

Chapter 1

ASP.NET 2.0 — Just Do It!

In This Chapter

- ▶ Identifying and installing the software you need
 - ▶ Creating an environment for developing and testing ASP.NET 2.0 pages
 - ▶ Planning how to create and execute a page
 - ▶ Exploring some common Visual Basic 2005 commands
 - ▶ Working with new pages and old ones
-

Most books (especially about .NET, it seems) give you lots of concepts and theories and detailed, boring analysis before they ever let you jump in and play with the stuff. Don't expect that here, though; I turn the traditional philosophy of chapter-order sequence on its head. In this book, I start by getting your server and development environment configured and set up so that you can create your first ASP.NET page in *this* chapter. There's no teacher like experience!

If questions pop up in your mind as you're going through this chapter, hang on! I probably address them in Chapter 2, where I take a step back and put what you've seen into context. There, I show you how ASP.NET came about and how it differs from other technologies you might have heard about.

Do I Need a Development Environment?

To work with most computer languages, you use a *development environment*. Usually this involves an editor that you use to write the source code and to access the utilities that make your development and debugging go faster and more smoothly. These environments put at your fingertips every tool you need to write, run, and test applications that you create.

ASP.NET is different. It's a development technology that is built into the .NET Framework. You can easily create great ASP.NET 2.0 applications with nothing more than Notepad or your favorite editor.

Of course, some integrated development environments (IDEs) *are* available for creating ASP.NET applications. The most prominent is Microsoft's own

Visual Studio 2005. This professional tool is full of features to aid in developing medium- and large-scale development projects. But it can be more than a little intimidating when you're first getting to know .NET.

Because of this, Microsoft has created an entry-level product that is much less intimidating and much more affordable: Microsoft Visual Web Developer 2005 Express Edition. This tool is relatively inexpensive and provides a lot of the best features of Visual Studio without all the stuff you don't care about when you're getting started or when you're working on smaller applications.

A variety of other development tools have been created and made available for free or for sale by other companies. A quick search of Google for "ASP.NET" and "IDE" is probably the fastest way to get an up-to-date list.

Because I don't know what development environment you're using — if any — I don't make any assumptions in this book. You can easily use Notepad or any other editor to enter, save, and test the examples throughout this book.

In the end, though, you don't need lots of complex tools and mystical icons to make ASP.NET work. ASP.NET is a simple language combined with standard HTML. What could be more elegant?

Everything You Need to Get Started

ASP.NET isn't a software package that you can just pick up at your local software store. ASP.NET is a *technology*, which is a fancy way of saying it's a cool feature that's built into some *other* piece of software. That other piece of software, in this case, is the .NET Framework. The .NET Framework is a free Microsoft product. It's the foundation for a new approach to software development, and ASP.NET is a key component of that. For more information on the .NET Framework, see Chapter 2.

To set up a machine for creating and testing ASP.NET 2.0 pages, you need three things:

- ✔ **An operating system that supports Internet Information Server (IIS):** For example, Windows 2003 Server, Windows XP Professional, or Windows 2000 Server/Professional.
- ✔ **IIS:** Windows 2003 Server and Windows 2000 Server install IIS automatically, by default. Windows XP and Windows 2000 Professional might not install IIS by default, but you can install it from the Windows installation discs.
- ✔ **The .NET Framework 2.0 Software Development Kit (SDK):** The .NET Framework 2.0 is available from Microsoft at no charge. You can download it at <http://msdn.microsoft.com/netframework>. You can also find instructions there telling you all you need to know about downloading and installing the .NET Framework 2.0.



The hosting option

If you don't want to go through the hassle of setting up your own Web server, you might want to find a Web site hosting service that supports ASP.NET 2.0.

Hosting services are companies that provide room on their Web server for you to place your site. You can even get a domain name (like `www.mywebsite.com`) that jumps right to your hosted site.

However, hosting services aren't all created equal. So, be sure to ask about everything that's important to you:

✔ **Does it host on Windows 2003/2000 machines?** Many hosting services use some form of UNIX operating system and, at least for now, ASP.NET 2.0 doesn't run on UNIX.

✔ **Does it support ASP.NET 2.0 development?** If the service says it supports ASP, it might be referring to Classic ASP or ASP.NET 1.x. Be sure to ask specifically about ASP.NET 2.0.

✔ **Does it provide Microsoft Access or SQL Server database support?** Most hosts allow Access databases, since there's no real setup necessary for the hosting service. However, most charge a setup fee and a monthly fee for the use of a SQL Server database.

