1

An Introduction to ASP.NET 2.0 and the Wrox United Application

At the end of the twentieth century something unprecedented happened to computers. Previously relegated to the realm of the bedroom and dedicated hobbyists who never saw the light of day, the explosion of the Internet lead to computers acquiring a glamour, an aura of excitement that had never been associated with them before. Prior to the 1990s it was almost embarrassing to admit you worked with computers, and then suddenly everyone wanted one. Every business had to be attached to the Internet, and many families wanted their own web site. If you had to name one piece of technology that became synonymous with the explosion, it was undoubtedly the web browser. However, without anything to view on a web browser, it becomes virtually useless. You need information, and like mushrooms sprouting up in a woodland, hundreds of web sites on every imaginable subject were born.

The late '90s were a time of vast upheaval. Business empires were founded on the simplest ideas — a search engine (Google) or an online store for buying books (Amazon). Everyone wanted to know how to build a web site for themselves. HTML (HyperText Markup Language) enabled them to do that, but it was soon obvious that it only went so far. You could display pictures and text, but what happened if you wanted more than that? What happened if you wanted a site that was reactive, that received information from your users and was automatically updated without someone having to beaver away writing new web pages every time? What if you wanted to attach a database to the Internet, or you wanted to display a stock catalogue, or you wanted to personalize your site to everyone who visited it, or you just wanted it to look good for your family and friends who visited it?

The race was on and several competing technologies were created for doing this, including CGI, PHP, and Java. Microsoft's own entry into the race was ASP and what made it particularly attractive was that it was simpler to pick up and learn than most of its rivals, but it also had some exciting features — the ability to store details of users as they moved through pages on a web site, and controls such as calendars and ad rotators that you could just stick into your pages like HTML tags. ASP was a huge success. Microsoft went one further; it created the .NET Framework, and

ASP.NET became a "grown up" version of its ASP technology, using its mature programming languages VB.NET and C#. The leap forward in power was amazing, but Microsoft lost partial sight of one critical aim — simplicity. Web sites suddenly became things you needed expensive consultants to build and cutting-edge designers to visualize. It was out of the hands of those who so empowered the boom.

ASP.NET 2.0 is the big step back in this right direction. Microsoft recognized that one thing people who build web sites don't want to do is have to code. Code is dull; code is geeky. However, Microsoft also recognized that some people still have to code for a living. And more than that, these coders have to build the same things, over and over again. A login mechanism, a menu system, a shopping cart, a funky theme for your site's backdrop applied to every page — something every web site requires. Two guiding principles seem to be at work here: make it easier for the novice to use and reduce the amount of repetitive work the developer has to do. Claims for ASP.NET 2.0 boast "70% less code" is needed; ASP.NET 2.0 also comes with a multitude of controls to enable the developer to create login systems and menus in minutes.

Late in 2003 we saw the previews of the new version of Active Server Pages named ASP.NET 2.0. Everyone knew that these claims weren't just hyperbole and that the way developers create web applications was going to change fundamentally. Microsoft expanded the powerful features of earlier ASP versions while greatly reducing the effort to implement those features. The ease of implementation meant a reduction in the cost of developing complex sites. Or, put another way, there would now be a large expansion of the number of people that have the capability to build a complex site.

In addition to ASP.NET 2.0 comes a new, affordable tool for creating these web sites: Visual Web Developer Express. Microsoft's previous attempts at tools for helping create web sites have been clunky (Front Page) or have never really taken off (Visual Interdev), but this time they've got it right. Visual Web Developer will be part of the Visual Studio.NET suite, but a limited version named Visual Web Developer Express will be sold inexpensively (or given away). It'll allow you to drag and drop a site together within minutes, but it'll also be instantly recognizable to developers, and allows easy creation and management of your web pages.

This book leads you step-by-step through creating dynamic, data-driven, complex web sites using ASP.NET 2.0. To those ends this chapter explains the basic ideas and examines the completed sample site. You then spend time learning how to use Visual Web Developer Express (VWD) to build ASP.NET 2.0 sites.

Specifically, this chapter covers five topics:

- An introduction to ASP.NET 2.0
- Review of the Internet programming problems that ASP.NET 2.0 solves
- An explanation of how ASP.NET 2.0 fits in with other technologies
- A tour of the dynamic features of a site built with ASP.NET 2.0
- Understanding the tool you will use to build ASP.NET 2.0 (ASPX) pages — Visual Web Developer Express (VWD)

In previous books we've been pleased if our readers can create a single page by the end of the chapter, but ASP.NET 2.0 inspires much greater ambitions, and you will have the structure and outline of a working web site up by the end of the second chapter. Your Web site will be focused around a hapless soccer

(football) team named Wrox United and will be able to display their news and results, sell their merchandise, screen their footage, and will offer different views of the site depending on whether you are a customer or an administrator. And, as always, a list of gotchas and some exercises are included to help you review the concepts covered in this chapter.

The Site You Will Build

Go to `www.wroxunited.net` and have a good look at the site (the main page is shown in Figure 1-1). This site is built entirely in ASP.NET 2.0 and is the site you will build in the book. Likewise, it is the site that you will learn how to create a working miniature of in just two chapters.

