Option'] ?>, <?php echo $_POST['Zip'] ?>
28 <br>
29 Phone: <?php echo $_POST['ShippingPhone_tel1'] ?>-<?php echo $_POST['ShippingPhone_tel2'] ?>-<?php echo $_POST['ShippingPhone_tel3'] ?>
30 <br>
31 Email: <?php echo $_POST['Email'] ?>
32 <br><br>
33 Please print a copy for your records. You will be contacted shortly.
34
35 </p>
36 </div>
```

Figure 7. Color coded vs. monochromatic PHP code in Expression Web 2.

Conclusion

I recall attending the launch for EW1 in San Francisco where one of the attendees was inquiring as to why a particular treasured feature which had been present in FrontPage seemed to be missing from Expression Web. The presenter responded with bad news for the questioner. That particular feature had been intentionally removed because it was not standards compliant and one of the overriding design goals for EW was to maximize standards compliance. Do not expect that feature to return, the questioner was told.

This principle has also been a guiding light for EW2 and, quite significantly, its scope has now been broadened to bring this standards compliant approach to PHP as well as to Microsoft's own ASP.NET technology. As should now be clear, most PHP developers who try it out are highly likely to be very impressed with Expression Web 2. I know that I am. ■

Related Resources

- [Introducing PHP Support in Expression Web 2](#) (appearing in the Microsoft Expression newsletter)
- [List of New Features in Expression Web 2](#) (from the Expression Web blog)

* This article originally appeared in the RIA Development Center at <http://www.devx.com/RIA/Article/39138>

Cal Schrottenboer is a C# developer with experience in building Windows Forms application front ends for SQL Server databases. He also teaches programming classes at Foothill College in Los Altos Hills, California and Microsoft Network Administration (MCSE) classes at Mission College in Santa Clara. Cal maintains a WPF blog at www.WPFLearningExperience.com. His outside interests include travel and photography (www.travelswithcal.com).



Microsoft®
Silverlight™

Light up the Web

Create next generation Web applications.

Microsoft® Silverlight™ is a cross-browser, cross-platform, and cross-device plug-in for delivering the next generation of media experiences and rich interactive applications for the Web.

Learn more and get started with Silverlight today: www.silverlight.net

Microsoft®