 CHAPTER 1
Introducing Silverlight

It all started when Microsoft presented its revolutionary user interface (UI) framework, Windows Presentation Foundation, to an enthusiastic crowd of graphics designers, software developers, and businessmen in March 2006 at the new MIX conference in Las Vegas. Microsoft also added one session about a lesser-known technology with the rather barbarian name Windows Presentation Foundation Everywhere, or WPF/E. There was nothing much to see yet, but the abstract was enticing: “With WPF/E you’ll be able to build rich, interactive experiences that run in major Web browsers on major platforms as well as on mobile devices.”

A little more than a year later, at the second edition of the same MIX conference, Scott Guthrie (general manager at Microsoft, responsible for most of the .NET teams) climbed on stage and gave the crowd an amazing software demonstration. The barbarian WPF/E was gone; in its place was Silverlight (see Figure 1.1).

A bright new logo revolved on the screens. Gradients and animations were all over the place. Planes flew over the web browser’s window, connecting US cities while Scott was planning his next trips; a chess application let the browser’s JavaScript engine play against .NET, demonstrating without any doubt the superior power of the compiled .NET application over JavaScript’s interpreted code. Yes, the browser was running .NET!—not only in Internet Explorer but also on Firefox! And yes, even on a Macintosh computer.
It was an intense hour! Later that day, more demos of the new Expression Encoder were presented, which allows users to produce movies and run them in the web browser, after having copied them to a streaming (or to a nonstreaming) web server. Even better, with the Expression Encoder you can now change the look and feel of the video player completely, basing your work on predefined templates, but also changing everything from colors to shapes in Expression Blend.

These first demos were only the start of an amazing three days packed with Silverlight content. In a little less than a year, Silverlight had made it from little-known side attraction to major technology with a huge potential.

Where Does Silverlight Come From?

A little history can be interesting to understand where Silverlight came from, and how it is positioned among the many UI technologies available today. Depending on your past experience, you may be well aware of the facts presented in the next sections. In that case, feel free to skip ahead (the next interesting section for you will probably be “Running on Multiple Platforms”). If you don’t want to skip ahead, or if you feel like refreshing your memory, keep reading as this section and the next few sections fast forward from the antiquity of the World Wide Web to the magic of today.

Web pages have been around for a long time. In fact, HTML was presented to the world as early as 1992. (You know the old line about dog years being equal to seven human years? Well, 1992 in computer years is, like, Middle Age for humans!) Of course, HTML was limited in the beginning. Web pages were simple, composed mostly of text and hypertext (the most revolutionary idea of HTML, and the origin of the name Hypertext Markup Language).

Very quickly, however, the inventors of HTML saw the need for richer content to be made available in web pages. The first step in that direction was the invention of the IMG tag, allowing images to be placed inline. The basis for richer content was there, and the first popular web browser, Mosaic, displayed these pages.

Automating Web Pages with JavaScript

The next big step forward in the quest for rich content was the addition of a JavaScript engine running in the web browser.
This programming language (added in 1995 to the Netscape web browser, and then in 1996 in Microsoft Internet Explorer) enabled a much richer interaction with the user. Earlier, the only possible user action was through a postback to the web server, meaning that any data entered by the user had to be sent back to the server for handling, and the result was returned as a response. Obviously, this model was slow, especially at the time of dial-up connections.

With JavaScript, it was possible to provide a basic treatment of data on the client directly, without postback. A first obvious use for this client-side technology was a prevalidation of data, making sure that the user was not sending useless information to the server, wasting bandwidth and valuable server time. With JavaScript enabled, errors could be caught early, and faulty data would never leave the client. Of course, as soon as a web server interaction was needed, for example, to retrieve data, for complex calculations, to authenticate users, and so on, a postback was needed. This model still prevails in many websites even today.

**Cascading Style Sheets**

Another big step forward in the creation of rich web content has been the creation of Cascading Style Sheets (CSS). Finally a separation of content and layout was made possible. The look and feel of a web page could be extracted from the page itself and governed by the use of rules placed in external files. This model has many advantages:

- The responsibilities can easily be shared by various teams, depending on their skills. Graphics designers can be in charge of the look, developers of the functionality, while marketing is in charge of the content.