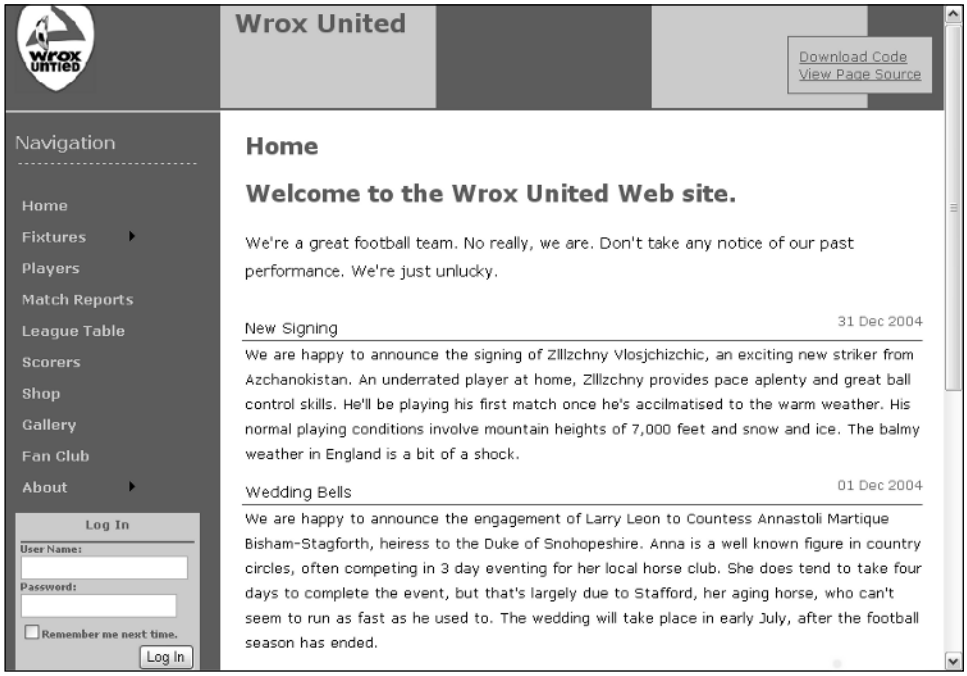

Figure 1-1

On the home page alone you can see a menu system, a login control, and some news items — these are all things that would have taken considerable time and code to create in any previous version of ASP or ASP.NET. If you take the example of a login mechanism, you'd have to think of accepting a user ID and password, checking that against an existing set of users and passwords, making sure the password wasn't corrupted in any way, and making sure that password was transmitted securely. So just to do something relatively trivial, you'd be talking an hour or two of your time at least and not much to show for it. Now this could take seconds.

Click the View Page Source link — it doesn't matter if you don't understand what you see yet — there are fewer than 10 lines of ASP.NET 2.0 code:

```
<%@ Page Language="VB" Trace="false" MasterPageFile="~/site.master"
AutoEventWireup="false" codefile="Default.aspx.vb" Inherits="_Default" %>
<%@ Register TagPrefix="wu" TagName="News" Src="News.ascx" %>

<asp:Content ID="Content1" ContentPlaceHolderID="mainContent" Runat="server">

    <h2>Welcome to the Wrox United Web site.</h2>
    <p>We're a great football team. No really, we are. Don't take any notice
    of our past performance. We're just unlucky.</p>

    <wu:news id="News1" runat="server" ItemsToShow="5"></wu:news>

</asp:Content>
```

Step through the different links in the menu and see how league tables and fixture lists work, and see how few lines of code there are. Notice how the theme and style of the site remains consistent throughout, yet there is no evidence of how this is done. Welcome to ASP.NET 2.0. This is about to revolutionize how you build web sites from now on. You're going to look at some of the features behind the Wrox United site in more detail shortly, but first let's talk about what ASP.NET 2.0 offers.

ASP.NET 2.0 — A Powerful Tool to Build Dynamic Web Sites

The World Wide Web (WWW) on the Internet provides a wide expanse of connectivity. Virtually everyone that uses computers has access to the Net. But this pervasive reach was achieved by establishing very minimal standards. Information is transmitted in ASCII characters, without a built-in capability for machine-level code. The client requirements are very minimal — in fact the Internet itself does not have any standards for how a browser works, and thus multiple browsers for multiple operating systems (OS) and platforms exist. It is easy for us, in 2005, to forget that the Internet was designed to send simple static pages of text with images and links.

The story of the past fifteen years of Internet programming is an effort to provide sophistication and complexity to the user experience while not violating the WWW rules that demand extreme simplicity in page design. Users expect an experience that comes close to desktop applications such as word processing and database access. But such a level of complexity has not been easy to implement in the Web given its minimal configuration.

ASP.NET 2.0 fundamentally reduces the barriers for development of complex web sites. The ASP.NET development team at Microsoft looked at thousands of pages, sites, and scenarios to create a list of common objectives of site owners. The list included about twenty goals, including reading data, a unified login and authentication procedure, consistency in site appearance, and customization of pages for different browser platforms. The team then set to work to create bundles of code that would achieve each objective in the right way, with a minimum of developer effort and with Microsoft performing extensive tests of that code. This set of capabilities is available as *classes* (encapsulated and ready-to-use batches of code) in ASP.NET 2.0. The end result is simple — developers can very quickly put together (and easily maintain) a complex site by merely assembling the building blocks Microsoft has developed in ASP.NET 2.0. Instead of writing 50 or so lines of code (as in earlier versions of ASP), the designer can now simply

drag and drop a control to the page and answer some questions in a wizard. This control generates a small amount of code for your page and the server uses that code to build pages in HTML that are then sent to the browser. Because HTML is sent to the browser, there is no requirement for special capability on the browser beyond display of HTML and the execution of a single simple JavaScript script. Any browser that can display HTML can display ASP.NET 2.0 pages. This includes not only desktop browsers, but also PDAs, cell phones, and other devices.

