1

Getting Started with ASP.NET 4

WHAT YOU WILL LEARN IN THIS CHAPTER:

- ➤ How to acquire and install Visual Web Developer 2010 Express and Visual Studio 2010
- ➤ How to create your first web site with Visual Web Developer
- ➤ How an ASP.NET page is processed by the server and sent to the browser
- ➤ How you can use and customize the development environment

Ever since the first release of the .NET Framework 1.0 in early 2002, Microsoft has put a lot of effort and development time into ASP.NET, the part of the .NET Framework that enables you to build rich web applications. This first release meant a radical change from the older Microsoft technology to build web sites called *Active Server Pages* (ASP), now often referred to as *classic* ASP. The introduction of ASP.NET 1.0 and the associated Visual Studio .NET 2002 gave developers the following benefits over classic ASP:

- ➤ A clean separation between presentation and code. With classic ASP, your programming logic was often scattered throughout the *HTML* of the page, making it hard to make changes to the page later.
- ➤ A development model that was much closer to the way desktop applications are programmed. This made it easier for the many Visual Basic desktop programmers to make the switch to web applications.
- ➤ A feature-rich development tool (called Visual Studio .NET) that allowed developers to create and code their web applications visually.
- ➤ A choice between a number of *object-oriented programming* languages, of which Visual Basic .NET and C# (pronounced as C-Sharp) are now the most popular.

- Access to the entire .NET Framework, which for the first time meant that web developers had a unified and easy way to access many advanced features to work with databases, files, e-mail, networking tools, and much more.

Despite the many advantages of ASP.NET over the older model, using ASP.NET also meant an increase of complexity and the knowledge you needed to build applications with it, making it harder for many new programmers to get started with ASP.NET.

After the initial release in 2002, Microsoft released another version of the .NET Framework (called .NET 1.1) and the development IDE Visual Studio .NET in 2003. Many people saw this as a service pack for the initial release, although it also brought a lot of new enhancements in both the framework and the development tools.

In November 2005, Visual Studio 2005 and ASP.NET 2.0 were released. To the pleasant surprise of many developers around the world, Microsoft had again been able to drastically improve and expand the product, adding many features and tools that helped reduce the complexity that was introduced with ASP.NET 1.0. New wizards and smart controls made it possible to reduce the code required to build an application, decreasing the learning curve for new developers and increasing their productivity.

Although Visual Studio 2005 and ASP.NET 2.0 were already very feature rich, Microsoft managed again to add a whole bunch of cool new features in Visual Studio 2008 and ASP.NET 3.5, which were released in November 2007. Major new functionality included LINQ (discussed in Chapter 14) and the integration of the AJAX Framework (which you learn more about in Chapter 10). In August 2008 Microsoft released Service Pack 1 for Visual Studio and the .NET Framework, introducing major new features like the ADO.NET Entity Framework (discussed in Chapter 14) and Dynamic Data.

The current versions, Visual Studio 2010 (often pronounced as "twenty-ten") and ASP.NET 4, build on top of the successful Visual Studio 2008 and ASP.NET 3.5 releases, leaving many of the beloved features in place, while adding new features and tools in other areas.

Over the next 19 chapters, you learn how to build full-featured ASP.NET web sites using Visual Web Developer 2010, Microsoft's development tool for ASP.NET web applications, which is part of the full Visual Studio 2010 suite. This book guides you through the process of creating a fully functional, database-driven web site, starting with a bare-bones web site in this chapter, all the way down to the deployment of it to a production environment in Chapter 19.

The sample site that comes with this book and all the examples are built with Visual Web Developer 2010 (VWD), so it's important that you have it installed on your development machine. The next section shows you how to acquire and install VWD. Once you have it up and running, you see how to create your first web site, followed by an extensive tour through the many features of VWD.

MICROSOFT VISUAL WEB DEVELOPER

Although you could theoretically write ASP.NET web applications with Notepad or another text editor alone, you really want to install a copy of Microsoft Visual Web Developer 2010. VWD is developed specifically for building ASP.NET web sites, and as such, hosts an enormous amount of tools that will help you in rapidly creating complex ASP.NET web applications.

Visual Web Developer comes in two flavors: as a standalone and free version called Microsoft Visual Web Developer 2010 Express, and as part of the larger development suite called Visual Studio 2010, which is also available in different editions, each with its own price tag. Although the Express edition of VWD is free, it contains all the features and tools you need to create complex and feature-rich web applications. All the examples you find in the book can be built with the free Express edition so there's no need to shell out big bucks for the commercial versions of Visual Studio 2010 to follow along with this book.

Getting VWD is easy. You can download it from the Microsoft site as discussed next.

Getting Visual Web Developer

You can get the free version of VWD from the Microsoft site at `www.microsoft.com/express/`. On the Express home page, follow the Downloads link until you reach the page that offers the downloads for the Express products, including Visual Web Developer 2010 Express. From this page, you can download Visual Web Developer 2010 Express as a Web Install, where you download only the installer, while the remaining files are downloaded during the installation process. Make sure you choose Visual Web Developer 2010 from the page, and not one of the other free Express products or one of the older editions of Visual Web Developer. The page also enables you to download all Express products conveniently as an ISO image that you can burn onto a DVD.

Don't be fooled by the file size of the Web Install download, which is around 3.5MB. The file you download is just the installer that downloads the required files over the Internet. The total download depends on your current system and will be somewhere between 180 MB and 270 MB.

If you want to try out the full version of Visual Studio 2010, which also contains Visual Web Developer, you can sign up for a free trial that you can get from the Microsoft site at `http://msdn.microsoft.com/vstudio`. You can choose to download an ISO image that you'll need to burn on a DVD.

Finally, you can download VWD as part of the *Microsoft Web Platform Installer* (WPI) application available for download at `www.microsoft.com/web` and at `www.asp.net/vwd/`. Besides VWD, this tool also gives you easy access to many other web development related tools and programs. The WPI is an excellent tool to get a whole bunch of web development-related programs and tools in one fell swoop. I often use it to get up and running real quick on a clean development machine.

Installing Visual Web Developer Express

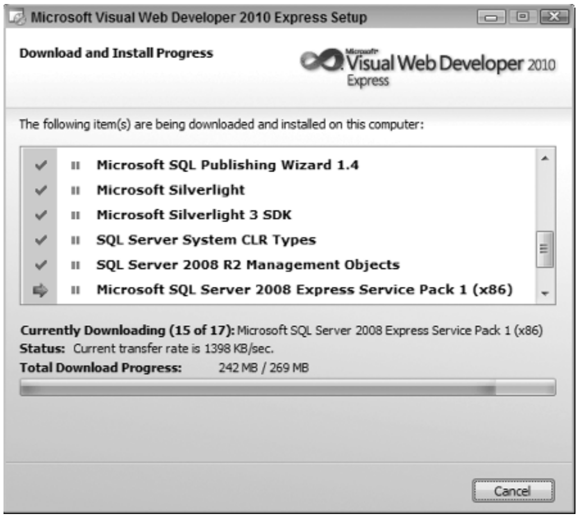
Installing Visual Web Developer is a straightforward, although somewhat lengthy, process. Depending on your installation method, your computer and your Internet connection speed, installing VWD may take anywhere between twenty minutes and an hour or even more.

TRY IT OUT Installing Visual Web Developer 2010 Express

This Try It Out exercise guides you through installing VWD Express on your computer. It assumes you're using the web download option as explained earlier, although the process for installing the Express edition from a DVD is almost identical. The steps you need to perform to install the full versions of Visual Studio 2010 are similar as well, although the screens you'll see will be somewhat different.

No matter which version of VWD you install, it's important that you also install SQL Server 2008 Express with Service Pack 1 — a required component if you want to follow along with many of this book's examples. When you install the full version of Visual Studio 2010, the option to install SQL Server is included on the list with features to install that you see during setup. If you install VWD Express, you get the option to choose SQL Server on the Installation Options dialog box. The Web Platform Installer has a similar option that enables you to install SQL Server 2008 Express with SP1 or later located under Web Platform ➪ Database.

1. When you're installing the web version, run the file you downloaded from the Microsoft web site. Otherwise, start the setup process from the Visual Web Developer DVD.
2. Once the installer has started, click Next, read and accept the license terms, and click Next once more.
3. On the Installation Options page, make sure you select Microsoft SQL Server 2008 Express. Although this option adds considerably to the size of the download, you really need it when building data-driven ASP.NET web applications. If you don't see the SQL Server option, you already have it installed. If you're unsure whether SQL Server 2008 is installed, refer to Appendix B. Click Next again. You may also see other optional components such as Microsoft Silverlight that you can install as well, although they are not required for this book.
4. On the Destination Folder page, you can leave the Install In Folder field set to its default if you have enough space on your primary disk. Otherwise, click the Browse button and select a different location.
5. Click the Install button. If you're using the web-based installer, the setup application will first download the files over the Internet to your computer. During the installation process, you'll see a screen (similar to Figure 1-1) that shows you the progress of the download and installation of VWD.


Microsoft Visual Web Developer 2010 Express Setup
Download and Install Progress
Microsoft Visual Web Developer 2010 Express
Cancel

The following item(s) are being downloaded and installed on this computer:

- ✓ Microsoft SQL Publishing Wizard 1.4
- ✓ Microsoft Silverlight
- ✓ Microsoft Silverlight 3 SDK
- ✓ SQL Server System CLR Types
- ✓ SQL Server 2008 R2 Management Objects
- ➪ Microsoft SQL Server 2008 Express Service Pack 1 (x86)

Currently Downloading (15 of 17): Microsoft SQL Server 2008 Express Service Pack 1 (x86)
Status: Current transfer rate is 1398 KB/sec.
Total Download Progress: 242 MB / 269 MB

FIGURE 1-1

6. The installer may need to reboot your machine during or after the installation. Once the installer has finished, VWD is ready for use.

