Part I: Silverlight Fundamentals for ASP.NET Developers

Chapter 1: Silverlight in a Nutshell

Chapter 2: Silverlight Architecture

Chapter 3: XAML Condensed

Chapter 4: Programming Silverlight

1

Silverlight in a Nutshell

This chapter is intended to give you a clear overview of Silverlight with the aim of helping you differentiate it from existing technologies and capabilities, as well as help you to understand when to use Silverlight and what to use it for. An overview of the required development environment is also shown toward the end of this chapter. If you are familiar with the general Silverlight principles, you can skip this chapter and move onto the more in-depth architecture chapter coming up next.

Uphill Struggle

As any ASP.NET developer will tell you, delivering a rich and engaging user interface via a browser is always a challenge when compared with doing the same thing in a classic rich-client application. Don't get me wrong — using ASP.NET enables you to create robust, enterprise-ready web applications. These same applications can, if written appropriately, scale to serve enormous numbers of users while providing a good-looking and logical user interface (with the backing of a good design time).

But creating something more than just a functional user interface, creating a user interface that actually excites and drives the user, creating something that leaps out and wows the user, has always been an uphill struggle because a standard web application simply cannot take advantage of the client's processing power to support a rich and powerful user interface (UI).

Trying to develop a rich user interface using only HTML and JavaScript (DHTML) can get you some great results, but managing and writing the amount of script required for truly advanced scenarios is difficult in itself as the cross-platform, cross-browser disconnected environment makes development even more error-prone and challenging. Couple this with managing thousands of lines' worth of supporting JavaScript, and you've got yourself a real headache.

Rich Client or Web Reach?

Because of the difficult nature of producing complex, highly interactive web applications, there has always been the trade-off of "rich versus reach." *Rich* refers to a traditional client application that has full access to the host operating system, API, and processing power and can therefore support an inherently richer user experience. *Reach* refers to web-based applications that are centrally deployed to potentially limitless numbers of users running different operating systems and software, but that cannot take advantage of the clients' full potentials to create a truly rich UI.

So, typically, web application developers have had to contend with finding a happy medium between *rich* and *reach*, delivering an application that can be easily deployed to many thousands or even millions of users but that is ultimately lacking in terms of richness of UI.

Up until now, the main solution to providing richer content via the Web was to use *Macromedia Flash*, a term that encompasses both the Flash Player (a cross-browser plug-in to display Flash content) and the development environment with which to author Flash content. The big drawback with this approach is the time needed to learn to develop in the Flash environment, including learning Flash ActionScript as well as keeping abreast of developments with ASP.NET — no mean feat. In point of fact, it's rare to find a single web developer who is both well-versed in Flash and well-versed in ASP.NET; therefore, when using both technologies, multiple developers are usually required.

Java has also been the tool of choice, as well as Flash, for delivering rich UIs embedded into the browser, but, again, this poses the same issues to an ASP.NET developer that using Flash does — inherently different technologies mixed together to produce the final output, requiring different skill sets and longer development cycles.

Silverlight Steps In

Silverlight 2 is a cross-platform, cross-browser plug-in that supports a stripped-down version of the .NET Framework API for programming Rich Internet Applications (RIA). Silverlight enables you to create visually stunning applications using a development environment and experience akin to that of Windows Presentation Foundation: UIs can be laid out and created using the declarative programming model provided by XAML and then brought to life using the power of the .NET Framework to drive it.

> **The term *Rich Internet Application* applies to any web application that has rich, desktop-like functionality. In effect, the web application feels and acts like a fat-client application. In the majority of RIA applications, this richness of functionality is provided via AJAX. However, it also encompasses Java, Flash, and, moving forward, Silverlight-enabled applications.**

Some of the high-level features provided in Silverlight 2 include:

- **Cross-Platform Support** — Silverlight provides true cross-browser and cross-platform support, running in all popular web browsers (IE, Firefox, Safari, and Opera) and on both Microsoft Windows and Apple Mac OS X platforms. A Silverlight application will run consistently across all these browsers and platforms, leaving you free to concentrate on designing and developing

the core of your application without worrying about conversion changes to implement. A third-party implementation named *Moonlight* has also been developed to allow Silverlight to run under GNU/Linux.

