

Chapter 1

Understanding Microsoft's Web Technologies

In This Chapter

- ▶ Exploring Microsoft's tools for creating Web pages
 - ▶ Understanding the technologies behind dynamic content
 - ▶ Delving client-side and server-side programming
 - ▶ Pinpointing the roles of LINQ, DHTML, XML, XAML, and AJAX
 - ▶ Deciphering postbacks and page refreshes
-

In the beginning, the World Wide Web (WWW) was flat. It was an electronic library where academics and scientists posted dissertations and dusty data for reading with clunky, text-only browsers. With the advent of graphical browsers, the consumer-oriented Web took off. Content became vastly more colorful. Remember where you were the first time you experienced the exciting `<blink>` and `<marquee>` tags? (I bet you wish you could forget those gems!) Anyway, the Web has evolved as a rich, interactive, and personalized medium.

In the new version of Web (Web 2.0), functional pages aren't enough. User experience (abbreviated as *UX* in geekspeak) is hot, and sites are cool. This chapter looks at Microsoft's tools and technologies for creating and delivering engaging Web content.

Introducing the Content-Creation Tools

Microsoft has a range of tools for authoring Web pages that appeal to several skill levels. Some tools are more suited to Web page design, while others are more appropriate to programming.

Microsoft Office (Including Word 2007)

When Bill Gates realized that Microsoft was lagging on the Internet front, the word went out to integrate Web support into every product. As a result, you can save Excel spreadsheets, Word documents, and PowerPoint slides as Web pages.

Many companies use the Office suite to place information on their intranet because most employees are comfortable in Word and Excel. These tools are quite adequate for creating static Web content that some call *brochure ware*. Although somewhat bloated, the pages are faithful reproductions of the original document — especially when viewed in Microsoft's latest Internet Explorer browser.

There's nothing to stop you from using a "saved-as HTML" page in an ASP.NET site. However, you may find that removing the unwanted HTML markup takes more time than building the page from scratch.

Expression Web

Expression Web took over from Microsoft FrontPage as the content editor for professional designers. Although some see Expression as an advanced word processor for HTML pages, it's actually much more, thanks to many important tools for Web designers. These tools include file management, link checking, style editing, and drag-and-drop support for HTML and ASP.NET controls.

Expression Web inherited the excellent split-view editor from FrontPage that lets you work in graphical and source code modes at the same time. The feature is so well done that Microsoft yanked the HTML editor from Visual Web Developer and substituted the superior Expression/FrontPage version.

Expression Blend

Expression Blend is mainly for the ponytail set (artistic types who prefer Macs) to create vector-based, animated, and three-dimensional graphics — much the way they do in Photoshop. Blend has a rich set of brushes, palettes, paint buckets, text, gradients, timelines, and event triggers for those with the skill to take advantage of them.

The XML-based files that Blend generates work in Windows Presentation Foundation (WPF) applications that run on Windows and in cross-platform Silverlight apps for the Web. (For more on Silverlight, see the section later in this chapter.



Blend's user interface (UI) is dim and funereal — a far cry from the cheerful Windows XP or glitzy Windows Vista UI. The theory is that a drab, flat design environment doesn't distract an *artiste* from his or her canvas.

Visual Web Developer (Including Express)