- In smaller projects, the developer can concentrate on the page’s structure and functionality first, and create the look later, after the content is fully known. This way of working is much more efficient than doing everything at the same time.

- The page’s look can be changed easily without touching the page itself.

- Same styles can be reused over and over again without having to rewrite the code. Classes can be located in a central location and referenced anywhere. In fact, it’s even possible to have styles on a central server and be used on other web servers, related or not.

As CSS became more and more sophisticated, the pages became more and more beautiful, to a point where the aspect of the pages, in some cases, becomes as important as the
content itself. One can argue endlessly whether that’s a good thing (after all, content is what users are really looking for, isn’t it?), but if you can choose between boring content and beautiful content, surely the beauty will make the experience more pleasurable.

Progressing to DHTML

As the JavaScript and CSS engines became more advanced, some interaction was made possible between them. **DHTML (Dynamic HTML)** was born. Actually, DHTML is not a technology; it’s a combination of JavaScript and CSS used in the context of HTML.

Using JavaScript as a programming language, it is possible to access elements of the page itself (content and style), and to modify them. Since this is client-side technology, no interaction with the server is required for this kind of effect.

Because CSS allows the setting of an element’s opacity, it is possible to create fade-in/ fade-out effects. With the possibility to set an element’s position in an absolute manner, JavaScript allows you to move an element across the screen. In addition to “just” adding some kind of intelligence and good looks to the web pages, the interaction between CSS and JavaScript enabled the creation of richer, smoother web pages.

Although this all sounds nice, DHTML is rather limited:

- Opacity is not supported in all browsers, and the ones that support it don’t use a standard syntax, making it kind of a maintenance nightmare.
- Moving elements across the screen can only be done at limited speed and is not as smooth as you would want.
- It is impossible to rotate elements. If you want to show a rotated button, you need to simulate this using multiple images, and the effect will not be smooth.
- Text (such as a title, in a logo, etc...) can only be rotated by using images. In that case, the localization becomes a nightmare, because you need to translate each image in each language you want to support.
- The “hit test area” for any element is always square. In other words, even if you have an image of a round button with a transparent background, you may activate the action even if you click outside the round area. The mouse cursor will turn to a hand even though the mouse is still on the transparent area.

Thankfully, Silverlight corrects all these shortcomings, and since it can be “blended” among HTML elements, it offers a real possibility to graphically enrich a web page.
Communicating in a Richer Way with AJAX

In addition to enabling DHTML, JavaScript brought a huge improvement to client-server communication, and to creating a smoother user experience: *Asynchronous JavaScript And XML (AJAX)*, a cumbersome name for a great technology. In short, AJAX allows JavaScript to send a request to the web server, and to get the response without the web page being refreshed. Instead of using the normal post-back mechanism, the whole communication occurs in the background, and the user doesn’t notice it.

With this technology, the user can get additional information from the server without losing the context he is working in. The user experience is also much smoother, because new information gets loaded without the “traumatic” reloading of the page and the temporary white screen caused by it.

Using Third-Party Plug-Ins

With the limitations of DHTML, various third-party additions to the web browser have been created with the explicit intent to enhance the user experience. The most notorious technologies are probably Java applets, ActiveX controls, and Flash applications. This section reviews the advantages and shortcomings of these technologies to understand better how Silverlight compares.

Using Java Applets

Java applets were extraordinarily popular for a short time and then became unpopular. While some web pages still use the Java technology, their number is decreasing. The main problem with Java is that the runtime is very slow to start. When you load a page with a Java applet on it, the initial loading time makes it a painful experience.

On the plus side, Java is a great programming language. It paved the way for .NET, and many of the best features of C# are greatly inspired by the Java language. It also enables advanced graphical effects, such as the famous ripple effect.