How It Works

The straightforward installation process guided you through the setup of VWD 2010 Express. In the Installation Options dialog box, you selected Microsoft SQL Server 2008 Express, Microsoft's free version of its database engine. SQL Server 2008 is discussed and used a lot in this book, starting with Chapter 12. Appendix B shows you how to configure security settings for the various versions of SQL Server 2008 using the free SQL Server Management Studio Express.

Now that VWD is installed, it's time to fire it up and start working with it. The next section shows you how to create your very first site in VWD. You see how to create a site, add content to a web page, and view that page in your browser.

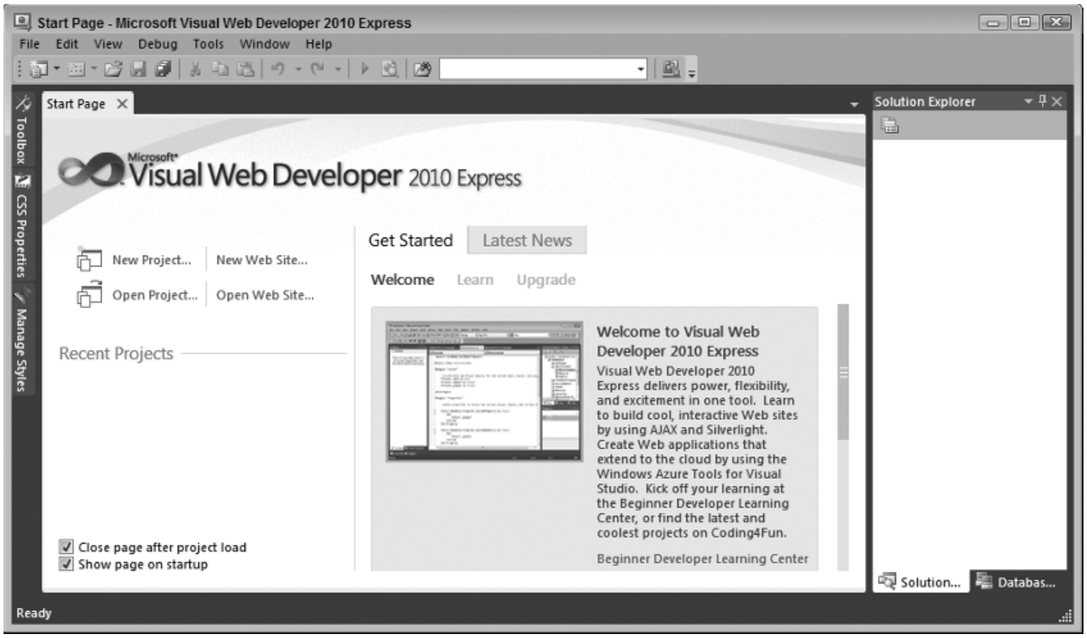
CREATING YOUR FIRST ASP.NET 4 WEB SITE

You probably can't wait to get started with your first ASP.NET web site, so instead of giving you a theoretical overview of web sites in VWD, the next Try It Out exercise dives right into the action and shows you how to build your first web project. Then, in the How It Works explanation and the section that follows, you get a good look at what goes on behind the scenes when you view an ASP.NET page in your browser.

TRY IT OUT Creating Your First ASP.NET Web Site

1. Start VWD 2010 from the Windows Start menu if you haven't done so already. The first time you start VWD, there might be a delay before you can use VWD because it's busy configuring itself. Subsequent starts of the application will go much faster.
2. If you're using a commercial version of Visual Studio, you also get a dialog box that lets you choose between different collections of settings the first time you start Visual Studio. The choice you make on that dialog box influences the layout of windows, toolboxes, menus, and shortcuts. Choose the Web Development settings because those settings are designed specifically for ASP.NET developers. You can always choose a different profile later by resetting your settings, as explained later in this chapter.
3. Once VWD is fully configured, you see the main screen appear, as shown in Figure 1-2.

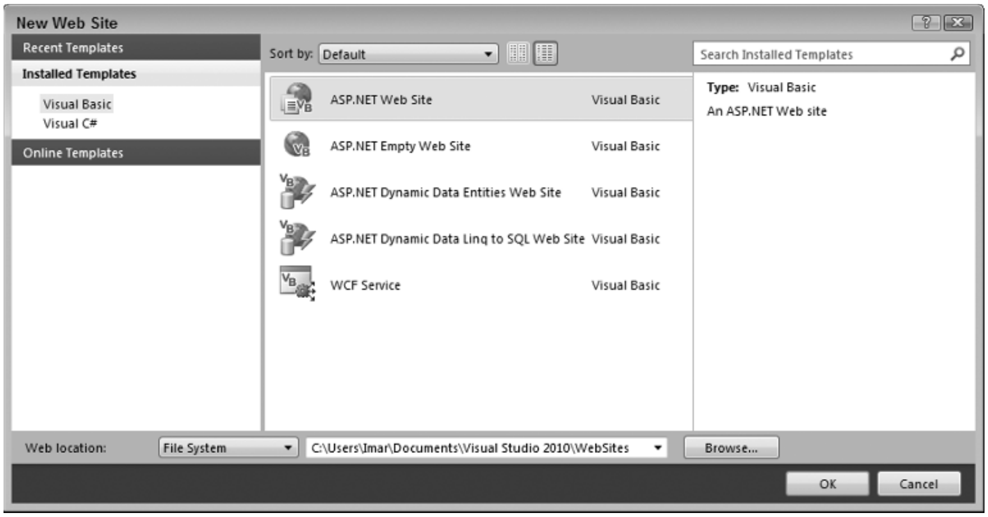
   You get a full description of all the windows, toolbars, panels, and menus in the next section, so for now, just focus on creating a new web site. Click the File menu in the upper-left corner and choose New Web Site. If you're using a commercial version of Visual Studio, depending on the settings you chose when starting Visual Studio the first time, you may have to open the submenu New first. (Make sure you don't accidentally use the New Project menu, because that is used to create different types of .NET applications.) The New Web Site dialog box appears as shown in Figure 1-3.



Welcome to Visual Web Developer 2010 Express

Visual Web Developer 2010 Express delivers power, flexibility, and excitement in one tool. Learn to build cool, interactive Web sites by using AJAX and Silverlight. Create Web applications that extend to the cloud by using the Windows Azure Tools for Visual Studio. Kick off your learning at the Beginner Developer Learning Center, or find the latest and coolest projects on Coding4Fun.

FIGURE 1-2



Sort by: Default

ASP.NET Web Site	Visual Basic
ASP.NET Empty Web Site	Visual Basic
ASP.NET Dynamic Data Entities Web Site	Visual Basic
ASP.NET Dynamic Data Linq to SQL Web Site	Visual Basic
WCF Service	Visual Basic

Type: Visual Basic

An ASP.NET Web site

Web location: File System C:\Users\Imar\Documents\Visual Studio 2010\WebSites

FIGURE 1-3

4. In the Installed Templates section on the left you can choose a programming language you will use for your site. This book shows all examples in both Visual Basic and Visual C# so you can choose a language to your liking.

5. In the list with templates in the middle, verify that ASP.NET Web Site is selected. Verify that File System is the selected option in the Web Location drop-down list at the bottom left. If you want, you could change the location on disk where the web site is stored by clicking the Browse button and choosing a new location on your computer's hard drive. For now, the default location — a folder under your Documents folder — is fine, so you can leave the location as is.

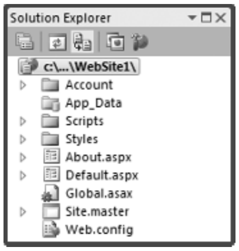


FIGURE 1-4

6. Click OK. VWD creates a new web site for you that includes a number of files and folders to jump start your web site as shown in Figure 1-4. It also opens the file `Default.aspx` so you can see the code for the page.

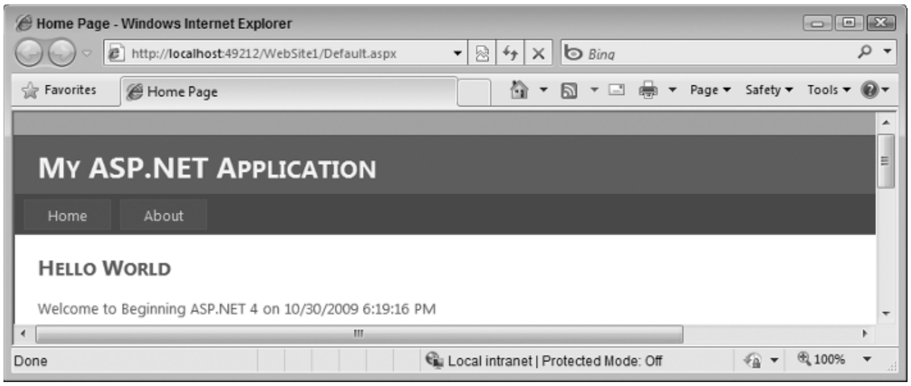
7. Remove the code inside the `<asp:Content>` block (it starts with `<h2>` and ends with `</p>`) and replace it with the following bolded text and code:

```
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainContent">
  <h2>Hello World</h2>
  <p>Welcome to Beginning ASP.NET 4 on <%= DateTime.Now.ToString() %></p>
</asp:Content>
```

You'll see code formatted like this a lot more in this book. When you are instructed to type in code formatted like this with some code in bold, you only need to type in the highlighted code. The other code should already be present in the file.

Don't worry about the code with the angle brackets (<>) and percentage symbol in the welcome message; you learn how it works later. Although this code may not look familiar to you now, you can probably guess what it does: it writes out today's date and time.

8. Press Ctrl+F5 to open the page in your default web browser. You see a page similar to the one shown in Figure 1-5.


Home Page - Windows Internet Explorer
http://localhost:49212/WebSite1/Default.aspx
Bing
Favorites
Home Page
Page
Safety
Tools
Done
Local intranet | Protected Mode: Off
100%

My ASP.NET Application

Home About

Hello World

Welcome to Beginning ASP.NET 4 on 10/30/2009 6:19:16 PM

FIGURE 1-5

If you get a dialog box asking for your user name and password, close your browser and go back to VWD. Right-click your site in the Solution Explorer (it's the first item in Figure 1-4) and choose Property Pages. In the Start Options section clear the checkbox for the NTLM Authentication item. Then click OK and press Ctrl+F5 again to view the page in the browser. If you see an information bar warning about Intranet settings in Internet Explorer, click the bar and choose Enable Intranet Settings. If you want to learn more about the implications of these settings first, choose What are Intranet Settings from the popup menu.