A hosting service might charge a setup fee and then usually a monthly fee of anywhere from \$10 up to hundreds of dollars, depending on what you want. Most services offer various packages at increasing price/feature levels. In some cases, you can add features *à la carte* for an added setup fee and monthly rate.

Lots of hosting services are out there, and shopping around can pay off. In fact, if you just want a simple ASP.NET site to practice on, you might find a few that offer minimal space (5MB or 10MB) for free!

For a list of ASP.NET 2.0 hosting options, go to www.aspdotnetfordummies.com/hosting.

You don't need an active Internet connection to create and test ASP.NET pages. IIS can also process and serve Web pages to you locally on the server itself or over a local area network (LAN).

After you have the operating system, IIS, and the .NET Framework installed, you're ready to start developing your own ASP.NET pages!



Visual Web Developer Express simplifies your "what you need" list. You can use any Windows XP, Windows 2003, or Windows 2000 operating system. You don't need IIS installed because Visual Web Developer Express has its own Web server for testing built right in. Just install Visual Web Developer Express, and you're ready to get started!

Watch Your Language!

Before you begin your ASP.NET journey, you need to decide which programming language you want to use for creating ASP.NET pages. Microsoft provides Visual Basic 2005, C#, C++ and others. Third-party companies have developed even more languages for .NET.

I've chosen to use Visual Basic 2005 for this book. It's easy to learn, easy to use and it will get you up to speed as quickly as possible. It's also the most popular computer language ever created!

Understanding the Development and Testing Process

In this section I cover a variety of topics that you need to understand before you begin actually creating ASP.NET pages. Among them: choosing the Web server you want to work with, understanding where IIS looks for your pages, where and how to save your pages and how testing works.

Choosing an ASP.NET Web server

Figure out what server you will use to test the pages you create. You might use an existing server at your company, set up your own server, or use a hosting service.

If you use an existing server at your company, there are probably procedures in place regarding how you can add new pages, where you can and cannot put test pages, and so on. Be sure you fully understand and follow those guidelines.

If you set up your own system at work or at home, you're responsible for installing everything and getting it up and running. You're in complete control, but it can be a big task, requiring knowledge of server operating systems and IIS.

Perhaps the easiest solution is to use a Web hosting service that supports ASP.NET. The host gives you the information you need to know to create new folders on your site, send files you create to the site, and test those files. And best of all, the hosting service handles all the installation and maintenance for you! For more information on this option, see the sidebar titled "The hosting option," in this chapter.

Understanding where IIS looks for your pages

If you're working directly with the Web server (*not* through a hosting service), you need to know a little about IIS and where it looks for its pages.

IIS creates a folder on the server's hard drive with the default name `inetpub`. The `inetpub` folder contains a subfolder called `wwwroot`. The `wwwroot` folder is the root for the Web site.

If you place a new Web page in `wwwroot` or any of its subfolders, your Web server has access to the page, and you can link to it from other pages on your site.

So when you begin a new project, create a folder under `wwwroot` for your application, and then, within that folder, create the pages you need. If you have lots of pages, you might want to create subfolders inside your application's folder to help organize things.



The process for creating, saving, and testing pages in Visual Web Developer Express is a little different from doing it in Notepad or a simple editor (and quite a bit simpler). If you like, you might want to jump ahead to Chapter 4, which focuses on Visual Web Developer Express.

Saving your page with the .aspx extension

All ASP.NET page filenames must end with the `.aspx` extension. That's how the Web server knows it should process the page as an ASP.NET page.



If you use Notepad as your editor to create ASP.NET files, be aware that Notepad almost *insists* that you name your files with a `.txt` extension. In fact, if you try to create a new file and save it as `test.aspx`, the file's name ends up `test.aspx.txt`. (Notepad pulls this trick only if you're creating new pages. When you edit and save an existing page, Notepad always saves the page with the extension it had originally.) Here's how you fix that problem: When you type in the filename, put it in quotes — for example, enter `"test.aspx"`. Notepad saves the file with exactly the filename you give it.

If you accidentally save a file with the wrong extension, just go into Windows Explorer and rename the file.



Many versions of Windows, by default, hide known extensions from you. This can be a problem if Notepad saves your file as `test.aspx.txt` because the file might show up in Explorer as `test.aspx` (hiding the `.txt` extension). It looks right, but the `.txt` extension prevents your Web server from recognizing it as an ASP.NET page. Of course, this can be very confusing. To show extensions in Windows Explorer, choose **Tools**⇨**Folder Options**. Then click the **View** tab and scroll down until you find the option labeled **Hide File Extensions for Known File Types**. Make sure that check box is *not* selected.