Limited Java-to-JavaScript and JavaScript-to-Java communication is possible. However, the interface used for this communication is not supported by all the web browsers that Java runs into. Also, the syntax is not easy to learn, and communication is awkward.
Java makes it possible to open alternative communication ways to the web server. For example, it enables the server to “talk” directly to the client, which is impossible with classic web communication. While this can be a huge technological advantage in certain scenarios, it can also, under certain circumstances, represent a security threat. Because of this, the acceptance of Java applets by big, security-sensitive corporations has been slow and has pretty much “killed” Java in the web browser.

Using ActiveX Controls
When Microsoft introduced the COM technology in 1993, it also made it possible to create so-called ActiveX controls, “packing” pieces of software and communicating with the external world using the COM interfaces. With this, it is possible to develop controls using classic Windows technology (including fast C++ code), and to place it in a web browser. It is even possible to have some limited interaction with JavaScript and through this to the web page.

The major shortcoming of ActiveX is that it uses an obsolete, almost 15-year-old technology. Also, ActiveX controls can only be used in Internet Explorer on Windows.

Using Flash Applications
Adobe Flash is probably the most popular third-party plug-in currently available. On the plus side, it allows advanced graphical effects, and it’s not rare to find websites programmed entirely in Flash. While Flash content is often called “Flash movies,” it’s probably fairer to call them “applications.”

Flash’s major shortcoming is that it is difficult to program. A (commercial) editor is needed to create the front-end user interface. Additionally, the code-behind can only be programmed in a language called ActionScript, which is a subset of JavaScript. The fact that Flash has been created primarily for graphics designers makes it difficult for software developers to use the tools and to create compelling content and functionality.

NOTE
Every book about web technology should make one thing clear: Java and JavaScript have strictly no relationship. When you program JavaScript, you do not program Java.

WARNING
Programming a website entirely in Flash (or in Silverlight) is not a good idea. Many platforms (especially mobile phones, PDAs, and so on) do not allow accessing such sites (or only in a limited manner). This type of technology should only be used to enhance the content of a website, not to replace it completely.

NOTE
Microsoft doesn’t advertise Silverlight as a Flash “killer.” This strategy would probably be doomed anyway, considering how many Flash designers there are, and how much Flash content is available online. Since both technologies can run safely side-by-side, there is really no need to start a religion war about which is the best!
A limited interaction between the Flash application and the containing HTML page is possible through JavaScript. Some interaction between the Flash application and the containing HTML page is possible through JavaScript.

Running on Multiple Platforms

One important factor when you develop a third-party web technology is to run it on as many platforms as possible. The Web is, by definition, ubiquitous, and it is not rare to find Internet-enabled devices of every shape and power. Adobe Flash is a good example of this: The plug-in is available for a great number of browsers on a great number of operating systems. This makes Flash a powerful platform, and Microsoft understands the need for Silverlight to run in multiple browsers too.

As said earlier, at the time of this writing, the Silverlight runtime is available for Internet Explorer and Firefox on Windows, and for Firefox and Safari on Macintosh. A version for Linux is in preparation. The real challenge is to offer consistent interfaces on all the platforms on which Silverlight runs. So far, this goal has been very successful, and is a great achievement in the quest for a unified web environment.

Some prototypes of Silverlight for smartphones have also been presented, but not much is known about what will be supported on these limited devices. A first version of Silverlight for mobile phone (running on Windows Mobile and also on Nokia mobile phones) should be released by the end of 2008. This should support Silverlight 1 applications, including video.

Because Silverlight runs on so many platforms, some limitations in the functionality (compared to the full .NET platform) are unavoidable. Many factors must be considered:

- The goal is to keep the runtime small, so that it can easily be installed over Internet connections. The goal in the foreseeable future is to keep it under (or equal to) SMB. When you compare to the size of the full .NET runtime (and even if you remove all the server-side bits), it’s obvious that not everything can be implemented.

- Features requiring a lot of hardware acceleration (especially 3D animations) will probably be too hard to implement consistently on multiple platforms (not even mentioning small limited mobile devices).

- Various platforms use various underlying graphic technologies, and some of these technologies

---

**NOTE**

In order to run Silverlight on many platforms in a consistent way, Microsoft is also collaborating with Novell and the makers of the open source .NET framework “Mono.”