If you don't see the date and time in the page, or if you get an error, look again at the code in the welcome message. It starts with an angle bracket (<) followed by a percentage symbol and an equals sign. It closes with a single percentage sign and another angle bracket (>). Also, make sure you typed in the code exactly as shown here, including capitalization. This is especially true when you are using C#, because that language is case sensitive.

9. Notice how a small icon with a screen tip appeared in the tray bar of Windows, visible in Figure 1-6.

If you don't see the icon, right-click the arrow near the other icons in the Windows tray and choose Customize Notification Icons. Then set the WebDev.WebServer40.exe option to Show Icon and Notifications. The icon belongs to the ASP.NET Development Server. This web server has been started by VWD automatically to serve the request for your page. You learn more about how the web server processes your page later in this chapter.

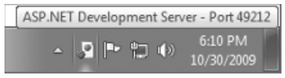


FIGURE 1-6

That's it. You just created your very first ASP.NET 4 web site with VWD.

How It Works

Although the web site you created in this Try It Out is quite simple, the process that eventually results in the page `Default.aspx` being displayed in your browser isn't so simple. All by itself, an ASP.NET page (also referred to as an ASPX page because of its extension) can't do much. It needs to be processed and served by a *web server* before your browser can display it. That's why VWD automatically started up the built-in ASP.NET Development Server to handle the request for the page. Next, it started up your default web browser and directed it to the address of the web server, `http://localhost:49212/WebSite1/Default.aspx` in the Try It Out example, although the actual number in the address may change every time you start the web server because the number is randomly chosen by VWD.

It's important to realize that the ASPX file you modified in VWD is not the same as the one that eventually gets displayed by the browser.

When you create a page in VWD, you add *markup* to it. The markup in an ASPX page is a combination of plain text, HTML, code for ASP.NET Server Controls (which you learn more about in this chapter and in Chapter 4), code written in Visual Basic.NET or C#, and more.

When you request an ASPX page in your browser, the web server processes the page, executes any code it finds in the file, and effectively transforms the ASP.NET markup into plain *HTML* that it then sends to the browser, where it is displayed. In the preceding Try It Out, the resulting HTML causes the browser to display the current date and time. HTML, or *HyperText Markup Language*, is the language that browsers use to display a web page. You learn how HTML looks and how to use it later in this chapter.

To see how the final HTML differs from the original ASPX page, open the source for the page in your browser. In most browsers, you can bring up the source window by right-clicking the page in the browser and choosing View Source or View Page Source. This brings up your default text editor, showing the HTML for the page.

If you already closed your browser after the preceding Try It Out, press Ctrl+F5 in VWD to open the page and choose View Source again.

Most of the HTML you see in the text editor is similar to the original ASPX page. However, if you look at the line that displays the welcome message and the current date and time, you'll notice a big difference. Instead of the code between the angle brackets and percentage signs, you now see the actual date and time:

```
<div class="main">
  <h2>Hello World</h2>
  <p>Welcome to Beginning ASP.NET 4 on 10/30/2009 6:19:16 PM</p>
</div>
```

When the web server processed the page, it looked up the current date and time from the server, and inserted it in the HTML that got sent to the browser. Depending on the language settings of your Windows installation, you may see the date and time formatted differently to accommodate the Windows Regional Settings.

In the following section, you see how ASP.NET works in much more detail.

AN INTRODUCTION TO ASP.NET 4

When you type a web address like `www.wrox.com` in your web browser and press Enter, the browser sends a request to the web server at that address. This is done through HTTP, the *HyperText Transfer Protocol*. HTTP is the protocol by which web browsers and web servers communicate. When you send the address, you send a *request* to the server. When the server is active and the request is valid, the server accepts the request, processes it, and then sends the *response* back to the client browser. The relationship between the request and response is shown in Figure 1-7.

Because you are using the built-in Development Web Server, the server and the client are really the same machine. However, in a real-world scenario, you'll host your web site on an external web server where it can be accessed by many different clients.

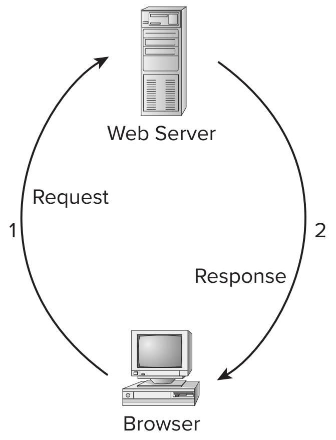


FIGURE 1-7

For simple, static files, like HTML files or images, the web server simply reads the file from its local hard drive and sends it to the browser. However, for dynamic files, such as ASPX pages, this is obviously not good enough. If the web server were to send the ASPX file directly to the browser as a text file, you wouldn't have seen the current date and time in the browser, but instead you would have seen the actual code (`<%= DateTime.Now.ToString() %>`). So, instead of sending the file directly, the web server hands over the request to another piece of software that is able to process the page. This is done with a concept called Application Mapping or Handler Mapping, where an extension of a file (`.aspx` in this example) is mapped to an application that is capable of handling it. In the case of an `.aspx` page, the request is eventually handled and processed by the *ASP.NET runtime*, part of the Microsoft .NET Framework designed specifically to handle web requests.

During the processing of the page, three main areas can influence the way the page eventually ends up in the browser:

- **Static text.** Any static text, like HTML, CSS, or JavaScript code you place in a page, is sent to the browser directly. You learn more about HTML, CSS, and JavaScript (a programming language used at the client) in this and subsequent chapters, including Chapter 3, which gives you a detailed look at CSS.
- **ASP.NET Server Controls.** These controls are placed in your ASPX page and when they are processed, they emit HTML that is inserted in the page. You learn more about Server Controls after the discussion of HTML in this chapter, and Chapter 4 is devoted entirely to ASP.NET Server Controls.
- **Programming code.** You can embed code, like Visual Basic .NET or C#, directly in a page, as you saw in the previous Try It Out. In addition, you can place code in a separate code file, called a *Code Behind* file. This code can be executed by the runtime automatically, or based on a user's action. Either way, execution of the code can greatly influence the way the page is displayed, by accessing databases, performing calculations, hiding or showing specific controls, and much more. You learn more about this Code Behind file in the next chapter, and programming your ASP.NET web pages is discussed in great detail in Chapter 5.

Once the page is done processing, and all the HTML for the page has been collected, the HTML is sent back to the browser. The browser then reads it, parses it and, finally, displays the page for you to look at.

Because HTML is so critical for displaying web pages, the next section gives you an overview of HTML.

Understanding HTML

HTML is the de facto language for creating web pages and is understood by every web browser that exists today. Since the beginning of the '90s it has been the driving force of the World Wide Web, the part of the Internet that deals with web pages. HTML documents are simple text files that contain markup, text, and additional data that influences that text.

HTML Elements and Tags

HTML uses text surrounded by angle brackets to indicate how your content should be *rendered* (or displayed) in the browser. The text with angle brackets is referred to as a *tag*; a pair of tags holding some text or other content is referred to as an *element*. Take another look at the HTML you saw in the previous Try It Out where you opened the source window for the page in the browser:

```
<h2>Hello World</h2>
<p>Welcome to Beginning ASP.NET 4 on 10/30/2009 6:19:16 PM</p>
```

The first line of this example contains an `<h2>` element with an opening tag (`<h2>`) and a closing tag (`</h2>`). This element is used to signify a heading at the second level (if you scroll up a bit in the final source in the browser, you also see an `<h1>` element). Notice how the element is closed with a similar tag, but with an additional forward slash (/) in it: `</h2>`. Any text between these opening and closing tags is considered part of the element, and is thus rendered as a heading. In most browsers, this means the text is rendered in a larger font. Similar to the `<h2>` tag are tags for creating headings up to level six, such as `<h1>`, `<h3>`, and so on.

Below the heading element, you see a `<p>` element, which is used to denote a paragraph. All text within the pair of `<p>` tags is considered part of the paragraph. By default, a browser renders a paragraph with some additional margin spacing at the bottom, although you can override that behavior.

Many tags are available in HTML; too many to cover them all here. The following table lists some of the most important tags and describes how they can be used. For a complete list of all HTML elements, take a look at the web site of the organization that maintains the HTML standard: `www.w3.org/TR/html401/index/elements.html`.