Because all the code for these controls is run on the server before a web browser ever gets a hold of a page, these controls are known as *server-side controls*. The next section looks at what some of these server-side controls can do.

Simple Solutions for Common Web Site Tasks

Microsoft's survey of sites in earlier versions of ASP created a list of common objectives that site programmers were implementing. Some objectives were easy to achieve but time-consuming, whereas others were too complex for all but the most sophisticated developers. Overall, the programmers' solutions varied from brilliant to dysfunctional. Not only were the observed solutions sometimes poor, but they also represented a tremendous waste of time, because thousands of programmers spent time planning, writing, and testing code that had the same purpose. This section goes through eleven of the objectives for which ASP.NET 2.0 offers built-in solutions. As you will see in Chapter 3 and beyond, these solutions are in the form of ASP.NET 2.0 server-side controls that contain code to execute settings and behaviors. By simply placing one of these controls on a page, the designer gets all of the behavior that would have been hand-coded in the past.

Consistency and Personalization

Web designers tend to desire two conflicting design features. On the one hand they want a consistent look to the entire site. But conversely, they desire the capability of users to customize the site to the user's taste in colors, font size, and other features. ASP.NET 2.0 offers a `MasterPage` control that allows a site to be consistent in the layout of its headers, menus, and links. Within that consistent layout a designer can add a control that allows users to pick one of several themes to apply to all pages.

Navigation

Every site requires tools for navigation, generally in three forms. Users need a menu. They also need to be able to see where they are currently located in a site. And last, they want to be able to easily navigate up or down a level. ASP.NET 2.0 supports an XML file called a SiteMap. ASP.NET 2.0 controls can then render menus and other navigation aids based on the site map and the name of the current page.

Login, Security, and Roles

Many sites need a login system that can check a potential user's ID and password against a list and then authorize or deny entry. Although basic implementations are not difficult, only a small percentage of programmers are successful in creating a system that conforms to best security practices. ASP.NET 2.0

offers a few controls that create and implement a logon better than most of us can program by hand. Beyond simple site entry, the system offers password reminders and a system to create new users. A user can also be assigned a *role* that determines what pages and features will be available for that user to view. For example, all users can view the employee phonebook, but only users with the role of Manager can view pages to change information about employees.

Connection to Data

Although most dynamic web sites are connected to data, few designers are successful in implementing the full suite of features that users desire. In sites built with older ASP versions Microsoft observed many problems in efficiency and security. Furthermore, even modest objectives required scores of lines of code. ASP.NET 2.0 provides a rich suite of data features through two groups of controls for working with data. *Data source* controls offer the behavior of connecting to sources of data. *Data-bound* controls take that information and render it into HTML. The several data source controls can connect to almost any source of data, and the data-bound controls offer the user tables, lists, trees, and other presentations. Working together, these controls offer the user the capability to page through data, as well as to sort and edit data.

Code

Almost every web site requires customized code because it is impossible for ASP or any other web site technology to anticipate the needs of all businesses. ASP.NET 2.0 supports more than 20 different languages. Regardless of the language the programmer uses, the code is translated into a single intermediate language prior to execution. ASP.NET 2.0 controls are executed on the server, but the programmer also has the option of writing code (for example Java or other client-side script) in a block to go out for execution on the client.

Componentization

Web sites are easier to develop and maintain if various parts can be created independently of each other. Traditional ASP relied on large pages containing content, HTML, ASP controls, and scripts of code. ASP.NET 2.0 provides more efficient models and structures to divide the site into logical parts. For example, code is normally kept in separate files from the presentation layer (the text and HTML tags). Furthermore, Visual Web Developer offers wizards to easily create objects to provide data resources.

Web Services

Enterprises offer information and services on their own sites. For example, from its worldwide headquarters, `www.Ford.com` can give you a list of colors and price quotes. But in the past few years we have seen a demand for those services to be available to other sites. For example, a local Ford dealer may want to offer the list of colors and prices at `www.YourLocalFordDealer.com`. Web Services allow a *consumer* site (the local dealer) to obtain information from a *provider* site (Ford headquarters). The local Ford dealer can display real-time data using Web Services provided by the Ford corporate site, but keep the user on the page of the local site. ASP.NET 2.0 offers a complete Web Services solution that conforms to the specifications of SOAP (Simple Object Access Protocol, a way to ask for data from a Web Service) and XML (Extensible Markup Language, a format for data).

Performance and Caching

After the designer writes a page on the development machine it is compiled into the single uniform language of .NET 2.0 called the Microsoft Intermediate Language (MSIL). Then it is copied to the deployment machine. The first time it is requested the page undergoes a further compilation into the Common Language Runtime (CLR), which optimizes the page for the hardware that will serve it. This two-step process achieves the dual aims of consistency for software and optimization for hardware. Both steps have undergone intensive performance analysis from the .NET 2.0 team. The great aspect for beginners is that all of this compilation occurs automatically.