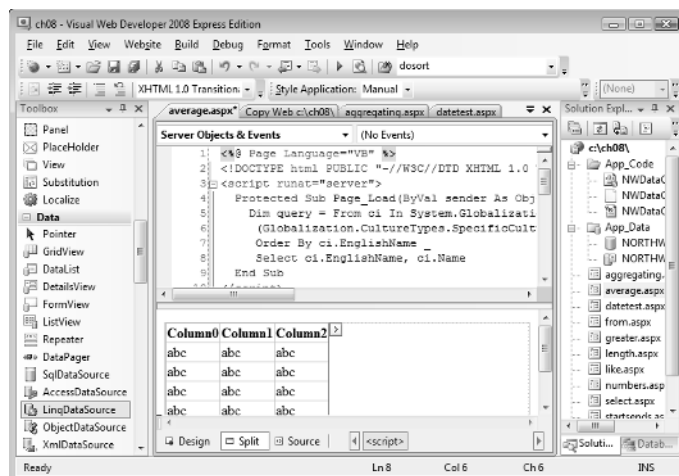
Visual Web Developer (VWD) is the premier tool for programming Web sites on the Microsoft platform. Just as Word is part of the Office suite, VWD is part of the bigger Visual Studio 2008 suite. Visual Studio includes Visual Basic .NET, Visual C#, and many other tools. Visual Studio comes in several versions to target teams of developers, database designers, testers, and system architects.

As an integrated development environment (IDE), Visual Web Developer helps you assemble and build the key elements of a Web application, including Web pages, images, controls, databases, style sheets, and, of course, the programming logic.

Visual Web Developer Express (VWDE), shown in Figure 1-1, is a somewhat stripped-down, freebie version intended for beginners and hobbyists. VWDE doesn't support add-ons, source control, extensibility, or macros — features that professional developers expect in a tool.

Most of this book's instructions are common to VWDE and VWD. You can do almost everything in this book with the free Express product. I note the few places in the book (mostly when debugging) that apply only to the upscale (\$\$\$) version of product. Chapter 3 gives you the cook's tour of VWD.

Figure 1-1:
Visual Web
Developer
Express
2008.



Meeting the Technologies behind Web Applications

The technologies that support Web applications come from different organizations and from different teams within Microsoft. Here's an overview of the parts that plug into — or on top of — each other.

Microsoft's .NET 3.5 Framework

The .NET Framework is the base of what geeks call the *stack*.

You can think of the stack as a multilayered wedding cake where layers depend on the layer below for support. The .NET Framework (technically, a compiled portion called the Common Language Runtime, or CLR) sits at the bottom, and its code talks to the underlying operating system, such as Windows Server 2008 and Windows Vista. ASP.NET 3.5 depends on the .NET 3.5 Framework. (See the next section for more on this framework.)

You hear geeks refer to *classes* or *class libraries* that make up the .NET Framework. They use dot-filled names like `System.Web`, `System.Data`, and `System.Xml.Linq`. This dotty stuff is just a way to organize and categorize thousands of chunks of prewritten code that programmers can tap into via programming languages, such as C#, C++, and Visual Basic.

Microsoft provides tons of reference documentation on everything that's in the .NET Framework. If you still don't find what you need, you can peek into its source code to see how Microsoft makes it all work.

ASP.NET 3.5

ASP.NET 3.5 is a technology to deliver interactive, data-driven Web applications over the Internet and intranets. ASP.NET includes a large number of prebuilt controls, such as text boxes, buttons, images, and data grids, that you can assemble, configure, and manipulate with code to create HTML pages that correctly appear in all popular browsers.

When combined with programming logic, ASP.NET lets you send HTML code that's specific to each user's circumstances or requests. For example, if a user wants a Web page to show HTML tables with green text and a purple background, your code can read the incoming request, verify that it's doable, and respond. This ability to create personalized, custom pages is known in the business as creating content *on the fly* and is a hallmark of server-side Web applications. Given that most people don't want green text on a purple background, the "special-orders-don't-upset-us" flexibility becomes a real bonus.

ASP.NET could have been XSP.NET

Instead of ASP.NET, the technology nearly became XSP.NET. In an interview with the Microsoft Architect Journal, Scott Guthrie, who helped establish Microsoft's core Web technologies, recalls the naming issue.

"We originally called it XSP; and people would always ask what the X stood for. At the time it really didn't stand for anything. XML started with that; XSLT started with that. Everything cool seemed to start with an X, so that's what we originally named it."

At another point, the technology was ASP+. That's before Microsoft's marketing department added a .NET suffix to almost everything that came out of Redmond.