Testing your pages

To test your ASP.NET pages, open a browser window, type the address of the page into the Address bar, and press **Enter**. That makes the browser request the page from the server and gives the server the opportunity to process the ASP.NET code on the page.



If your previous experience is primarily with HTML pages and JavaScript code, you may need to break some old habits when you create and test your ASP.NET pages. In the past, when testing pages, you might have opened the page by simply dragging and dropping it onto the Internet Explorer browser. Or you might have used the browser's **File**⇨**Open** command to open the page. You can't use either of those techniques for testing ASP.NET pages. Both techniques open up the page directly without any Web server involvement, so the server has no opportunity to process the server-side code in the page and nothing happens! Because ASP.NET works on the server side, you have to let the server find the page, run the code, and then send the results to the browser.

Getting Your ASP in Gear: Creating an ASP.NET Page



When you have your editor or IDE selected and a Web server where you'll test your pages, you can create your first ASP.NET page:

- 1. Create a folder under wwwroot and give it a name.**

For my examples, I use a folder by the name of `hello`.

If you're using a hosting service, you don't see `wwwroot`. Just create a folder in your root directory. Again, your hosting service can tell you how to do this.

- 2. Create a new page in Notepad or the editor of your choice.**

Most HTML editors work fine, as long as they let you work with the raw HTML tags and don't try to hide them from you. If you're using Visual

Web Developer Express, click the Source tab. If you're using Microsoft FrontPage, click the HTML tab. If your editor puts any tags or other stuff in the page automatically, delete all that so you start with a blank slate.

3. Enter the code in Listing 1-1.

Listing 1-1: The Hello and Welcome Page

```
<%@ Page Language="VB" Debug="True" %>
<html>
<head>
<title>Hello and Welcome Page</title>
</head>
<body>
<center>
<% Dim TextSize As Integer %>
<% For TextSize = 1 To 7 %>
<font size = <%=TextSize%>>
Hello and Welcome!<br>
</font>
<% Next %>
</center>
</body>
</html>
```

Carefully type the code as it is listed. Don't worry about whether something is entered in upper- or lowercase. It will work either way.

4. Save the page in the folder you created earlier under wwwroot. Give it a filename with the extension .aspx.

For my examples, I name the file `welcome.aspx`.

If you're using Notepad, put quotes around the name so that Notepad doesn't add the `.txt` extension.

If you're using a hosting service, save the file to a spot on your local hard drive, and then send the file to the server.

5. Open your browser, and then open the page you created by typing `http://localhost/foldername/filename` into the Address bar at the top of the browser. (Fill in *foldername* and *filename* with the folder and filename you gave these items when you created them.)

If you aren't working on the Web server machine that's running your page, access your page by simply typing the intranet URL. Usually, you just type the name of the server in place of `localhost`, as in `http://bigserver/hello/welcome.aspx`.

If your Web site is on a hosting service, use the address the service gave you or your own domain name to access the page, as in `http://www.mydomain.com/hello/welcome.aspx`.

You should see the page you created in the browser. It looks a lot like Figure 1-1.



Figure 1-1:
The
welcome.
aspx page.

If you don't see this page, go back and check to make sure you entered the code exactly as it appears in Step 3.

Understanding How This Example Works

In the preceding section, I show you how to create a Web page so that you can understand the *process* for creating and testing pages. Unless you already know Visual Basic 2005 or are a proficient programmer, you probably don't know exactly how the page works just by looking at it. I explain exactly what the code in Listing 1-1 does here.

The programming commands always appear inside `<%` and `%>` symbols. Those symbols are called *delimiters*. They keep the commands separate from the HTML tags.

The very first line in Listing 1-1 is a page header:

```
<%@ Page Language="VB" Debug="True" %>
```

A page header isn't required, but including it at the top of any ASP.NET page is a good idea. It does two things: It specifies that the ASP.NET language used in this page is Visual Basic 2005, and it sets `Debug` to `True`. The `Debug` setting helps by providing more detailed error messages for you if you make a mistake.

The first Visual Basic 2005 command on the `welcome.aspx` page is `Dim`. You use the `Dim` command to create variables. Here, it creates a variable named `TextSize`:

```
<% Dim TextSize As Integer %>
```

A *variable* is a place to store information to be used later. The `As Integer` clause at the end of that line indicates that this variable will hold whole numbers. No information is put into the variable, yet. (For more information on using `DIM` to declare variables, see Chapter 5.)