TAG	DESCRIPTION	EXAMPLE
`<html>`	Used to denote the start and end of the entire page.	`<html>` `...All other content goes here` `</html>`
`<head>`	Used to denote a special section of the page that contains data about the page, including its title and references to external resources.	`<head>` `... Content goes here` `</head>`
`<title>`	Used to define the title of the page. This title will appear in the browser's title bar.	`<title>` `Welcome to Planet Wrox 4` `</title>`
`<body>`	Used to denote the start and end of the body of the page.	`<body>` `Page body goes here` `</body>`
`<a>`	Used to link one web page to another.	`<a href="http://www.wrox.com">` `Visit the Wrox site` `</a>`
`<img>`	Used to embed images in a page.	`<img src="Logo.gif" />`
`<b>` `<i>` `<u>`	Used to format text in a bold, italic, or underline font.	`This is <b>bold text</b> while` `<i>this text is in italic</i>`
`<form>` `<input>` `<textarea>` `<select>`	Used for input forms that enable users to submit information to the server.	`<input type="text" value="Some Text" />`

continues

(continued)

TAG	DESCRIPTION	EXAMPLE

`<table>`
`<tr>`
`<td>`
	These tags are used to create a layout with a table. The `<table>` tag defines the entire table, and the `<tr>` and `<td>` tags are used to define rows and cells, respectively.	`<table>`
		`<tr>`
		`  <td>This is a Cell in Column 1</td>`
		`  <td>This is a Cell in Column 2</td>`
		`</tr>`
		`</table>`

`<ul>`
`<ol>`
`<li>`
	These three tags are used to create numbered or bulleted lists. The `<ul>` and the `<ol>` tags define the looks of the list (either unordered, with a simple bullet, or ordered, with a number), and the `<li>` tag is used to represent items in the list.	`<ul>`
		`  <li>First item with a bullet</li>`
		`  <li>Second item with a bullet</li>`
		`</ul>`
		`<ol>`
		`  <li>First item with a number</li>`
		`  <li>Second item with a number</li>`
		`</ol>`

`<span>`	This tag is used to wrap and influence other parts of the document. It appears as *inline*, so it adds no additional line break on the screen.	`<p>This is some normal text while`
		`<span style="color: red;">this text`
		`appears in red</span></p>`

`<div>`	Just like the `<span>` tag, the `<div>` tag is used as a container for other elements. However, the `<div>` acts as a *block* element, which causes an explicit line break after the `<div>` element by default.	`<div>`
		`  This is some text on 1 line`
		`</div>`
		`<div>`
		`  This text is put directly under the`
		`  previous text on a new line.`
		`</div>`

HTML Attributes

In addition to the HTML elements, the examples in the preceding table also showed you HTML *attributes*. Attributes contain additional information that changes the way a specific element behaves. For example, with the `<img>` tag that is used to display an image, the `src` attribute defines the source of that image. Similarly, the `<span>` tag contains a `style` attribute that changes the color of the text to red. The value of the `style` attribute (`color: red;`) is part of a *Cascading Style Sheet (CSS)*, which is discussed in much more detail in Chapter 3. Just as with the HTML elements, there is a long list of available attributes on the W3C web site: `www.w3.org/TR/html401/index/attributes.html`.

You don't need to memorize all these elements and attributes. Most of the time, they are generated for you automatically by VWD. In other cases, where you need to enter them by hand, VWD offers you *IntelliSense* to help you find the right tag or attribute. IntelliSense is discussed in the next chapter.

The Difference Between HTML and XHTML

In addition to HTML, you may also run into the term *XHTML*. Although the two have very similar names, they have some interesting differences that you need to be aware of. XHTML is a reformulation of HTML in *XML — eXtensible Markup Language*. This is a generic, text- and tag-based language used to describe data and is used as the base language for many other languages, including XHTML.

So, XHTML is in fact largely just HTML rewritten with XML rules. These rules are pretty simple, and most of the time VWD will help you get it right or show you a list of errors and suggestions on how to fix them.

Always Close Your Elements

In XHTML, all elements must be closed. So when you start a paragraph with `<p>`, you must use `</p>` somewhere later in your page to close the paragraph. This is also the case for elements that don't have their own closing tags, like `<img>` or `<br>` (to enter a line break). In XHTML, these tags are written as *self-closing tags*, where the closing slash is embedded directly in the tag itself as in `<img src="Logo.gif" />` or `<br />`.

Always Use Lowercase for Your Tag and Attribute Names

XML is case sensitive, and XHTML applies that rule by forcing you to write all your tags in lowercase. Although the tags and attributes must be in all lowercase, the actual value doesn't have to be. So, the preceding example that displays the logo image is perfectly valid XHTML, despite the uppercase L in the image name.

Always Enclose Attribute Values in Quotes

Whenever you write an attribute in a tag, make sure you wrap its value in quotes. For example, when writing out the `<img>` tag and its `src` attribute, write it like this:

```
<img src="Logo.gif" />
```

And not like this:

```
<img src=Logo.gif />
```

You could also use single quotes to enclose the attribute value, as in this example:

```
<img src='Logo.gif' />
```

It's also sometimes necessary to nest single and double quotes. When some special ASP.NET syntax requires the use of double quotes, you should use single quotes to wrap the attribute's value:

```
<asp:Label ID="TitleLabel" runat="server" Text='<%# Eval("Title") %>' />
```

You'll see this syntax used a lot more in later chapters in this book.

For consistency, this book uses double quotes where possible in all HTML that ends up in the client.

Nest Your Elements Correctly

When you write nested elements, make sure that you first close the inner element you opened last, and then close the outer element. Consider this correct example that formats a piece of text with both bold and italic fonts:

```
<b><i>This is some formatted text</i></b>
```

Notice how the `<i>` tag is closed before the `<b>` tag. Swapping the order of the closing tags leads to invalid XHTML:

```
<b><i>This is some formatted text</b></i>
```

Always Add a DOCTYPE Declaration to Your Page

A `DOCTYPE` gives the browser information about the kind of HTML it can expect. By default, VWD adds a `DOCTYPE` for XHTML 1.0 Transitional to your page:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

The `DOCTYPE` greatly influences the way browsers like Internet Explorer render the page. VWD's default `DOCTYPE` of XHTML 1.0 Transitional gives you a good mix between valid markup and pages that render the same in all major browsers.

NOTE *If you want to learn more about XHTML, get a copy of Wrox's* Beginning Web Programming with HTML, XHTML, and CSS, 2nd Edition*, ISBN: 978-0-470-25931-3.*

Besides HTML, an ASP.NET web page can contain other markup as well. Most pages will have one or more ASP.NET Server Controls on the page to give it some additional functionality. The next section briefly looks at these ASP.NET Server Controls, but you get an in-depth look at them in Chapter 4.

A First Look at ASP.NET Markup

To some extent, the markup for ASP.NET Server Controls is similar to that of HTML. It also has the notion of tags, elements, and attributes, using the same angle brackets and closing tags as HTML does. However, some differences also exist.

For starters, most of the ASP.NET tags start with an `asp:` prefix. For example, a button in ASP.NET looks like this:

```
<asp:Button ID="Button1" runat="server" Text="Click Me" />
```

Note how the tag is self-closed with the trailing slash (/) character, eliminating the need to type a separate closing tag.

Another thing you may have noticed is that the tag and attribute names are not necessarily in all lowercase. Because an ASP.NET Server Control lives on the server, it doesn't have to adhere to the XHTML rules used in the browser at the client. However, when a Server Control is asked to emit its HTML to a page that is configured to output XHTML, it will do so in XHTML. So, the code for the same button looks like this when rendered in the browser as XHTML:

```
<input type="submit" name="Button1" value="Click Me" id="Button1" />
```

Notice how the entire tag and its attributes conform to the XHTML standard. The process of converting the Server Control to its HTML representation is similar to the code you saw earlier that displayed the current date. The Server Control is processed at the server by the ASP.NET handler. This processing results in HTML, which is sent to the browser where it's displayed.

Now that you understand the basics of an ASP.NET page and the HTML that it generates, it's time to look at VWD again. Knowing how to use the application and its many tools and windows is an important step in building fun, good-looking, and functional web sites.

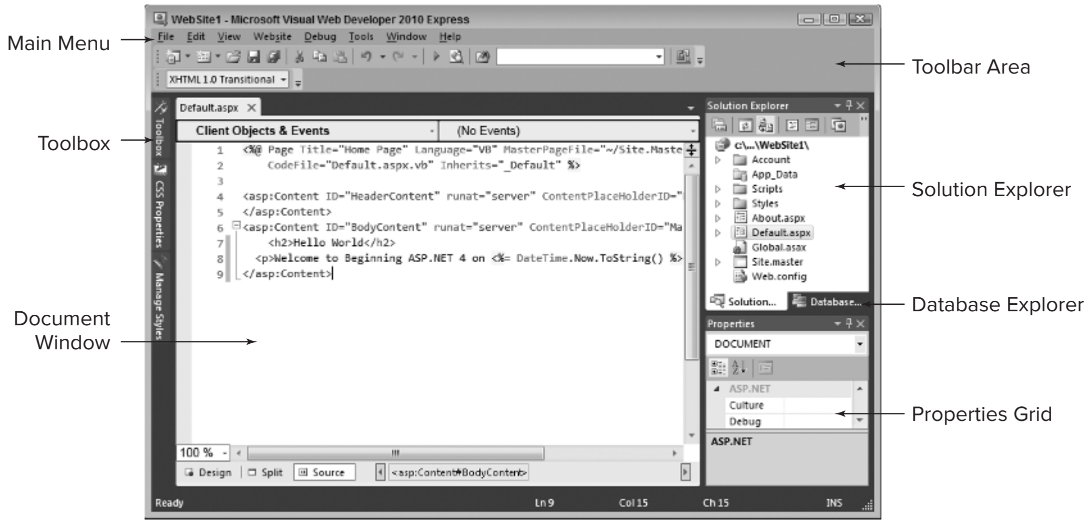
A TOUR OF THE IDE

VWD is by far the most extensive and feature-rich *integrated development environment* (IDE) for building ASP.NET web pages. The abbreviation IDE refers to the way all the separate tools you need to build complex web applications are integrated in a single environment. Instead of writing code in a text editor, compiling code at the command line, writing HTML and CSS in a separate application, and managing your database in yet another, VWD enables you to perform all of these tasks, and more, from the same environment. Besides the efficiency this brings because you don't have to constantly switch tools, this also makes it much easier to learn new areas of VWD, because many of the built-in tools work in the same way.

The Main Development Area

To get familiar with the many tools that are packed in VWD's interface, take a look at Figure 1-8. It shows the same screen you got after you created your first web site in VWD, but now it highlights some of the most important screen elements. If you are already familiar with a previous version of Visual Web Developer, you could skip this section and pick up again at the next Try It Out exercise later in this chapter.