- **Mobile Support** — Silverlight has initial support for Windows Mobile and Nokia S60 devices.
- **Easy Installation** — The Silverlight plug-in is supported by a lightweight download that can install in seconds.
- **Streaming Media** — You can stream audio and video, from mobile devices to HDTV video.
- **DRM** — Silverlight has support for Digital Rights Management of media files.
- **AJAX-Style Updating** — There is no need to refresh the page for changes.
- **WPF-Like Graphics** — Access to a powerful graphics system with Windows Presentation Foundation (WPF)-like support
- **.NET Framework** — Silverlight is based on a subset of the .NET Framework — therefore a familiar development environment. Silverlight applications can be written in both C# and Visual Basic .NET.
- **Rich Control Library** — Silverlight comes with a plethora of UI controls with support for data-binding and automatic layout.
- **DLR** — Support for dynamic languages like Ruby and Python has also been included, operating under the Dynamic Language Runtime (DLR).
- **LINQ** — Silverlight includes support for language-integrated query, allowing you to program your data access code using a native syntax and strongly typed objects.
- **Communications** — Silverlight includes a host of communications options, allowing you full access to any server-based resources you have via XML Web Services, WCF Services, REST, and ADO.NET Data Services.
- **JavaScript Extensions** — Silverlight provides extensions to the standard JavaScript language to allow more control over the browser UI and hook-ins to work with the UI elements.
- **HTML/Managed Code Bridge** — Silverlight allows interaction between HTML and Managed Code, and vice versa.

If the features above weren't enough, another key selling point for Silverlight is that it's primarily built upon existing technologies and so should feel at least somewhat familiar to anyone who has used .NET, and even more familiar to anyone who has used .NET 3.0 or 3.5. Also, as the native development environment is Visual Studio it shouldn't pose any problems for .NET developers either. This all adds up to an incredibly fast ramp-up time for existing .NET developers once the initial setup and syntax queries have been brushed aside, and therefore a potential lower initial development cost as opposed to that for .NET developers taking on Flash or Java, for instance.

All that is required to run Silverlight in your browser is the Silverlight plug-in, which is a completely free download from Microsoft. If users do not have the plug-in installed and they navigate to a page hosting a Silverlight application, they will be automatically prompted to install it. Because of its small size, on most user connections this will take only seconds to complete.

The Impact of Silverlight on Your Existing ASP.NET Real Estate

Silverlight is all about delivering next-generation media experiences and rich Internet applications (RIAs) via the Web. It allows you to easily add video, animation, and improved interactivity to your web sites, delivering a more intense and consuming experience for your users. Silverlight provides a unified media format that scales from high definition to mobile using WMV and also supports WMA and MP3 for audio. Vector-based graphics are also catered for out-of-the-box, allowing your graphics and animations to scale to any size without losing quality. All this adds up to a much richer, more immersive UI than you can put together with DHTML alone. And to make it even easier to pick up and run with, Silverlight streaming by Windows Media Live provides a free streaming and application hosting solution enabling you to deliver your media-enabled RIAs with ease.

But if you decide to replace large chunks of your current real estate with Silverlight, will it affect the discoverability of your application by search engines? As the user interface of Silverlight applications is defined in text-based XAML, they can still be indexed and searched easily promoting their discoverability via search engines, so this shouldn't be a problem.

If you currently use JavaScript heavily to create a complex UI on the client, Silverlight can be used to replace this with one that not only performs better, but is easier to create and maintain thanks to a XAML-defined UI and type-safe .NET code-behind. And the same code will work cross-browser, cross-platform, saving you the headache of writing custom code for each scenario.

And if your web site relies heavily on advertising, imagine having full ad insertion capabilities at your fingertips, including the ability to deliver broadcast-style video and animated advertisements without loss of motion quality or visual fidelity.

One of the overlooked capabilities that Silverlight can provide you with is a new mechanism for delivering your applications via *Software as a Service* (SaaS). This term basically refers to a web-native application hosted on the Internet for use by paying customers — so they pay for using it, not owning it. As Silverlight helps you develop incredibly rich UIs, it will make it much easier for you to develop and provide applications that can be delivered in this manner, especially with the free hosting offered via Windows Live.

In short, Silverlight will give you the ability to add the wow factor to your ASP.NET applications and give you the ability to add it with relative ease.

What You Should Still Do in ASP.NET

As you're now aware, Silverlight brings a wealth of functionality to the table, but this isn't to say that every ASP.NET application you write from now on should simply be a container for a Silverlight application providing the full site content and experience (well, not yet anyway …). The fact remains that there are some things that you will still need to do in ASP.NET. A few high-level examples are listed below that you can extrapolate to make your own decisions:

- **Security Consciousness** — It's worth keeping in mind that Silverlight sits in the browser on the client machine. Therefore, you most certainly wouldn't want to consider automatically moving highly sensitive logic and/or data over to it without good reason. Sensitive operations should

be kept and maintained on the server unless a formal threat modeling exercise has shown that this isn't necessary.