Before the development of ASP.NET many of us learned to build sites with Active Server Pages, Microsoft's first Web scripting platform. ASP (now called ASP Classic) got its name during Microsoft's "Active" phase as in ActiveX, Active Desktop, and Active Directory.



Unlike static HTML pages that are stored on disk in a fully complete state, ASP.NET pages usually exist in a skeleton-like state on disk. It's only when a user requests a page that ASP.NET analyzes the markup, fills in all the content (often from a database), and sends HTML that the browser can render.

That's a very quick summary of what ASP.NET does. Don't fret if you don't grasp it all yet. You can fill in the blanks as you jump around the rest of the book.

ASP.NET Futures

The ASP.NET Futures releases consist of controls and technologies that the ASP.NET team is tinkering with or would like to demonstrate. It's a way of getting feedback, testing scenarios, and pushing the envelope without making a commitment to release the product.

The Futures items have no official support, even though some work quite well. Some components, such as the dynamic data controls, get their start in ASP.NET's Futures farm team and end up as professionals in an ASP.NET release or extensions update.

ASP.NET 3.5 Extensions

The ASP.NET team continues adding controls between official releases. These are packaged as extensions that you can download and install. As of this writing, the ASP.NET 3.5 Extensions include the `Silverlight` and `MediaPlayer` controls for presenting rich media on ASP.NET pages. Other

recent extensions and templates include Dynamic Data controls for displaying database content and an advanced architectural framework called Model View Controller (MVC).



Microsoft has many terms for unfinished software such as alpha, beta, preview, community technical preview (CTP), and release candidate. For critical production use, check whether an ASP.NET extension has made it to the Released to Web (RTW) or Released to Manufacturing (RTM) stage.

Web services

Web services let you deliver data and calculations to remote computers without restricting your client base to those running Windows. The most popular exchange format is the Simple Object Access Protocol (SOAP), which lets different platforms talk to each other by using XML.

Microsoft put a big push into Web services via ASP.NET in previous .NET releases. The follow-on emphasis has been on services using Windows Communication Foundation (WCF). WCF services are more robust and easier to secure, especially for enterprise applications where you may be sharing healthcare data with a company that handles the billing.

Smaller Web sites also have some interesting uses for services, especially when hooked in with technologies such as ASP.NET AJAX. See Chapters 9 and 15 for examples of Web services.

JavaScript and client-side code

Modern browsers understand an internal programming language called JavaScript. When the browser encounters JavaScript code (*script* in geekspeak) inside an HTML page, it runs the program's instructions. The browser (the client) doesn't need a connection to the server to run JavaScript code — it's completely independent. Client-side script uses the processing power of the computer on which the browser is running. That's a tremendous advantage because it takes the pressure off the Web server and distributes tasks to individuals.



Client-side scripting becomes complicated — and extremely powerful — when combined with logic on the server. Imagine this scenario: The Web server sends a stream of HTML that contains JavaScript instructions. Those instructions include JavaScript code that checks whether the anonymous user has typed a number from 1 to 10 in a text box. The browser sees the script and executes it locally. Until the user has typed a number from 1 to 10, the Web server isn't involved. When the browser sends the number back to the Web server, the return action is known as a *postback*. (See the sidebar “Postbacks and the rural mail carrier.”)

Postbacks and the rural mail carrier

What better way to explain the concept of a Web page postback than by bringing in a mail carrier from Rural Route #2, Powassan? Say that I'm sending a snail-mail letter to my publisher. I address the envelope, affix a stamp, and carry the letter to Alsace Road and Ruth Haven Drive where the rural mailboxes are lined up. In this scenario, consider me the Web browser (that is, a client).

Along comes Sheila (the mail carrier) on her daily run. I hand Sheila the letter, which she takes to the postal station in Powassan. For this discussion, consider the postal station (and the postal workers in the building) as the Web server. In browser terms, I've just done a postback by sending in the letter for processing.