---

**TIP**

Hardware acceleration is the process through which some of the complex calculations are executed by hardware processors specializing in graphics (as opposed to software). Hardware graphic processors are much faster than anything that software can calculate.
may not allow all the desired effects. It’s already pretty amazing to see how consistent the implementation is on both Windows and Macintosh. For Silverlight to be viable, it is imperative that it remain totally compatible on all supported platforms and browsers.

All these reasons create a lot of intense discussions to decide what will make it to Silverlight and what won’t. It will be interesting to observe the development of the platform!

**Tip**

Microsoft is listening to you! Don’t hesitate to contact evangelists and other Silverlight developers, either through the Silverlight forums (http://silverlight.net/forums), or through their blogs (check the section titled “Reading Silverlight-related blogs” in Chapter 24, “Silverlight: Continuing the Journey”). Let them know what you do with the technology, and what your expectations are. Just like third-party developers influenced the way WPF is growing, the same process is true for Silverlight! Let them hear your voice!

### Making the Web Application Secure

Security in web applications is a huge challenge. Even though Microsoft has more often than any other company been the target of virulent criticisms because some of their applications or operating systems were not secure enough, it’s fair to say that any popular web technology is exposed and has had security issues (Java, Firefox, the Linux operating system, the new Google Chrome web browser and many others have all been the target of attacks).

Microsoft is taking the security of Silverlight-based applications very seriously. Silverlight applications run in a “sandbox” that limits the functionality of the application and protects the computer it runs on against attacks. Every time a feature is allowed by the sandbox, it is carefully tested to make sure that the change doesn’t present a hole for a potential attack.

If humanity was intrinsically good, programming would be easier, but of course, security is not just a matter of attackers, it also has a lot to do with bad code, memory corruptions, and so on. Thankfully, the .NET programming platform (the so-called “managed code”) is a secure way of programming. Unlike older unmanaged C++ code, many bugs are eliminated already at the source by the much safer managed programming languages. Memory corruption is almost impossible; memory is automatically freed by the Garbage Collector when it is not used anymore, making memory leaks much more infrequent.

Obviously it would be naïve to believe that Silverlight will never be attacked, or that Silverlight applications will never crash. However, thanks to the experience gathered by the .NET teams, thanks to the managed programming languages, and thanks to the
extra attention invested by Microsoft into that matter, Silverlight should provide a very safe environment.

**Introducing Silverlight.net**

The community website http://silverlight.net contains a collection of valuable information about Silverlight, including quickstarts, samples, tutorials, and so on. Maybe the most interesting section is the gallery, available at http://silverlight.net/community/communitygallery.aspx.

Here you will find a lot of samples, made by Microsoft and third-party developers. This is a good starting point to get an idea of what Silverlight is capable of. Currently, the gallery is divided into two subsections: Silverlight 1.0 (with JavaScript) and Silverlight 2 (with .NET). However, if you install Silverlight 2 (as we recommend), you can still execute older applications.

**What Do You Need to Run Silverlight?**

Silverlight is an add-on to the web browser. It gets installed separately and adds functionality to your web pages. It is currently available for Internet Explorer and Firefox on the Windows XP and Vista operating systems; it is also available for Firefox and Safari on the Macintosh. At the time of this writing, a version for Linux is in preparation, following a historical agreement between Microsoft and Novell.

To run Silverlight applications, you need a compatible web browser. When you point a Silverlight-capable web browser to a web page with Silverlight content, you get a small Install Microsoft Silverlight icon, as shown in Figure 1.2, instead of the Silverlight content.

- Click on the icon to be taken to a Microsoft web page where you can download and install the Silverlight runtime on your PC.
- After installing Silverlight in Internet Explorer, you don’t even need to restart your web browser. In Firefox, you might have to, but it’s not that big a deal.

Alternatively, you can install Silverlight 2 from http://silverlight.net/GetStarted.