- **Architectural Awareness** — In keeping with n-tier architecture, you should still leave your database access code (and similar code) in ASP.NET and provide access points for Silverlight. This also promotes abstraction of the databases, which is a good architectural decision.
- **Environmental Concerns** — The Silverlight plug-in is not going to be allowed in *all* environments, be they corporate, educational, or private. In situations in which it's against someone's policy, you have no choice but to leave everything in ASP.NET. As well as this, as broad-reaching as Silverlight is, currently it is not supported in every browser on every OS, so you may still need to cater for these exceptions with ASP.NET throughout.
- **Ease of Development** — There are some things that are (at the moment) going to be quicker, easier, and more tried and tested to do in ASP.NET. One such example is form creation, including validation for classic data entry. ASP.NET has a proven track record of allowing you to quickly create data entry applications, thanks to the wealth of controls that can be quickly developed against. There would be no perceived improvement in moving the data entry portions of your application into Silverlight at the moment.

The Development Environment Overview

The development environment for Silverlight is very easy to set up. First things first: You're going to need an Integrated Development Environment (IDE) to work with, and that IDE is Visual Studio 2008. To provide the Silverlight project templates, developer runtime, IntelliSense, debugging support, and other development requirements, you will also need to install the Silverlight Tools for Visual Studio 2008. These two items will complete the setup of your development environment, so you're free to use Visual Studio to create and edit Silverlight applications.

Once you have installed all of the above, you can fire up Visual Studio. By selecting File and then New Project, you will have access to the Silverlight project templates as shown in Figure 1-1.

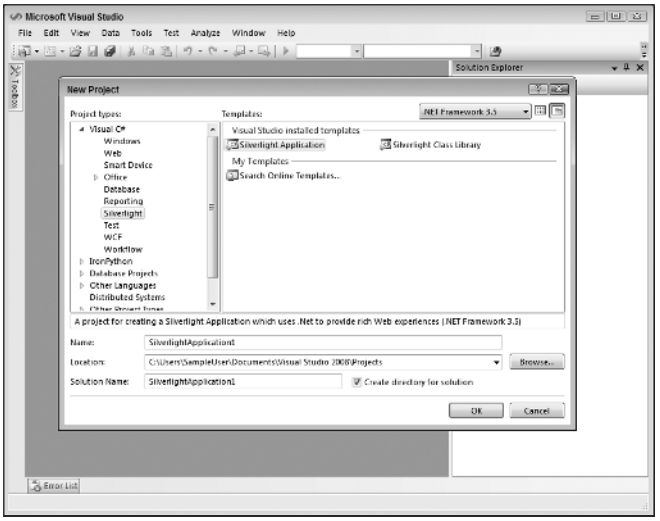

Figure 1-1

From here you can elect to create either a Silverlight application or a Silverlight Class Library (SCL), and then you're on your way. That's all there is to the development environment. In Chapter 3, "XAML Condensed," you take your first steps in actually using this development environment to create a Silverlight application and start learning XAML.

As well as the Visual Studio IDE for Silverlight development, if you fancy yourself to be a bit of a designer as well as a developer, you might want to consider downloading and installing Microsoft Expression Blend, a first-class environment for designers that can be used to work on both WPF and Silverlight applications. Chapters 3 and 5 show how Expression Blend can be used to quickly and easily output XAML that can then be used within your Visual Studio project.

Summary

In this first chapter, you learned at a high level what Silverlight is and how it can help you deliver much more engaging, immersive web applications without the overhead of increased development complexity.

You learned that prior to Silverlight, developing rich, immersive UIs in ASP.NET was challenging for various reasons, primarily arising from the very nature of developing in a disconnected environment with only HTML and JavaScript. This raised the trade-off of "rich versus reach," where you had to make a decision between a graphically rich UI or ease of deployment and uptake, but you couldn't have both.

Silverlight was intended to help solve the problem of "rich versus reach" by allowing you to create visually complex, engaging web applications that can run on a variety of operating systems and browsers. You saw how Silverlight had a simple installation from over the Web and that it provides streaming media support, AJAX-style updating, stunning graphics, and perhaps most importantly, a stripped-down version of the .NET framework to tie it all together.

Because Silverlight was designed to deliver next-generation media experiences and improved interactivity, it supports out-of-the-box the creation of a much more intense and consuming user experience, helping you to give your existing web site the edge over its competitors or to create a brand-new, cutting-edge site.

Importantly, you then learned that using Silverlight doesn't necessarily mean that all of your code, logic, and UI from now on should be moved across from ASP.NET. Four high-level considerations were discussed covering security, architecture, ease of development, and environment that showed what you probably wouldn't want to do in Silverlight and why.

Finally, you took a look at setting up the development environment to allow you to create Silverlight applications. You saw that two main components are required: Visual Studio 2008 and the Silverlight Tools for Visual Studio 2008.

In the next chapter, "Silverlight Architecture," you will take an in-depth look at the components that form the building blocks of Silverlight and the touch points that exist between ASP.NET and Silverlight.