ASP.NET 2.0 easily enables *caching* of pages so that subsequent requests are served faster. When cached, the final version of a page is stored in the server's RAM so that it can be immediately sent on the next request rather than having the server rebuild the page. Furthermore, the designer can specify that only parts of pages can be cached, a process known as *fragment caching*. Fragment caching accelerates the service of non-changing portions of a page while still allowing the dynamic fragments to be custom generated. If you are using Microsoft SQL Server 7 or higher you also have the option of *data invalidation caching* for portions of the page that are data-dependent but less variable (perhaps a list of employees or your retail outlets). Data invalidation caching will keep a page in cache until it gets a message from SQL Server that the underlying data has changed. You cache a set of data with a designation to receive a SQL data changed notice. SQL Server will notify .NET when the data has changed, which triggers ASP.NET 2.0 to perform a reread.

Errors and Exception Handling

Any web site, indeed any system, needs to respond to errors. ASP.NET 2.0 provides a system to respond to errors. The response can be in code or it can be a redirect to an error page. The error page can be unique to the error or it can be a single error page for the entire site. The .NET 2.0 Framework also allows multiple levels error handling. If an error occurs in a data read it can be handled at the level of the data source. If it is not handled, the error bubbles up to the next level and can be handled there. Unhandled errors continue to bubble up through layers with the designer having the option to resolve the problem at the most effective level.

Deployment

In the past, sites deployed to Windows or Linux required a series of setup steps that registered and configured the site on the host machine. The ASP.NET 2.0 team set a goal of XCOPY deployment, naming it after an old DOS command that performed a simple copy of a folder and all of its subfolders. A simple XCOPY deploys your site from the development machine to the deployment host. All registrations and machine-level customizations occur automatically when the first request hits the site.

Development Tools

Although not part of ASP.NET 2.0, Microsoft has spent considerable effort improving tools for building ASP pages, namely the Visual Studio, Visual Web Developer, and Visual Web Developer Express products. These IDEs (Integrated Development Environments) allow drag-and-drop building of pages. Most common actions are either automatic or guided with wizards. In cases where typing is required, the IDE provides intelligent completion of most phrases. This book uses the freely downloadable VWD Express.

Where Does ASP.NET 2.0 Fit with Other Technology?

Many people have questions about how ASP.NET 2.0 fits in with all of the other web-related terms (most of them acronyms). We will clarify this now — where does ASP.NET 2.0 fit with other software that is running on the server? What is its role, and what are the roles of the other pieces of technology? ASP.NET 2.0 is part of the .NET 2.0 Framework. The .NET Framework is a brand of Microsoft that sets software standards for Internet connectivity using Web Services and XML. Many Microsoft products conform to the .NET standard, including various server software, data management systems, and desktop applications. ASP.NET 2.0 is the subset of .NET 2.0 that offers software for generating dynamic web sites. The software is published in a set of *classes* holding various controls that execute their behavior on the web server. In our day-to-day designing of pages we work with these server-side controls. Because ASP.NET 2.0 is a subset of the .NET 2.0 Framework, in this book we sometimes uses features of ASP.NET 2.0 and sometimes uses features of the .NET 2.0 Framework. Use of these various features will be essentially seamless.

As a Microsoft product, ASP.NET 2.0 runs on Windows. For development, it will work on the desktop with Windows 2000 or later (including both XP Home and XP Pro). At deployment the normal OS is Windows Server 2003 or another Windows OS version designed for higher loads. Within Windows, ASP.NET 2.0 works with the Internet Information Server to read pages from disk and send them to requestors. Alternatively, on the development desktop, ASP.NET 2.0 can be tested with a lightweight web server named Cassini that is distributed with development tools such as VWD.

When a designer uses the ASP.NET 2.0 controls to connect with data, two more levels of interaction are introduced. The data controls use a technology named ActiveX Data Objects (ADO.NET), but fortunately the use of ADO.NET is behind the scenes for us. Those ADO.NET objects, in turn, interact with the source of data. The source of data can be Microsoft SQL Server (as used in this book) or almost any other source of data including relational databases such as Oracle or MySQL, and non-relational sources such as XML or text files.

Microsoft offers tools for several levels of developers to build ASP.NET 2.0 web sites. The most comprehensive product is Visual Studio 2005, aimed at professional developers. A low-cost, and only slightly less function version, is Visual Web Developer Express. Front Page can work, but it focuses more on static HTML pages and thus lacks the set of tools that makes designing the dynamic, data-intensive ASP.NET 2.0 pages such a pleasure. Creating pages in Notepad was long the preferred method of ASP developers and is still theoretically possible; however, the necessary management of web sites and web pages make this impractical, laborious, and far more prone to errors.

Enough of the theory; let's see ASP.NET 2.0 in action. During the course of this book you will build a complete web site for a hapless football (soccer) team named Wrox United. A completed example is hosted at `www.wroxunited.net`, which you explore in the next section to observe the range of features ASP.NET 2.0 supports. Then in the remainder of the book you will build the same site on your desktop. For this exercise you do not have to install software on your machine. The remainder of the book, however, relies on your completion of the setup outlined in Appendix B.

Exploring the Wrox United Application

This section explores the site as built by the authors and that you will build, and which is hosted at `www.wroxunited.net`. Open your browser and direct it to that address.