But wait a minute! A worker in the post office checks the stamp and sees that the postage is insufficient to send a letter to the United States. She sticks a label over the letter describing the problem and puts the letter back in the RR #2 bin to return to the sender. The next day, Sheila brings back my

letter. I read the error message on the label, grumble, add more postage, and put the letter in the mailbox again. Sheila eventually takes the letter to the post office (the Web server) to resume its delayed journey.

My postback wasted time and resources because of the incorrect postage. Here's a preferable scenario that avoids a useless postback:

When I hand Sheila the letter, she glances at the address and checks the stamp.

"Sorry, Ken," she says. "You need 93 cents to send this!" and she hands the letter right back. (Remember, I'm the Web browser trying to submit something to the post office/server). I add the postage on the spot, and Sheila confirms the amount, accepting it without delay. This time, the postage was validated "on the client" without an unnecessary round trip.

When you hear about client-side validation, think of Sheila on RR #2, Powassan!

The powerful part is that the logic on the server can determine that 20 is an acceptable maximum number for a different customer and send a 20 in the JavaScript rather than the value 10. This way, the server is creating customized, client-side JavaScript on the fly.

ASP.NET AJAX

Asynchronous JavaScript and XML (AJAX) is a technology that reduces unnecessary and wasteful full page refreshes by limited the transfer of data to and from the Web server. (See the sidebar "Demolishing the house to change a window.")

On an AJAX-enabled page, you can type your credit card number in a text box, click the Submit button, and get a response such as "Credit Card Accepted" without disrupting the images, menus, and text elsewhere on the page. The browser sends only the required data to the server. When the message comes back, AJAX uses JavaScript code and Dynamic HTML to write into the designated part of the page.

Demolishing the house to change a window

To understand the benefits of AJAX, consider a renovation scenario. You've decided you want a stained glass window beside the front door. The renovator removes the existing plain glass and window frame, takes it to the shop for replacement with the stained glass, and returns to reinstall it. He obviously has no need to touch the other windows or — to be completely

ridiculous — tear down the house and replace everything in the process.

The same concept applies to a Web page. If you just want to change the content in one area of the page, you don't need to wipe out the existing page and ask the server to resend all the images and HTML markup. AJAX works like the renovator, doing just what's required but not more.

Microsoft's flavor of AJAX is an integral part of ASP.NET 3.5 rather than an add-on as in previous releases. As a result, if a bug or security flaw exists, Microsoft can fix its AJAX code via Automatic Updates or during the monthly celebration known as "Patch Tuesday."

You see AJAX in action throughout this book, but specifically in Chapters 4 and 15.

Dynamic HTML

While not exclusively a Microsoft technology, Dynamic HTML (DHTML) plays an important role in making Web pages responsive, interactive, and more like a regular Windows program.

When the browser analyzes the HTML code for a page, it creates an in-memory document. This document has a hierarchical structure where child elements nest inside their parent containers. For example, table rows are nested inside tables that are nested within the document's body.

The word *dynamic* in DHTML refers to the ability to change the characteristics of an element by using JavaScript. You've seen this ability many times without necessarily paying attention. For example, you're seeing DHTML at work when you hover the mouse over an image, and the image changes. Likewise, DHTML is at work when you click a plus sign to expand a paragraph of text. Chances are, JavaScript is instructing the text (or its container) to become visible — even though the original code sent from the server set the text as hidden.

The ability of JavaScript and ASP.NET AJAX to manipulate and rewrite almost any part of a Web page (the text included) is what makes most dynamic effects possible.

Extensible Markup Language (XML)

Although Microsoft had a hand in the specifications for Extensible Markup Language (XML), the standards come from the World Wide Web Consortium (W3C). Microsoft uses XML extensively in its Web technologies as a way of passing data around. These data exchanges include browser-to-server, server-to-browser, server-to-server, and from one program to another. You see XML in Chapter 7 as part of LINQ to XML and again in Chapter 9 within Web services. XML is also a big part of AJAX.