The next line identifies the beginning of a `For . . . Next` loop. A *loop* provides a way to repeat the same commands (or HTML) over and over again. This loop repeats the lines between the `For` line and the `Next` line. These lines (shown here in bold) are called the *body* of the loop:

```
<% For TextSize = 1 To 7 %>  
<font size = <%=TextSize%>>  
Hello and Welcome!<br>  
</font>  
<% Next %>
```

When does a loop stop looping? Well, different kinds of loops exist, but a `For` loop decides when it's done by counting. In this case, it counts from 1 to 7. The loop sets the value of the `TextSize` variable to 1 the first time through the loop. Then, the body of the loop is executed. The `Next` statement identifies the end of the loop, so the code jumps back up to the top again. The second time through the loop, `TextSize` is set to 2; the third time, it is set to 3; and so on through 7, when the loop ends. (For more information on the `For` loop, see Chapter 6.)

In the body of the loop, the `` tag sets the size of the text. The size is set equal to `<%=TextSize%>`. What's that? The `<%` and `%>` are the normal delimiters, but there are no commands inside! Just an `=` sign followed by the `TextSize` variable name. This special syntax gets at the *value* of the variable. So the first time through the loop, the HTML `` tag that's generated looks like this: ``. The second time through, it looks like this: ``. Each time through the loop, the font size is set to the value of `TextSize`.

Then, `Hello and Welcome!` is displayed on the page. Because the font size is set to a bigger number each time the loop executes, the same line is printed again and again in increasingly larger sizes. And that's exactly what you see in Figure 1-1, shown earlier.

If you choose `View` → `Source` from the Internet Explorer menu while looking at this page, you see the following HTML:

```
<html>
<head><title>Hello and Welcome Page</title>
</head>
<body>
<center>
<font size=1>
Hello and Welcome!<br>
</font>
<font size=2>
Hello and Welcome!<br>
</font>
. . .
<font size=7>
Hello and Welcome!<br>
</font>
</center>
</body>
</html>
```

As you can see, there's no indication of a loop in the HTML that was sent. In fact, you don't see any programming code at all — just pure HTML. One of the beauties of ASP.NET is that the code is executed on the server and *produces* the HTML that is sent to the browser.

Don't feel like you need to completely understand this example before you go on. You get a better understanding of ASP.NET itself in Chapter 2, and I cover Visual Basic 2005 commands in Chapters 5, 6, and 7. For now, just make sure you understand the process for creating and testing an ASP.NET page.

Modifying, Retesting, and Creating Pages

The information in this chapter and the steps described in the previous section give you the information you need to create and run the examples in this book. However, if you make a mistake, you'll need to know how you can make a change to a page and then retest it to see if you fixed the problem. In addition, there are a few other common tasks that you'll need to understand as you begin creating your own ASP.NET pages. In this section I'll provide some steps you can use to accomplish these goals.

Modifying and retesting pages

If you want to modify a page and then test it again, follow these steps:

1. **In your preferred editor, open the page file and make your changes.**

2. **Save the file. Save the file. Don't forget to save the file.**
3. **Go to your browser window.**
4. **Click Refresh in the browser window.**

The page is requested from the server and executed. The results are then displayed in your browser. If they still aren't what you expected, just lather, rinse and repeat.

Creating new pages

Whenever you want to create a new ASP.NET page, follow these steps:



1. **Create the page in your editor and save it with the `.aspx` extension to the appropriate folder.**

Put quotes around the name if you're saving from Notepad!

2. **Type the page's address into the browser's Address bar and then press Enter.**

The server retrieves and processes the page, and the results are sent to your browser.

Converting HTML pages to ASP.NET pages

Later, when you're ready to start working on your own Web site, you'll probably want to convert some of your existing HTML pages into ASP.NET pages. That's easy:

1. **Find the file on your server's hard drive by using Windows Explorer.**
2. **Rename the file. Change the `.htm` or `.html` extension to `.aspx`. (Or you could copy the file and give the copy the `.aspx` extension. This allows you to keep the old HTML file, if you like.)**
3. **Edit the page in your editor and add any Visual Basic 2005 commands you like, and then save the page.**



The only difference between normal HTML files and ASP.NET files is the extension. The extension tells the Web server how to process the pages. If you find that your code isn't executing, make sure you set the extension to `.aspx`!