- **MasterPages and Site map (discussed in Chapter 2):** Click through several pages to observe the uniform layout across the top and left side of the page. This design consistency derives from an easy-to-implement feature called MasterPages. Second, note the maroon box in the lower-right of each page that indicates your current page and its relationship to parent pages back to the home page. This feature was created with the ASP.NET 2.0 Site Map and Navigation controls.
- **Server-Side controls (discussed in Chapter 3):** Go to the Players page. All of the data comes from two server-side controls — a data source control to connect to the database and a data-bound control to display the information. Most of the behavior of ASP.NET 2.0 pages is encapsulated in server-side controls. These include links like the shopping cart at the bottom-left, images such as the logo at top-left, and text boxes such as the logon section at the lower-left.
- **Login and Security system (discussed in Chapter 4):** On the home page, log in as User Name `Lou` and Password `lou@123`. Then log out. Authentication systems can require a tremendous amount of work to create and even then frequently contain security holes. ASP.NET 2.0 offers a very simple system based on several server-side controls including the login and password verification schemes, and a system to e-mail a clue for forgotten passwords.
- **Events (discussed in Chapter 6):** Browse to the Shop page, click an item and add it to your cart (of course, this is not a real shopping site, just a demo). An event occurred as you clicked the Add to Cart button and that event was handled by custom code that created an order and added the item to the order.
- **Data Reads (discussed in Chapter 7):** Browse to the Players page where the names and joining dates are read from a SQL Server Express database. Many kinds of information on the site are held in data stores that are read by ASP.NET 2.0 server-side controls on the page. Browse back to the home page and observe the menu. Even these menu choices are read from an XML file that holds a map of the site.
- **Data Writes (discussed in Chapter 8):** Browse to Shop, click car sticker and click Add to Cart. You have just written a value to a database. The behavior of writing your order to the database was implemented by two ASP.NET 2.0 server-side controls. The designer of the site did not have to do any custom code.
- **Code behind the controls (discussed in Chapter 9):** From the home page click Shopping Cart at the lower-left of the page. We have written custom code that executes when the page is loaded that checks if there are currently any items in the shopping cart and renders a page appropriate for the cart contents: either empty or a list of contents. Although the capabilities of the ASP.NET 2.0 server-side controls are impressive, they cannot cover every possible case. An ASP.NET 2.0 site offers numerous places that a designer can add custom code.
- **Components (discussed in Chapter 10):** Browse to the Fixtures page. Although the data is stored in a SQL Server database, the ASP.NET 2.0 page does not read it directly. There is a component that reads that data and then sends it on to the ASP.NET 2.0 page. That component can be reused by other web sites or by Windows applications that run on a local network without the Internet.

- **Roles (discussed in Chapter 11):** If you had administrative rights you could log in and see different screens. Once you have installed the site on your local machine you will experiment with this feature in Chapter 4. ASP.NET 2.0 goes beyond just logging in visitors. They can be authorized to have sets of privileges called *roles*. The public site does not allow non-authors to log in as administrators, so no need to take action at this point.
- **E-Commerce (discussed in Chapter 13):** From the menu go to Shop, and click a few items to add to your cart. Now on the bottom of the menu click Shopping Cart and view its contents. The most complex part of the site is the shopping cart. ASP.NET 2.0 does not have a pre-built e-commerce solution, but because so much behavior is built into the ASP.NET 2.0 controls, designers can develop features such as e-commerce much more quickly than in the past.
- **Performance (discussed in Chapter 14)**
- **Errors and exception handling (discussed in Chapter 15)**
- **Deployment (discussed in Chapter 16):** At this point we will not walk through a deployment. However, keep in mind that for ASP.NET 2.0 the transfer for a site from a development machine to a deployment machine is generally only a few steps that copy the databases to the data server and then do a simple file copy of the site folder and its subfolders to the new server.

This walk-through gave you a taste of what you will learn to create in this book. Most of the features explored were implemented with very little code that we wrote. The behavior was performed by code that Microsoft baked into a set of server-side controls that are the components of ASP.NET 2.0. We merely placed these controls on pages and set various properties.

Getting Started with Your Wrox United Site

Having observed the finished site as publicly hosted, now is the time to begin creating the same site on your desktop. If you have not installed Visual Web Developer Express, SQL Express, the sample database, and the sample site (outlined in Appendix B) then do so now. Start by reading the overview at the beginning of the appendix and then work your way through each step. You can be sure of your installation by performing the check at the end of each section.

This chapter and the next chapter set up the basic framework of the site as you learn how to use VWD and establish some design parameters for the site. Because VWD offers drag-and-drop solutions to most tasks, you will be able to create the entire site with a minimum of typing. In the cases where some typing is necessary you can cut and paste from text files in this book's download at `www.wrox.com`. All pages are in the download in their final form, but we strongly believe that creating ASPX pages yourself is a better way to learn ASP than merely copying completed pages from our reference set.

VWD Express — A Development Environment

A fundamental difference between most animals and humans is the ability to use tools. In the early days of programming the tools to write programs were very primitive. Today we enjoy the benefits of very sophisticated tools for software development. Engineers have taken almost every area of human weakness (primarily related to the capacity of memory and the brain's interface to the world) and created

compensating tools. These tools are pulled together into a type of software called an Integrated Development Environment (IDE). The IDE used in this text is Visual Web Developer Express (VWD).

VWD contains a number of development tools. First is an editor in which you can build a web page. This editor is enhanced with IntelliSense, a tool that finishes typing commands and offers appropriate choices for the developer. In addition, a toolbar contains icons that can be dragged to the editor and will automatically type a block of code into the editor. Another way to automatically get code into a page is with the many wizards that pop up when attempting a more complex task. VWD also contains a mini File Manager to organize ASPX and associated files and folders. Similarly, there is a Data Explorer that offers navigation through the data sources of the web site. A suite of troubleshooting tools is also included. Finally, VWD ships from Microsoft with a web server for testing named Cassini, which is covered in the next section. If you go beyond the scope of this book you can discover tools for more complex scenarios including the management of code versions among a team of developers.