**WARNING**

If you worked with Silverlight before and you already installed a version older than the one required by the current application, you will also see the “Install Microsoft Silverlight” button. You can check the Silverlight version by right-clicking on any Silverlight application, choosing the Silverlight Configuration menu, and selecting the About tab.
Updating Your Runtime—Automatically

Once the runtime is installed, the Silverlight content will automatically be started. Also, a version check will happen when Silverlight is started; if a newer version is available, you can choose to install it automatically. This can be set up in the Silverlight configuration dialog by following these steps:

1. Navigate to http://silverlight.net/samples/1.0/Page-Turn/default.html. This starts one of the earlier Silverlight demos created by Microsoft: The PageTurn sample application.
2. Right-click on the Silverlight application and choose Silverlight Configuration.
3. Click on the Updates tab.
4. Choose the options you want (see Figure 1.3).

NOTE
Silverlight gets installed only once, and then runs in Firefox or IE (on the Mac, on Safari and Firefox) with the same runtime. If you change the options in Firefox, it will be valid in the other browser installed on the same PC too!

TIP
You can turn pages by clicking and dragging your mouse on the bottom-right corner of the photo album. If you click on the small open book icon on the bottom right, you see a list of thumbnails.

FIGURE 1.3 Silverlight configuration

Trying Silverlight Demos

The cool thing with Silverlight is that it’s so easy to deploy: Simply put the files on a web server, and anyone in the world with a Silverlight-capable browser can see the applications.
The applications developed by Microsoft for demo purposes are also available online. Additionally, third-party developers can also publish their own sample applications to the Silverlight.net website. As mentioned previously Microsoft used two demo applications when they first introduced Silverlight to the public: the Chess application and the Silverlight Airline sample.

Playing Chess Against Silverlight

At the time of writing, the Chess demo (see Figure 1.4) is not available yet for Silverlight 2. You can, however, see a video illustrating this application at http://on10.net/blog/tina/Microsoft-Silverlight-Chess. Also, keep checking the Silverlight.net gallery where the demo and its source code should be posted soon.

FIGURE 1.4 Chess application

In this application, you get a chance to play against the machine: The Human button is selected on the bottom. On the top, you can choose to play against another human (booooring), against JavaScript (remember, JScript is Microsoft’s name for its implementation of the technology) or against .NET.

The implementation for the chess engine is exactly the same in JavaScript and in .NET. So the most interesting thing to do in that application is to let JavaScript play against .NET:

1. On top, make sure that the .NET button is selected.
2. On the bottom, select the JScript button.
3. Observe how both engines are playing against each other. The most interesting data is the number of nodes per second calculated by each engine. This is an indication of how fast each engine can process data. Typically, .NET calculates approximately 1,000 times faster than JavaScript!
Running this application always leads to the same result: .NET wins over JavaScript, because it is just so much faster. This is a good proof (if it was needed) of some of the advantages we will encounter when we use .NET for advanced calculations instead of the JavaScript engine.

Planning Your Trips with the Airline Application

The airline application shown in Figure 1.5 was also shown on stage at MIX07 and presents what could be an airline reservation system in the (near) future. Here too, the demo is not yet available for Silverlight 2 at the time of writing but should be updated soon in the Silverlight.net gallery. A video showing this application is available at http://on10.net/blogs/tina/Microsoft-Silverlight-Airlines.

![Airline application](image)

FIGURE 1.5 Airline application

1. Click and hold the departure city. Then, still holding down the button, move the cursor to the target city. You can release the mouse now.

2. Select a date for your trip in the Calendar on the right.

3. The system calculates a number of possible itineraries for your trip. They are shown under the map. Pass your mouse cursor over one itinerary to see it drawn on the screen, and to see a small plane fly from city to city.

This example is interesting, because strictly speaking, Silverlight doesn’t offer new functionality here. Online reservation systems are already available today. However, the user interface is better with Silverlight and makes the user experience better and more attractive.

**NOTE**

Thanks to the goodness of vector-based graphics, you can also resize the window and note how the whole user interface gets resized to fit the screen.
DeepZooming into the Hard Rock Café

At MIX08, probably the most exciting demo was given by the Hard Rock Café and its memorabilia collection of rock souvenirs: How would you like to be able to see high resolution pictures of all the souvenirs normally exposed in various Hard Rock Cafés around the globe (see Figure 1.6)? Even better, how would you like to be able to select them by artist, year, type, and so on? And the most awesome feature of all: Zoom smoothly into the pictures until you are able to see every detail, like small scratches or a handwritten dedication on a guitar! Navigate to http://memorabilia.hardrock.com to try it yourself.