If you had a previous version of Visual Studio installed, your screen may look different, because Visual Studio 2010 is able to import settings from older versions.



```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Maste
    CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="Ma
    <h2>Hello World</h2>
  <p>Welcome to Beginning ASP.NET 4 on <%= DateTime.Now.ToString() %>
</asp:Content>
```

FIGURE 1-8

Choosing Your Development Profile

Because Visual Web Developer Express targets people new to ASP.NET development as well as seasoned web developers, you can choose among different developer profiles: Basic Settings, Code Optimized, and Expert Settings. In Basic Settings mode, many menu items you don't frequently use have been hidden or are placed in their own sub menu. The Code Optimized profile is great for pure coding sessions where you're not interested in many of the design features of VWD. It hides items like the Toolbox and the Properties Grid, both shown in Figure 1-8. Expert Settings mode gives you access to the full functionality of VWD. You can switch between settings using the Tools ➪ Settings menu. This book assumes you are using Expert Settings mode right from the beginning. You may not need all features you see right from the start but you sure will use most of them by the end of the book. Since the menu items change location depending on the profile you choose, I decided to use Expert Settings mode right away, to make it easier to refer to a specific menu item or feature.

The Main Menu

At the top of the application, right below the Windows title bar, you see the main menu. This menu bar contains familiar items you find in many other Windows applications, like the File, Edit, and Help menus as well as menus that are specific to VWD, such as the Website and Debug menus. The menu changes dynamically depending on the task you're working on, so you'll see menu items appear and disappear as you work your way through the application. You can use the Help ➪ Manage Help Settings menu to configure online and offline help. Offline help needs to be installed first, while online help requires a connection to the Internet.

The Toolbar Area

Right below the menu, you see the toolbar area that is capable of showing different toolbars that give you quick access to the most common functions in VWD. In Figure 1-8, only two of the toolbars are enabled, but VWD comes with many other toolbars that you can use in specific task-oriented scenarios. Some toolbars appear automatically when you're working on a task that requires a particular toolbar's presence, but you can also enable and disable toolbars to your liking. To enable or disable a toolbar, right-click an existing toolbar or the menu bar and choose the toolbar from the menu that appears.

The Toolbox

On the left of the main screen, tucked away at the border of VWD, you see the tab for the Toolbox. If you hover your mouse over the tab, the Toolbox folds out, giving you a chance to see what it contains. If you click the little pin icon in the upper-right corner of the Toolbox (or any of the other panels that have this pin icon), it gets pinned to the IDE so it remains open.

Just as with the menu bar and the toolbars, the Toolbox automatically updates itself to show content that is relevant to the task you're working on. When you're editing a standard ASPX page, the Toolbox shows the many controls you have available for your page. You can simply drag an item from the Toolbox and drop it on a location of your page where you want it to appear. These controls are discussed in great detail in Chapter 4.

The Toolbox contains multiple categories with tools that can be expanded and collapsed as you see fit to make it easier to find the right tool. You can also reorder the items in the list, add and remove items from the Toolbox, and even add your own tools to it. Customizing the IDE is discussed later in this chapter.

If the Toolbox is not visible on-screen, press Ctrl+Alt+X to open it or choose View ➪ Toolbox, provided you have chosen the Expert Settings option in the Tools ➪ Settings menu.

The two additional tabs below the Toolbox tab, CSS Properties and Manage Styles, are discussed extensively in Chapter 3.

The Solution Explorer

At the right of the screen, you see the Solution Explorer. The Solution Explorer is an important window because it gives you an overview of the files that comprise your web site. Instead of placing all your files in one big folder, the Solution Explorer enables you to store files in separate folders, creating a logical and organized site structure. You can use the Solution Explorer to add new files to your site, move existing files around using drag and drop or cut and paste, rename files and delete them from the project, and more. Much of the functionality of the Solution Explorer is hidden behind its right-click menu, which changes depending on the item you right-click.

At the top of the Solution Explorer, you see a small toolbar that gives you quick access to some functionality related to your web site, including opening the Properties Grid for the selected item, refreshing the Solution Explorer window, an option to nest related files, and two buttons that allow you to copy and configure your web site. Most of this functionality is discussed later in the book.

You can access the Solution Explorer by choosing View ➪ Solution Explorer from the main menu or by pressing Ctrl+Alt+L.

The Database Explorer

This window, hidden behind the Solution Explorer in Figure 1-8, enables you to work with your databases. If you have a commercial version of Visual Studio, such as Visual Studio 2010 Professional, this window is called the Server Explorer and may be located at the left of your screen.

The Database Explorer is discussed in more detail in the chapters about databases, starting with Chapter 12.

The Properties Grid

With the Properties Grid, you can view and edit the properties of many items in Visual Studio, including files in the Solution Explorer, controls on a web page, properties of the page itself, and much more. The window constantly updates itself to reflect the selected item. You can quickly open the Properties Grid by pressing F4. This same shortcut can be used to force the Properties Grid to show the details of a selected item.

The Document Window

The Document Window is the main area in the middle of the application. This is where most of the action takes place. You can use the Document Window to work with many different document formats, including ASPX and HTML files, CSS and JavaScript files, code files for VB and C#, XML and text files, and even images. In addition, you can use the same window to manage databases, create copies of your site, view the pages in your site in the built-in mini-browser, and much more.

At the bottom of the Document Window in Figure 1-8, you see three buttons called Design, Split, and Source. These buttons appear automatically when you're working with a file that contains markup, such as ASPX and HTML pages. They allow you to open the Design View of a page (giving you an idea of how the page will look in the browser), its Markup View (the HTML and other markup), or both at the same time. How this works is explained in more detail in Chapter 2, but for now, it's important to realize you can switch between Markup, Split, and Design View by clicking the appropriate buttons. The Markup View is also often called the Source View or Code View window. However, to avoid confusion with the code editor that is used to edit Code Behind files, this book uses the term Markup View exclusively.

The Document Window is a tabbed window by default, which means it can host multiple documents, each one distinguished by a tab with the file name at the top of the window. The right-click menu of each tab contains some useful shortcuts for working with the file, including saving and closing it and opening the file's parent folder in Windows Explorer.

To switch between documents, you can press Ctrl+Tab or you can click the down arrow in the upper-right corner of the Document Window, next to the Solution Explorer, shown in Figure 1-8. Clicking the down arrow reveals a list of open documents so you can easily select one.

Another way to switch documents is to press Ctrl+Tab and then hold down the Ctrl key. On the window that pops ups, you can select a document you want to work with in the right-hand column. You can then use the cursor keys to move up and down in the list with open documents. This makes it super easy to select the correct file.

On the same dialog box, you see a list with all active tool windows. Clicking one of the windows in the list will show it on-screen, moving it in front of other windows if necessary.

The Start Page

Whenever you start up VWD, the Start Page is loaded in the Document Window. With the Start Page, you can quickly create new or open existing web sites and other projects. The Start Page also provides a number of links to related news and information about web development.

To get a feel for how you can use all these windows, the following Try It Out shows you how to build a simple web page that contains a few ASP.NET Server Controls.

TRY IT OUT Creating Your First ASP.NET Web Page

This Try It Out exercise guides you through creating a new web site with a single page that contains a number of ASP.NET Server Controls. You see how to use windows like the Document Window and the Solution Explorer, and how to use the Toolbox and the Properties Grid to add ASP.NET Server Controls to the page and change their looks.

1. Make sure Visual Web Developer 2010 is started.
2. If you're using the Express edition, choose Tools ➪ Settings and choose Expert Settings to turn on the developer profile that gives you access to the full feature set of VWD.
3. On the File menu choose New Web Site. If you are using a commercial version of Visual Studio, you may have to choose File ➪ New ➪ Web Site instead. This triggers the New Web Site dialog box.
4. In this dialog, make sure that ASP.NET Empty Web Site is selected and not the ASP.NET Web Site item that you used in a previous exercise. Ensure that File System is chosen in the Web Location drop-down list. Click OK to create the new site.
5. Next, right-click the new web site in the Solution Explorer. Make sure you click the uppermost element that says something like `C:\..\WebSite2\`. It's the highlighted element in Figure 1-4. From the context menu that appears, choose Add New Item.
6. In the new window that appears, click Web Form and type **`ControlsDemo`** as the name. The ASPX extension is added for you automatically when you click the Add button. You can leave the other settings in the dialog box at their default settings. The page should open in Markup View, showing you the default HTML, like the `<html>`, `<head>`, `<title>`, and `<body>` elements that Visual Web Developer adds there for you automatically when you create a new page.
7. Switch the page to Design View by clicking the Design button at the bottom of the Document Window.
8. If the Toolbox isn't open yet, press Ctrl+Alt+X to open it or hover your mouse over the Toolbox tab to show it and then click the pin icon to make the Toolbox visible at all times. Drag a `TextBox` and a `Button` from the Toolbox into the dashed area in the Design View of the page. You should end up with a Design View that looks similar to Figure 1-9.

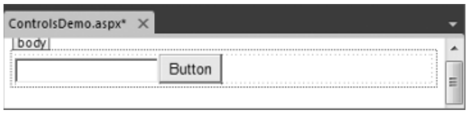


FIGURE 1-9

9. Right-click the button in Design View and choose Properties. In the Properties Grid, locate the Text property under the Appearance category (shown in Figure 1-10) and change it from Button to Submit Information. As soon as you press Tab or click somewhere outside the Properties Grid, the Design View of the page is updated and shows the new text on the button.

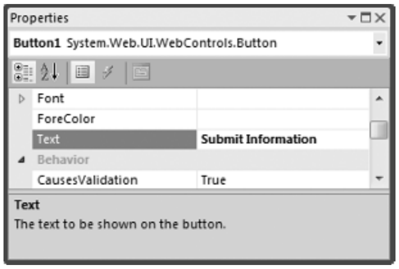


FIGURE 1-10

10. Press Ctrl+F5 to open the page in your default browser. Note that it's not necessary to explicitly save the changes to your page (although it's a good idea to do this often anyway using the shortcut Ctrl+S). As soon as you press Ctrl+F5 to run the page, VWD saves all changes to open documents automatically.

NOTE *If you don't like this behavior, you can change it. Choose Tools ➪ Options from the main menu. Then make sure that Show All Settings is checked, open the Projects and Solutions node, and choose Build and Run. In the Before Building list, you can change the way VWD behaves when you open a page in your browser.*

11. Type some text in the text box and click the button. Note that after the page has reloaded, the text is still displayed in the text box. Other than that, not much has happened because you didn't write any code for the button yet.