Introducing Cassini

Earlier, Cassini was mentioned as the lightweight web server that comes with VWD. Both Cassini and IIS (included with the .NET Framework) can serve all ASPX and associated pages, so at deployment there is no need to make changes to your site. But a number of differences exist between the servers.

The two servers use different security models. IIS is a service and every service in Windows requires a user. The special user for IIS is named ASPNET. Cassini runs as an application that uses the currently logged-in Windows user. That makes Cassini easier to install because there is no need to create a specific ASPNET account in Windows. In fact, the installation of Cassini is transparent when VWD is installed.

Cassini has three downsides. First, it is a tool for designers to test pages on their development machine and thus it does not scale to more than one user. Second, because of the simplifications to the user model, Cassini cannot support a robust security scheme. Cassini should only run in a closed environment or behind a robust firewall. Third, when you run a page in Cassini it locks the page back in VWD. In order to unlock the page you must close the browser, which can be inconvenient when making and testing many changes to a site. Therefore, many developers use IIS even on their development machines so they do not have to close a page in the browser before working on it in VWD. The downside is that you have to configure your development machine to provide IIS, set up the appropriate authorizations, establish security controls, and create a virtual root. You learn how to set this up in Appendix B. If you don't want to go through the IIS setup you can still use Cassini and just close the browser between modifications.

VWD's Solution Explorer

An ASP.NET 2.0 web site is stored as a family of files. You need to be able to organize these files, including the tasks of viewing their names and relationships, copying, creating, and deleting. They can be viewed and manipulated in Windows Explorer. But it is inconvenient to switch between VWD and Windows Explorer, so VWD includes an explorer-like tool called the Solution Explorer, shown in Figure 1-2. The Solution Explorer is displayed by default on the right of the screen, or you can redisplay it by pressing Ctrl+Alt+L. Think of it as a Windows Explorer that considers your web root to be the highest level and does not require you to switch out of VWD. Note that the Toolbox may be placed behind the Solution Explorer, as in the case of Figure 1-2.

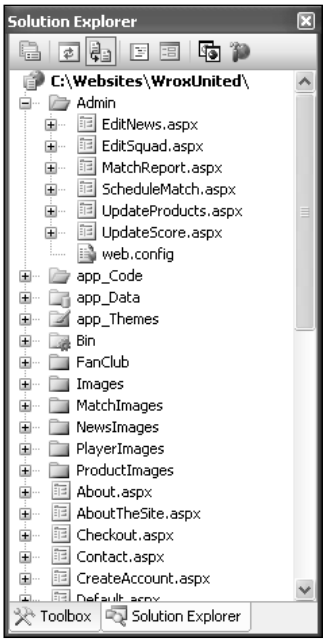

Figure 1-2

The layout of Solution Explorer is instantly familiar to anyone who uses Windows Explorer. Click the plus and minus icons to expand or contract folders. The icons in the toolbar start from the left with a tool that switches from the Solution Explorer to a view of properties (more on that later in this chapter). The double horizontal arrows perform a refresh. The double file icon automatically hides or expands nested sub-files. The next two icons open the selected files to display either their user interface (design) or their code. The double browser icon is used to copy the entire site to the deployment machine. The right-most icon, with the hammer, opens a Web Administrator tool to manage features of the site.

At the bottom of the Solution Explorer may be a small task bar that shows tabs for the Solution Explorer, Toolbox, Data Explorer, and/or Properties windows. To conserve monitor real estate these four tools are frequently stacked and the tabs offer quick switching. For example, in Figure 1-2, the Toolbox is also open (albeit hidden behind the Solution Explorer) and clicking the Toolbox tab would hide the Solution Explorer behind it. They are not strictly part of the Solution Explorer, but rather the pane that holds the four stacked tools.

In the main pane of the Solution Explorer is the list of files that make up your site. At the top is the root, generally in `C:\websites\MyWebSiteName`. In the case of WroxUnited we have used C:\BegASPNET2\WroxUnited. The files are displayed in their subfolders. Using the same techniques as Windows Explorer, you can expand, collapse, cut, copy, and paste files among folders. Solution Explorer recognizes the implied link between an .ASPX file and its .ASPX.VB or .ASPX.CS file (more on these in Chapters 6 and 9). If you copy the .ASPX file the code file will move with it.

You can also right-click a folder and select Add Existing Item. The resulting dialog box allows you to navigate anywhere on your computer and network (including FTP sites) to bring in files. For example,

when you begin to build your project there are times you will be asked to use an image or text file from this book's download at `www.wrox.com`. You can use Solution Explorer to add the item to your web site from your download folder.

ASP.NET 2.0 sites do not contain, during development, a special system of file registration. The files, including ASPX pages, code files, data files, and images are all contained in a normal Windows folder and subfolder structure. If you do not have VWD open, you can cut, copy, and paste your site files directly from Windows Explorer.

The Solution Explorer, like the Toolbar, Data Explorer, and Properties window discussed in the next section, can be placed on the page in one of two modes: *floating* or *dockable*. Floating allows the window to be placed anywhere on the screen, similar to a normal window of a base size (not maximized). Dockable will automatically place the window in one of five locations: top, bottom, left, right, or stacked on another window. Change the mode by selecting one of the windows, and click through Window ➪ Floating or dockable in the menu. In dockable mode you will see, when you drag the window's title bar, some translucent positioners (see Figure 1-3). Drag the title bar onto one of these positioners, and the window will automatically size and place itself in the correct dock.