XML data has three big advantages:

- ✓ It's generated as plain text so that it passes easily through firewalls.
- ✓ Humans can read it and make at least some sense of it.
- ✓ You can create, parse, and manipulate XML on any platform, not just on Microsoft's operating systems.

Silverlight

Silverlight is Microsoft's cross-browser, cross-platform multimedia plug-in. It works on Windows, Macs, and even the rival Linux platform.

You've almost certainly seen Macromedia (now Adobe) Flash movies on a Web page. Silverlight is like Flash, only faster, more technologically advanced, and easier to program, especially in .NET languages. This so-called Flash killer uses a form of XML markup called XAML (sounds like zamel and rhymes with camel) to generate its graphics and behaviors.

You can use Silverlight, shown in Figure 1-2, to embed everything from screencams to animated cartoons to full-motion video using live, streaming broadcasts. The download size is reasonable, and Silverlight runs in its own isolated area, known as a *sandbox*, so the program should be secure enough for most uses.

Figure 1-2:
Silverlight
video may
become
more
common
than Flash.



Silverlight is very appealing as a multimedia platform. It promises to be a very big deal as the tools and technologies become more advanced. Expect to see entire database-driven applications running on Silverlight that maintain their appearance even when you resize the browser. You can dip into Silverlight and other rich media types in Chapter 16.

Language Integrated Queries (LINQ)

Language Integrated Query (LINQ) is a set of additions to the C# and VB.NET programming languages that make it easier to deal with data. LINQ comes in several dialects, including LINQ to SQL, LINQ to XML, and LINQ to objects. After you master LINQ's statements and syntax, you can apply the knowledge to all sorts of data. In fact, LINQ lets you combine data from multiple sources, such as a database, Web service, and XML file.

For most people, the big payoff is LINQ's support for SQL Server. Instead of writing complicated SQL statements — and crossing your fingers that no syntax errors occur — LINQ lets you use familiar keywords in queries. Visual Web Developer (as with other members of the Visual Studio 2008 family) watches what you type and alerts you to problems.

Chapter 7 shows how to use LINQ to select, sort, and group data of all kinds. Chapter 8 focuses on the `LinqDataSource` control and `DataContext` object in ASP.NET applications and shows how to massage SQL Server data by using LINQ to SQL.

ADO.NET

ADO.NET is Microsoft's technology for working with data and databases of all types. When a Web application talks to a database such as Microsoft SQL Server, it's probably using ADO.NET. The introduction of LINQ has hidden much of ADO.NET from view in Visual Web Developer.

SQL Server

SQL Server 2005 and 2008 are key products in Microsoft's Web technology strategy. The phrase "It's all about the data" applies to most serious Web applications. Whether you're tracking user preferences, generating complex reports, or storing customer orders, you need a fast and reliable data engine and relational database.

Microsoft provides SQL Server Express for free (but, as they say, "connect charges may apply"), making it a great choice for beginners. The skills and data you acquire by using SQL Express are directly transferable to the latest versions of SQL Server from standard to enterprise. You use SQL Server (mostly the Express version) throughout the book.

Internet Information Services

Internet Information Services (IIS) is Microsoft's premier Web server product that comes free with the latest versions of Windows.

As a platform, IIS delivers Web pages and Web services as requested by a browser or other application. ASP.NET 3.5 meshes seamlessly with IIS to produce the dynamic pages you're reading about in this chapter.

You can run IIS on your developer workstation, over your company's intranet, or expose it to the vast public on the Internet. However, unless you're running a large business on the Internet, you probably use IIS through an independent hosting company. These *hosters* are specialists who rent space on their servers, sell bandwidth, maintain connections to the Internet, and schedule backups.

During the development stage in Visual Web Developer, you may not use IIS at all. VWD includes a light Web server that does almost everything you need on your local development machine. When you're satisfied with the pages and code, you transfer the site to an IIS machine from within VWD. (For details on deployment, see Chapter 20.)