How It Works

When you dragged the `Button` and the `TextBox` from the Toolbox on the page in Design View, VWD added the corresponding code for you in Markup View automatically. Similarly, when you changed the `Text` property of the button in the Properties Grid, VWD automatically updated the markup for the control in Markup View. Instead of using the Properties Grid, you could also have typed the text directly between the quotation marks of the `Text` property in Markup View.

After changing the `Text` property, your page should now contain this code in Markup View:

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Submit Information" />
```

When you press Ctrl+F5 to view the page in the browser, the web server receives the request, the page is processed by the ASP.NET runtime, and the resulting HTML for the page is sent to the browser.

After you type in some text and click the button, the same process is more or less repeated: the web server receives the request, the page is processed, and the result gets sent back to the browser. When you click the button, you cause a *postback* to occur, where any information contained in the page — such as the text you typed in the text box — is sent back to the server. ASP.NET reacts to the postback by rendering the page again. However, this time it prepopulates controls, like the `TextBox`, with the values that were sent to the page.

Take a look at the resulting HTML for the page using the browser's View Source command (rerun the page from VWD by pressing Ctrl+F5 if you already closed it). You should see code similar to this:

```
<input name="TextBox1" type="text" value="Hello World" id="TextBox1" />
<input type="submit" name="Button1" value="Submit Information" id="Button1" />
```

Just as with the earlier example, you can see that the resulting HTML is substantially different from the original ASPX markup.

Postbacks are an important concept in ASP.NET, and you see more about them in other chapters, including Chapters 4 and 9.

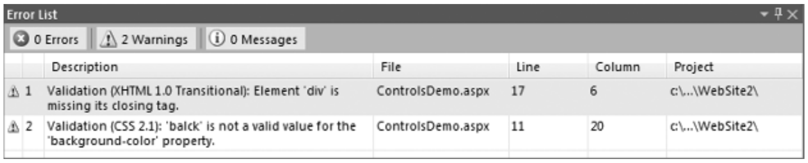
VWD hosts many more windows and tool panels than those you have seen so far. The next section briefly touches upon some of the windows you'll most frequently use when building ASP.NET web pages. All of the windows mentioned are accessible from the main View menu in VWD if you're using the Expert Settings mode.

Informational Windows

In addition to the windows that are visible by default when you start VWD, many more windows are available in VWD. You see most of them in action in the remainder of this book, but some are worth highlighting now. You access all windows that are discussed next from the main View menu.

The Error List

The Error List gives you a list of the things that are currently somehow broken in your site, including incorrect markup in your ASPX or HTML files and programming errors in VB or C# files. This window can even show you errors in XML and CSS files. The Error List shows its messages in three categories — Errors, Warnings, and Messages — that signify the severity of the problem. Figure 1-11 shows the error list for a page that has some problems with its CSS and XHTML.



		Description	File	Line	Column	Project
⚠	1	Validation (XHTML 1.0 Transitional): Element 'div' is missing its closing tag.	ControlsDemo.aspx	17	6	c:\...\WebSite2\
⚠	2	Validation (CSS 2.1): 'balck' is not a valid value for the 'background-color' property.	ControlsDemo.aspx	11	20	c:\...\WebSite2\

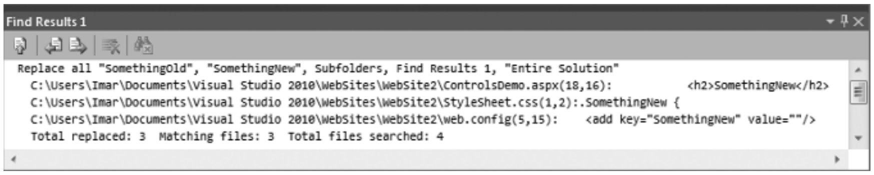
FIGURE 1-11

The Output Window

When you try to build your site using the Build menu, the Output window tells you whether or not the build succeeded. If the build failed, for example because you have a programming error, it tells you why the build failed. In the commercial versions of Visual Studio, the Output window is used for other information as well, including the status of external plug-in programs. Building — or compiling — web sites is discussed later in this book, including Chapter 19, which deals with deployment of your web site.

The Find Results Window

The Find and Replace features of VWD are invaluable tools when it comes to managing the content of your site. You will often need to replace some text in the current document or even in the entire site. Find in Files (Ctrl+Shift+F) and Replace in Files (Ctrl+Shift+H) both output their results in the Find Results window, as shown in Figure 1-12.



```
Replace all "SomethingOld", "SomethingNew", Subfolders, Find Results 1, "Entire Solution"
  C:\Users\Imar\Documents\Visual Studio 2010\WebSites\WebSite2\ControlsDemo.aspx(18,16):          <h2>SomethingNew</h2>
  C:\Users\Imar\Documents\Visual Studio 2010\WebSites\WebSite2\StyleSheet.css(1,2):.SomethingNew {
  C:\Users\Imar\Documents\Visual Studio 2010\WebSites\WebSite2\web.config(5,15):    <add key="SomethingNew" value=""/>
  Total replaced: 3  Matching files: 3  Total files searched: 4
```

FIGURE 1-12

Because having several informational windows open at the same time may take up precious screen space, it's often a good idea to dock them. This way, only one of them is visible at a time, while you still have quick access to the others. You learn how to customize the IDE, including the docking of windows, next.

CUSTOMIZING THE IDE

Although the standard setup of VWD and its tool windows is pretty useful, there's a fair chance you want to customize the IDE to your liking. You may want to rearrange some of the windows to a location where they are easier to reach, or you may want to open additional windows you frequently use. VWD is fully customizable and enables you to tweak every little detail of the IDE. In the next section, you learn how to perform the most common customization tasks.

Rearranging Windows

To give each window the location it deserves, you can drag and drop them in the main IDE. Simply grab a window's title bar or its bottom tab and drag it in the direction of the new location. Once you start dragging, you'll see that VWD gives you visual cues as to where the window will end up (see Figure 1-13).

If you drag the window over one of the four square indicators at the sides of the indicator, VWD shows a preview of how the window will be docked *next to* an existing window. Once you drop it, the window will pop to its new location. If you drop the window on the square in the middle of the large indicator, the window will dock *with* that window, sharing the same screen space. Each window has its own tab, as can be seen with the windows at the bottom of Figure 1-13.

In addition to docking windows with others in the IDE, you can also have floating windows. To change a docked window into a floating one, either drag it away from its current location and drop it somewhere in the IDE without hitting one of the visual cues on the screen or choose Window ➪ Float from the main menu.

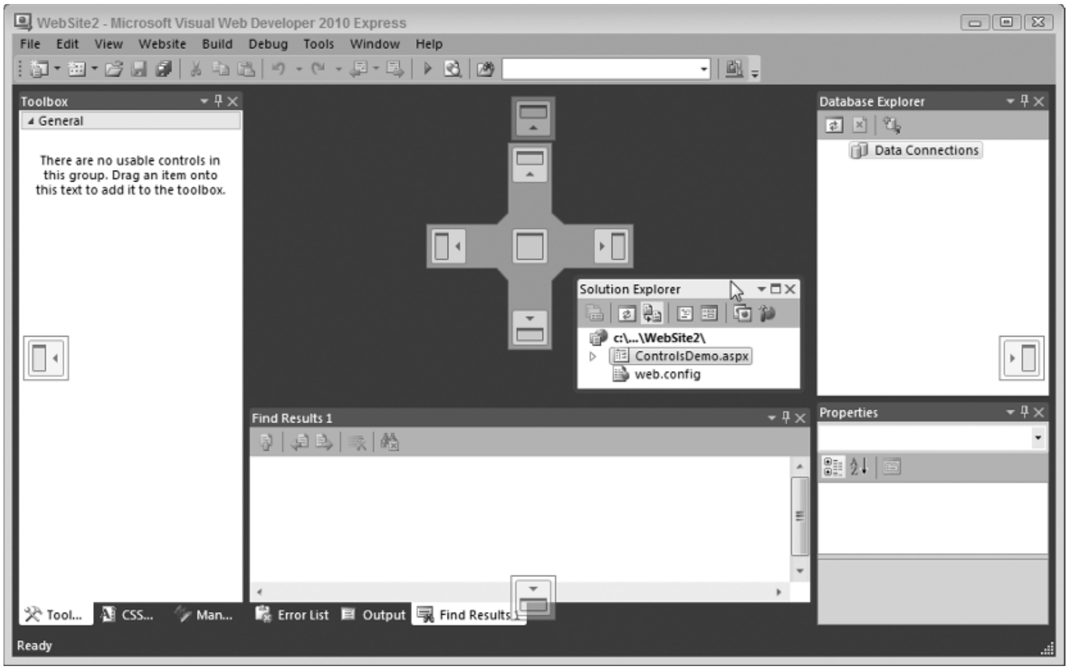

WebSite2 - Microsoft Visual Web Developer 2010 Express
File Edit View Website Build Debug Tools Window Help
Toolbox
General
There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.
Database Explorer
Data Connections
Solution Explorer
c:\...\WebSite2\
ControlsDemo.aspx
web.config
Find Results 1
Properties
Tool... CSS... Man... Error List Output Find Results 1
Ready

FIGURE 1-13

To restore a floating panel to its previous docked location, choose Window ➪ Dock from the main menu. Make sure you don't choose Dock as Tabbed Document for the tool windows like the Toolbox or the Solution Explorer or they'll end up in the main Document Window. This makes it difficult to use these tool windows together with an open file as the two windows will share the same space.

Modifying the Toolbox

The Toolbox can be modified as well. By default, the items are sorted alphabetically but you can reorder them using drag and drop. To do this, open the Toolbox (press Ctrl+Alt+X), and drag an item (such as the TextBox under the Standard category) to a different location. You can also delete items from the Toolbox by right-clicking them and choosing Delete from the context menu. Don't worry about items getting lost forever; you can reset the Toolbox again by choosing Reset Toolbox from the same menu.

You can also add your own items to the Toolbox. The most common use for this is code snippets. Simply highlight some text or code in the Document Window and drag it to the Toolbox. You can then right-click the item and choose Rename Item to give it a more meaningful name that you can easily recognize.

To avoid cluttering up the Toolbox with your own code snippets, consider creating a separate category for them. You can do this by choosing Add Tab from the Toolbox's right-click menu. Enter a name and press Enter, and your Toolbox tab is ready for use.

In the next Try It Out exercise, you get the chance to play around with the VWD IDE so you can customize it to your liking.

TRY IT OUT Customizing the IDE

In this exercise you practice with opening and rearranging the windows in the Visual Web Developer IDE. Don't be afraid to mess up the IDE. A little later in this chapter, instructions are given on how to reset the IDE to the way it was when you opened it the first time.

1. If you closed your web site since the previous Try It Out, open it again, or create a new one using the File menu.
2. From the View menu, choose Error List to open the Error List window. If you don't see the Error List item directly, choose Tools ➪ Settings ➪ Expert Settings first. Notice how the Error List gets docked below the Document Window by default.
3. From the same View menu, choose Task List. By default, it will be docked in the same space as the Error List, with the tabs for both windows next to each other.
4. Click the tab of the Task List and while holding down your mouse button, drag the Task List away from its location in the direction of the Document Window. Once you release the window, it will appear as a floating window in the IDE. To restore the window, right-click its title bar and choose Dock. Notice how the tab returns to the same tab group, but possibly at a different position. To change the order in which tabs appear in a tab group, drag a tab over the other tabs and release it at the desired location.
5. If you want, you can repeat the previous steps for other windows that are visible in the IDE by default or for the ones you find under the View menu. Spend some time familiarizing yourself with all the different windows and how you can arrange them on-screen. Because you'll be working a lot with these windows in the remainder of this book, it's good to be familiar with their locations.
6. Next, open the `ControlsDemo.aspx` page (or add a new ASPX first if you created a new web site) from the Solution Explorer by double-clicking it. When the page opens, the Toolbox becomes visible automatically. If it doesn't, press Ctrl+Alt+X to open it.
7. Right-click the Toolbox and choose Add Tab. Type **`HTML Fragments`** as its new name and press Enter. This adds a new category to the Toolbox that behaves just like all the others.
8. With the Document Window showing your ASPX page in Markup View, type `<h1>` directly after the opening `<div>` tag. Note that VWD automatically inserts the closing `</h1>` for you. You should end up with code in Markup View looking like this:

```
<form id="form1" runat="server">
  <div>
    <h1></h1>
  </div>
```

9. Highlight the opening and closing `<h1>` tags, and then drag the selection from the Markup View window onto the new Toolbox tab you created in step 7. The selection shows up as Text: `<h1></h1>`.
10. Right-click the Toolbox item you just created, choose Rename Item, and type **`Heading 1`** as the name.

11. Put your cursor in the Document Window again and press Ctrl+K directly followed by Ctrl+D to format the document in the Document Window. Alternatively, choose Edit ➪ Format Document from the main menu. This formats the document according to the rules you have set in the Text Editor options dialog. Formatting is also available for a number of other document types, including C# and VB.NET code and XML files.

From now on, whenever you need a heading in your document in Markup View, simply place the cursor in the Document Window where you want the heading to appear and double-click the appropriate heading in the Toolbox.

How It Works

Most of the steps in this Try It Out are self-explanatory. You started off by opening a few windows that you frequently need when building web sites. You then used the drag-and-drop features of the IDE to rearrange the window layout to your personal preferences.

You then added an HTML fragment to a custom tab in the Toolbox. When you drag any markup to the Toolbox, VWD creates a Toolbox item for it that contains the selected markup. Whenever you need a copy of that markup in your page, simply double-click the item or drag it from the Toolbox into Markup View. This is a great time saver for HTML fragments that you frequently use. You typically use this technique for larger blocks of code; for elements like the `<h1>` VWD has a better tool called Code Snippets, which you meet later in this book.

At the end you used VWD's document formatting option to change the layout of the code in the document. This helps to keep the code organized and easier to read. You can fully change how the code is formatted by using the options dialog accessible through Tools ➪ Options. Then expand the path Text Editor ➪ HTML ➪ Formatting, and click Tag Specific Options.

Besides the Window layout and the Toolbox, VWD enables you to customize a lot more in the IDE. The following section explains how to customize three other important IDE features: the Document Window, toolbars, and keyboard shortcuts.

Customizing the Document Window

Visual Web Developer gives you great flexibility with regard to how text is displayed in the Document Window. You can change things like font size, font color, and even the background color of the text. You can access the Font and Colors settings by choosing Tools ➪ Options, making sure that Show All Settings at the bottom of the dialog box is selected, and then choosing Environment ➪ Fonts and Colors.

One thing I like to customize in the Document Window is the tab size, which controls the number of spaces that are inserted when indenting code. To change the tab size, choose Tools ➪ Options, and then under Text Editor choose All Languages ➪ Tabs. If you don't see this option, choose Show All Settings at the bottom first. I usually set the Tab and Indent Size to 2 spaces, leaving the other settings in the Tab panel untouched. Another thing I like to customize is the number of line breaks before and after HTML elements. The Options window gives you full control over this by selecting Text Editor ➪ HTML ➪ Formatting and then clicking Tag Specific Options.

With the exception of the Tab Size being set to 2 and the number of line breaks around a few HTML elements, all screen shots in this book show the default setup of Visual Web Developer.

Customizing Toolbars

Toolbars can be customized in three ways: you can show or hide the built-in toolbars, you can add and remove buttons on existing toolbars, and you can create your own toolbars with buttons you often use.

Enabling and Disabling Toolbars

You disable and enable existing toolbars by right-clicking any existing toolbar or the menu bar and then selecting the appropriate item from the list. Once the toolbar is displayed, you can use its drag grip at its left to drag it to a new location within the Toolbar area.

Editing Existing Toolbars

If you feel that an existing toolbar is missing an important button or that it contains buttons you rarely use, you can customize the buttons on the toolbar. To do this, right-click any toolbar or the menu bar, choose Customize, switch to the Commands tab and select the toolbar you want to modify from the Toolbar drop-down. With the command buttons at the right, you can add new and remove existing commands, or change their order.

If you want to move a toolbar to the left, right or bottom of the window, switch back to the Toolbars tab of the Customize window, select the toolbar and click Modify Selection.

Creating Your Own Toolbars

Creating your own toolbar is useful if you want to group some functions that you frequently use. To create a new toolbar, open the Customize window as explained in the preceding section. Click the New button and type a name for the toolbar. Then switch to the Commands tab and modify your toolbar as you would do with existing toolbars.

Customizing Keyboard Shortcuts

Another setting many developers like to change is keyboard shortcuts. Keyboard shortcuts are a good way to save time because they allow you to perform a task with a simple keyboard command instead of reaching for the mouse and selecting the appropriate item from the menu. To change the keyboard shortcuts, choose Tools ➪ Options, expand Environment, and click Keyboard. Locate the command for which you want to change the shortcut in the list with commands. Because this list contains many items, you can filter the list by typing a few letters from the command. For example, typing `print` in the Show Commands Containing field gives you a list of all print-related commands.

Next, in the Press Shortcut Keys field, type a new shortcut and click Assign. VWD allows you to enter a double shortcut key for a single command. For example, you can bind the command Close All Documents to the command Ctrl+K, Ctrl+O. To perform this command, you need to press both key combinations in rapid succession. Although a double shortcut key may seem like overkill, it greatly increases the number of available shortcut keys.

Resetting Your Changes

Don’t worry if you feel that you have messed up VWD by trying out the numerous customization options. You have many ways to restore VWD to its previous state.

Resetting the Window Layout

This setting, accessible from the Window menu, resets all windows to the position they were in when you first started VWD. This command is useful if you misplaced too many windows and ended up with a cluttered IDE.

Resetting the Toolbox

If you removed an item from the Toolbox by mistake or even deleted an entire tab, you can reset the Toolbox to its original state by right-clicking the Toolbox and choosing Reset Toolbox. You need to think twice before you use this command because it will also delete all your custom code snippets.

Resetting All Settings

If you followed along with the previous Try It Out exercises, and then started experimenting with the customization possibilities, your IDE is now probably in one of two states: it either looks exactly the way you want it, or it looks like a complete mess. In the latter case, it’s good to know that it is easy to clean up the chaos.

To completely revert all VWD settings to the way they were right after installation, choose Tools ➪ Settings ➪ Import and Export Settings or Tools ➪ Import and Export Settings, depending on the version of Visual Web Developer you're using. Next, choose the Reset All Settings option and click Next. If you want, you can create a backup of the existing settings; otherwise, choose No, Just Reset Settings. You get another screen that enables you to choose among a number of settings collections. Choose Expert Settings or Web Development because these options give you access to all features you need to follow along with this book. Finally, click Finish. This action will cause all settings to be reset to their defaults, including the Windows layout, Toolbox and Toolbox customizations, shortcut keys, and everything you may have changed in the VWD Options dialog box. So, use this command only when you’re really sure you want a fresh, new setup of VWD.