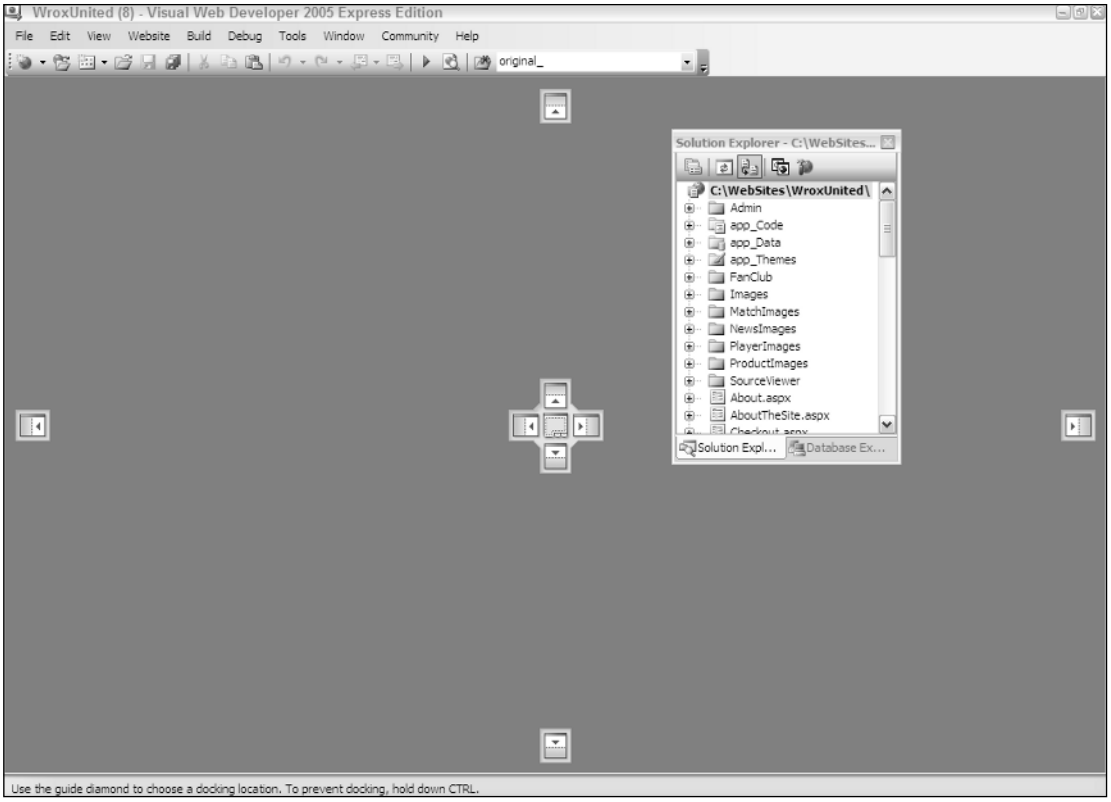

Figure 1-3

Having learned to modify the VWD Express IDE, you can now move on to the techniques of creating web sites and pages within those sites.

Creating, Opening, and Using Web Sites and Pages with VWD

To create a new site you only have to open VWD and click Menu ➪ File ➪ New Site. From the wizard, select an ASP.NET Web Site. Assuming you will create a local copy for development, set the location to File System and browse to the path. Normally this would be `C:\Websites\MyWebsiteName` (our practice site allows backward compatibility with earlier books by using `C:\BegASPNET2\WroxUnited`). You can pick either Visual Basic or C# as your language. Visual Basic is the default for VWD and the samples in the book. VWD will automatically create for you a folder, a default page, and a subfolder for data.

ASP.NET 2.0 introduces a very simple model for file organization and code registration for a web site. All files for the site are stored in a folder and its subfolder. At the time of deployment, that entire group is copied from the development machine to the host. Therefore, you are not required to create a virtual root as in former versions of ASP. Deployment is further simplified by VWD: If you select Menu ➪ Website ➪ Copy Website, VWD opens an FTP screen that you can use to send new or updated files to your host.

To edit an existing site, choose Menu ➪ File ➪ Open Web Site. If you are working locally you can browse to the folder. On the left side of the screen, VWD presents a menu with options to work directly on pages via FTP or through a local IIS installation.

Once the web site is created pages can be added. We usually start by adding some folders to organize the pages. Right-click an existing folder and click the option to add a folder. Special ASP.NET folders, such as those for Code, Themes, and Web References, have their own menu option.

To add a page, right-click a folder and click Add New Item. A wizard presents many choices. For now, you simply need to create a Web Form, but take a look at the other options to get a feel for the capability of VWD and ASP.NET 2.0. Give your new page a name and set its language. Later chapters discuss the two checkboxes. Having introduced you to creating web sites and pages, the following Try It Out puts that knowledge to use, asking you to create the Wrox United site and a couple of practice pages.

The Sample Code Directories

To make things easy, each chapter has its own code, and there are two directories for each chapter, held under one of two higher-level directories. There is a Begin directory, which contains the samples ready for you to work through – it's the samples without any changes. The End directory contains the samples with the Try It Outs completed, so you can use these as a reference as you are working through the examples, or to cut and paste code if the example directs you to do this.