![Figure 1.6 Hard Rock Memorabilia](image)

This application is enabled by a feature introduced in Silverlight 2, named DeepZoom. A very high resolution picture is prepared by a tool and divided in multiple, lower-resolution pictures. A Silverlight control is loading these images dynamically (according to the level of zoom chosen) and displays them. Additionally, you can pan the image simply by clicking and dragging the mouse, with a lifelike movement. All these seemingly very complex features are made rather easy thanks to Silverlight 2 (though one should not underestimate the value of the developers and designers having created this application!).

Finding More Demos

Many more samples and demos are available online. Make sure to browse through the Silverlight.net gallery and play with demos to see what Silverlight can do at http://silverlight.net/community/communitygallery.aspx.
What Do You Need to Develop Silverlight?

The term “Silverlight” also qualifies the programming platform, which is developed in parallel to .NET 3.5. Since Silverlight is in development now, not all the .NET classes are available yet. In fact, some of them will never make it to Silverlight, as mentioned earlier. However, because the .NET basis is stable now, we can expect to see big improvements in Silverlight in the coming months as Microsoft releases new versions.

We are not going to create standalone applications here. We will only create applications embedded in a web page and unable to run on their own. The nice thing with this way of programming is that distributing (deploying) your application is as easy as copying some files to your web server. If you are adding Silverlight content to an existing web page, you probably already have a web server. If you don’t, you will need to get web server space eventually to publish your application.

During development, however, you will be able to use the web server IIS built in the Windows operating system. Even better, if you use Expression Blend or Visual Studio 2008, starting a web application is as easy as pressing F5! You learn more about deploying web applications in Chapter 7, “Deploying to a Web Page.”

Developing Silverlight can be done with just a text editor. Notepad (or its Mac equivalent) is an option, seriously. You can write your XAML markup (a new programming language we will introduce later) in there, and then run it in the web browser to see the result. However, developing this way can be painful. Fortunately, we now have the support of great tools to make that experience a real pleasure. The best place to start and install the development environment for Silverlight is from http://silverlight.net/GetStarted.

Expression Studio

When Microsoft announced WPF, it also gave us a new suite of tools named the Expression Studio. Included are three important tools for the Silverlight developer: Expression Design, Expression Blend, and Expression Encoder.

This book uses these tools to create Silverlight content. Expression Blend is studied more in detail in Chapters 4, “Expressing Yourself with Expression Blend,” and 6, “Blending a Little More” (and throughout the book). This great tool is located at the border between designers and developers and is used by both professions. It is an almost unavoidable tool when you create WPF and/or Silverlight content. Unfortunately, at the time of writing there is no free edition of Expression Blend. A fully functional demo version can be downloaded from the Microsoft Expression website at www.microsoft.com/expression.
In addition to Expression Blend, we will be using Expression Encoder in this book. This application can also be downloaded from the Microsoft Expression website.

Expression Design can be used to create XAML-based “assets” (pieces of design). Later, you can import these elements in your Silverlight application. We will talk about this in Chapter 24.

**Visual Studio 2008**

Additionally, we will enjoy a better programming tool than Notepad: The powerful Integrated Development Environment (IDE) Visual Studio 2008 is the tool of choice to program Silverlight applications. In this book, we will use mostly Visual Studio 2008 and Expression Blend to program Silverlight applications. A fully functional 90-day demo version can be downloaded at http://msdn.microsoft.com/en-us/vs2008/products/cc268305.aspx.

In addition to Visual Studio, you must also install the Silverlight tools available from the “Get Started” page at http://silverlight.net/GetStarted/.

You learn more about Visual Studio 2008 in Chapters 9, “Understanding .NET” and 10, “Progressing with .NET” about .NET, and in following chapters, where we will use these tools to create C# code-behind.