With some basic knowledge about ASP.NET pages and VWD, it’s time for some real action. In the next chapter, you see how to create ASP.NET web sites and web pages in much more detail. You learn how to organize your site in a logical and structured way, how to add the many different types of files to your site and how to use them, and how to connect the pages in your site.

However, before you proceed to the next chapter, there is one more important topic you need to look at: the sample application that comes with this book.

THE SAMPLE APPLICATION

Building web sites is what this book is all about, so it makes a whole lot of sense that this book comes with a complete and functional sample site that is used to showcase many of the capabilities of ASP.NET.

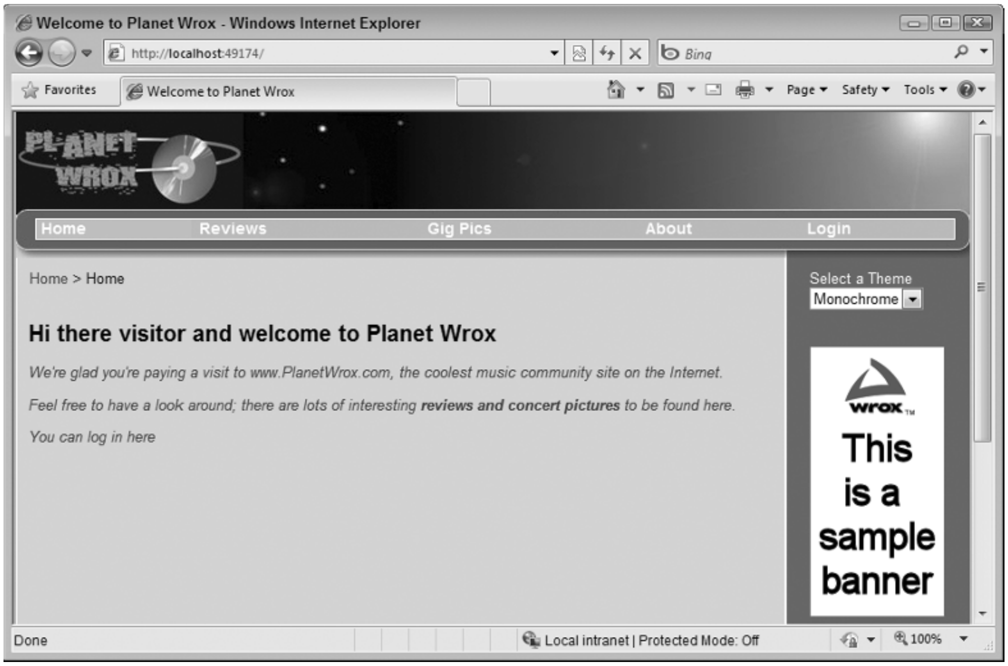
The sample site you build in this book is called Planet Wrox, a site that serves as an online community for people interested in music. The site offers the following features to its visitors:

- Reviews about CDs and concerts that have been posted on the site by the administrator.
- The Gig Pics section, an online photo album where users can share pictures taken at concerts.
- The ability to switch between the different graphical themes that the site offers, giving you a chance to change the look and feel of the site without altering the content.
- Store musical preferences that influence the information users see on the site.
- Access to bonus features for registered users.

From an administrative perspective (that is you, as the owner of the site) the site enables you to do the following:

- Add and maintain the reviews.
- Manage the different musical genres in the system.
- Manage photo albums created by visitors to the site.

Figure 1-14 shows the Planet Wrox home page.



Hi there visitor and welcome to Planet Wrox

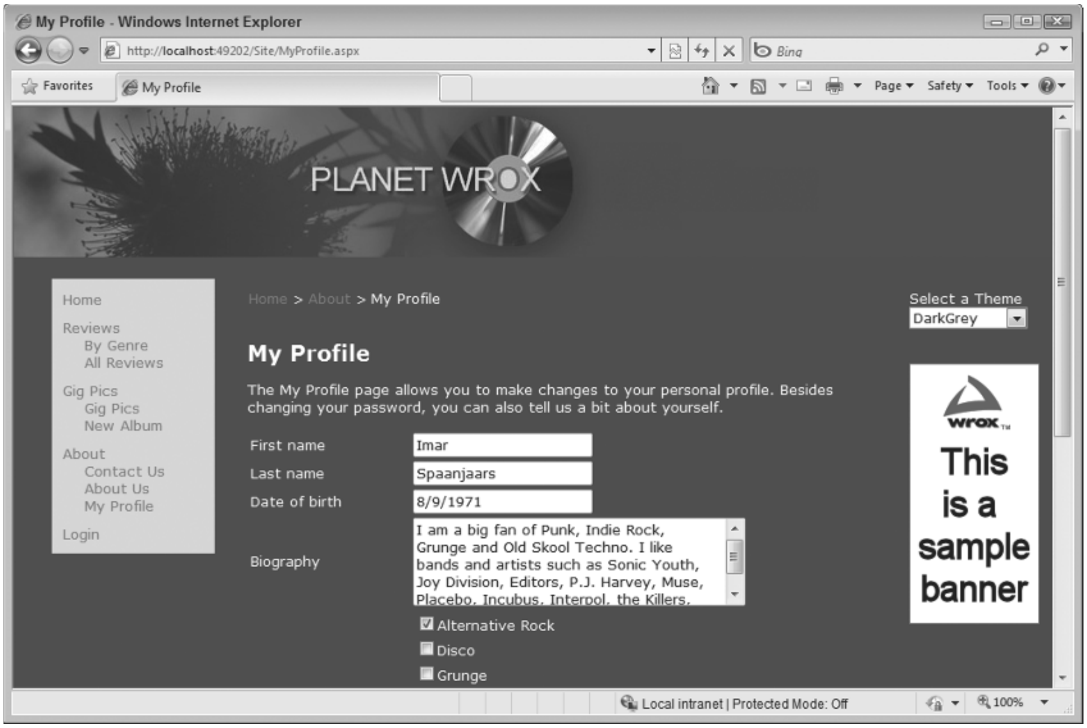
We're glad you're paying a visit to www.PlanetWrox.com, the coolest music community site on the Internet.

*Feel free to have a look around; there are lots of interesting **reviews and concert pictures** to be found here.*

You can log in here

FIGURE 1-14

Figure 1-15 shows another page from Planet Wrox, but with a different theme applied. This page enables users to enter their personal information and specify preferences with regard to their favorite musical genres.



My Profile

The My Profile page allows you to make changes to your personal profile. Besides changing your password, you can also tell us a bit about yourself.

First name	Imar
Last name	Spaanjaars
Date of birth	8/9/1971
Biography	I am a big fan of Punk, Indie Rock, Grunge and Old Skool Techno. I like bands and artists such as Sonic Youth, Joy Division, Editors, P.J. Harvey, Muse, Placebo, Incubus, Interpol, the Killers,

[✓] Alternative Rock
[ ] Disco
[ ] Grunge

FIGURE 1-15

You can find an online running example of the site at `www.PlanetWrox.com`. There you can play around with the site from an end user's perspective.

You can also download the source for the sample application and all other examples from this book from the Wrox web site at `www.wrox.com`.

By the end of this book, you'll be able to build all of the functionality from the sample site (and hopefully even more) in other web sites. Don't worry if it sounds like an awful lot of complex things. I guide you, step by step, from the beginning of the application all the way to the last feature. As long as you keep having fun doing this, I'm sure you'll make it all the way.

PRACTICAL TIPS ON VISUAL WEB DEVELOPER

Most of the chapters in this book end with a short section of useful tips. These are tips that either didn't fit in anywhere in the text or that encourage you to further explore or test out things. Sometimes they may seem irrelevant or hard to understand at first, but you'll find that as you make your way through this book and look back at tips from previous chapters, things start to make sense. Don't worry if you don't understand certain things completely the first time you see them. Give the idea some thought and revisit the topic a few days later. Hopefully, by letting the ideas sink in a little, things start to make more sense automatically. This applies not only to the Practical Tips section, but to the entire book.

- Before you move on to the next chapter, play around with VWD some more. Add a couple of pages to your site, drag and drop some controls from the Toolbox onto your pages, and view them in your browser. That way, you'll have a better understanding of the tools and the many controls available when you start the next chapter.
- Familiarize yourself with the many options to tweak the Visual Web Developer IDE. When building web sites, you spend most of your time in this IDE, so it makes sense to tweak it as much as possible to your liking. Don't be afraid to mess it up; you can always revert to previous settings.
- Take some time to browse through the settings you find in the Options dialog box of VWD (accessible through the Tools ➪ Options menu). Many of the settings are self-explanatory and can really help in further tweaking the IDE to your liking.

SUMMARY

This chapter covered a lot of important ground to get you started with ASP.NET 4 and VWD 2010. It started off with a brief history of the Microsoft .NET Framework in general and ASP.NET in particular.

You then learned how to acquire and install Visual Web Developer 2010 Express. VWD is the most extensive and versatile tool available for creating ASP.NET 4 web pages. To enable you to work with it effectively, this chapter showed you how to use and customize the main features of the IDE. In subsequent chapters, you use and extend this knowledge to work with the many tools found in VWD.

It's important to understand how a page in VWD makes it to your web browser. Some knowledge of the web server that serves the request and how the page is processed to deliver the final HTML in the browser is critical in understanding ASP.NET. This chapter gave you a short introduction in the way a web page is requested and served to the browser.

In the next chapter, you get a much more detailed explanation of creating web sites.

EXERCISES

1. Explain the differences between the markup of a page in VWD and the final HTML page in the browser.
2. Explain the difference between HTML and XHTML. How are the two related?
3. Imagine you have a number of HTML fragments that you expect to use a lot throughout the site. What's the best way to make these fragments available in VWD?
4. What are three of the ways you can reset part or all of the IDE customization settings?
5. If you want to change the property of a control on your page, for example the text of a button, which two options do you have available to make the change?

Answers to Exercises can be found in Appendix A.

▶ WHAT YOU LEARNED IN THIS CHAPTER

Attribute	Extra information in a tag to define or change its behavior
Element	A pair of tags holding some text or other content
HTML	HyperText Markup Language: the language that browsers use to display a web page
HTTP	HyperText Transfer Protocol: the protocol by which web browsers and web servers communicate
IDE	Integrated Development Environment: an integrated collection of applications and tools to develop applications
JavaScript	A programming language used to interact with a web page in the client's browser
Tag	Text surrounded by angle brackets to create HTML elements
Visual Studio 2010	The development environment to build .NET applications
Visual Web Developer	The part of Visual Studio (but also available separately as the free Express edition) that enables you to build ASP.NET web applications
XHTML	HTML rewritten with XML rules