Microsoft released so-called Express editions of Visual Studio. Each Express edition presents a limited set of features targeted at a specific type of software development (for example, web applications, C#, VB.NET, and so on). These limited editions are free to download and install. According to Microsoft, you will be able to use these Express editions to program Silverlight. At the time of writing, however, an exact release date was not public.

**Reading the Documentation**

This book does not contain a full reference documentation about Silverlight but will give you a great head start in this technology. In addition, you will use the Silverlight documentation, available in various forms.
Browsing the Online Reference


Finding More Documentation

With the number of classes available in the .NET framework (and even in the Silverlight subset), and with the number of members that each class contains, a comprehensive documentation is a key factor when you program against this framework. Microsoft released a Silverlight 2 Software Development Kit (SDK). It contains technical documentation, libraries and tools to help you in your Silverlight experience.

1. Navigate to the Get Started page at http://silverlight.net/GetStarted
2. Scroll down until you see the link to Microsoft Silverlight 2 SDK. Follow this link and download the SDK setup file.
3. Execute the file “silverlight_sdk.exe” you just downloaded. This will extract the content to C:\Program Files\Microsoft SDKs\Silverlight\v2.0
4. For more information about the SDK, technical documentation, links to external resources, etc., check the “Welcome” page. You find it in your Start menu, under “All Programs / Microsoft Silverlight 2 SDK”.

The Silverlight documentation is also available as a compiled HTML file that can be installed locally (see Figure 1.7). The download link is on the SDK welcome page.

The Search tab allows you easy access to a specific topic. The documentation contains many interesting topics, and it is absolutely worth browsing through it.

Learning with Tutorials

Many tutorial websites are available for WPF, Silverlight, and Blend. The community website Silverlight.net has already been mentioned. Another exciting and innovative site is the Nibbles Tutorials website at www.nibblestutorials.net.
This website does an amazing job of presenting and teaching the technology. Using small, bite-sized tutorials, the Nibbles website allows you to test some of the knowledge acquired in this book.

Additionally, the website's author, Celso Gomes, is a talented graphics designer working for Microsoft, and his tutorials give you the designer's point of view on Silverlight (while this book gives you the developer's point of view).

Looking into Silverlight’s Future

Web technologies are evolving fast, and any attempt to look into the future and predict where they are heading in the long term will result in mortifying failure for the psychic...Silverlight is no exception. It's almost impossible to predict where the technology will be five or ten years from now.

One thing is sure, though: Microsoft is putting a lot of effort in Silverlight and porting the .NET framework to multiple platforms. Mobile Silverlight will “hit us” in the very close future. And because so many developers are already active on the .NET platform, we will see more and more Silverlight content soon.

It is also safe to write that Silverlight and WPF will become always more compatible with each other. Silverlight 2 is already delivering a great number of WPF-compatible classes. Learning either WPF or Silverlight will offer an easy passage to the other.

Is ASP.NET/HTML/JavaScript Dead?

With the advent of Rich Interactive Applications (RIAs) made with Silverlight or Flash, one can wonder if classic ASP.NET will still be available in the future. One thing is sure: ASP.NET is a very strong and important pillar of Microsoft’s client application strategy. In fact, Microsoft also released updates to the ASP.NET framework with its .NET framework 3.5 this year.

With “classic” ASP.NET (HTML/CSS/JavaScript) for “thin clients,” Silverlight for RIAs and WPF for desktop applications, you cover the whole range of client applications. And since all these platforms run .NET, you can use the same programming language of your choice (amongst C#, VB.NET and many others) to create your libraries.

So in short: No, ASP.NET is definitely not dead, and HTML-based web pages will continue to entertain and inform us on the Internet for many years to come.
Summary

In this chapter, we took a preliminary look at Silverlight and understood where it comes from and what technologies are its “ancestors.” We also started playing with demo applications and installed the development environment.

Now it is time to start getting our hands dirty, and to write code. The next chapters introduce you to the new programming language XAML and give you the occasion to write markup and to see your creations running. Gradually, we will build knowledge and help you create more elaborate code. Eventually, we will build applications together, bringing your knowledge of Silverlight to a productive level.