These Begin and End directories appear under a Chapters directory, with each chapter having its own directory. So the starting set of samples for this chapter is under Chapters\Begin\Chapter01, while the finished code for this chapter is under Chapters\End\Chapter01. Some chapters work on the main WroxUnited application, and will therefore contain a copy of the WroxUnited directory, while others have non-WroxUnited samples. The reasons for this is that for some techniques it is easier to see them in smaller, easier to digest samples, than in a fully working application. All of the techniques however, are used in the main application.

Having separate directories, some with complete copies of WroxUited, does mean that the samples are quite large, but the advantage is that each chapter is kept separate from the others, and allows you to work through chapters without mixing up which code came from which chapter.

As well as the code for the chapters, there is a WroxUnited application that contains the final application. This may differ slightly from the samples, but only in that the data may be more complete, and some of the pages look a little nicer.

Try It Out Creating the Wrox United Site and Two Practice Pages

1. Open VWD. Choose Menu ➪ File ➪ New Web Site. Select ASP.NET Web Site located in the File System at `C:\BegASPNET2\Chapters\Begin\`; you can use the Browse button to navigate to this directory, and when you have, type Chapter01 on the end, so that the new Web site is located at `C:\BegASPNET2\Chapters\Begin\Chapter01`. Make sure the Language list is set to Visual Basic. Click OK. You should see your folder on the right side of the screen in the Solution Explorer. If not, choose View ➪ Solution Explorer. Note that VWD automatically builds three items: a folder named `App_Data`, a page named `Default.aspx`, and (if you expand `Default.aspx`) a file named `Default.aspx.vb`, which will hold code for the default page.
2. In the center of VWD you will see a space for editing pages with the `Default.aspx` page opened. Note in the bottom-left a choice of Design and Source. Click each in turn to observe the code and the results of the code. When in Design View, click the page and type the simple text **Home Page**. Press Ctrl+S to save.
3. Create a folder for images by going to your Solution Explorer and right-clicking the root of your site (this will probably show as `C:\...\Chapter01`, as the Solution Explorer hides part of the path) and then clicking Add Folder of the regular type. Name the nascent folder Images.
4. You can manipulate your site's files and folders outside of VWD. Open Windows Explorer and navigate to `C:\BegASPNET2\Chapters\Begin\Chapter01` to see the same set of folders and files as you see in VWD's Solution Explorer.
5. Returning to VWD, right-click the nascent Images folder and click Add an Existing Item. Browse to the folder where you stored the download for this book, probably named `C:\ BegASPNET2\WroxUnited`. Open the Images folder and select all the images. Click Add to actually copy those images from the download folder to your site's image folder.
6. Staying in VWD, now create your first page, the history of Wrox United. Right-click the site's root (`C...\Chapter01`at the top of the Solution Explorer) and select Add New Item from the menu. Select the Web Form template and give it the name History. Accept the other default settings. Click Add, and switch to Design View. Rather than typing text on the page you can copy a short history of the team from a file included in the download. Switch to Windows Explorer and navigate to your `C:\BegASPNET2\Chapters\End\Chapter01` folder. Look for the file named `History.txt`, open it, and select the paragraphs. Switch back to VWD and paste the text into the page. Click the diskette icon of the toolbar to save.
7. Repeat Step 6 for a Mishaps page, whose contents come from `Mishaps.txt`.

How It Works

In this exercise you created your site and the first few pages. By using the menu choices in VWD to create a site, you automatically get some standard folders and files. You could have followed the Microsoft recommendation of storing the site in the `C:\Websites` directory, but you can also put your site in an

alternate folder as we have done at C:\BegASPNET2\ – we did this to keep the book samples separate from any other Web sites you create. As you saw with the Images folder, it is easy to add subfolders to the root to organize your files.

When you created a page in VWD you were offered a few dozen templates. You selected Web Form as the standard plain ASP.NET 2.0 web page. By using cut and paste you have no problems bringing in text from other files.

You also learned that there is no requirement for a special file indexing or storage mechanism in VWD. The files sit in the folders organized by Windows on the hard drive. However, it is better to create and add files in VWD when possible to keep the Solution Explorer view and other VWD features immediately up to date with your changes.

Running a Page

Once a page is created it can be served to a user. Because the server actually executes code in the server-side controls to create the final HTML page, this serving of the page is also called *running* the page, as if you were running a program. VWD has a green triangle tool icon to initiate a run or you can press F5 or choose Menu ➪ Debug ➪ Run. VWD then performs several steps:

1. All pages in the site are compiled to the Microsoft Intermediate Language (MSIL) that is then stored with supporting files in an assembly. At this point development language differences (for example VB and C#) disappear because the result is in MSIL. However, there is no optimization for the hardware that will serve the page.
2. The assembly is Just In Time (JIT)–compiled from MSIL to Native Code that is optimized for the serving machine.
3. A lock is placed on the page that prevents changes in VWD Design View while the page is opened by Cassini.
4. VWD starts Cassini and your browser is opened with a request to Cassini for the page.

> **A common mistake for beginners is to attempt to change a page in VWD's Design View while it is still open in a browser served by Cassini.**

As your site gets larger, you'll find that the compilations take longer. You can press Ctrl+F5 to run a page with a compilation of only that page. In the following Try It Out, you practice running the History and Mishaps pages created in the previous Try It Out.

Try It Out Running a Page

1. In VWD's Solution Explorer, double-click the `History.aspx` file to open it (if it is not already open).
2. Click the Run icon (green arrow) on the toolbar. If there is a message to add a `Web.config` with a Debug, accept the suggestion. Note that your Browser opens and displays the History page.

3. In the Windows tray, the icon of a yellow page with a gear indicates that Cassini is running. Double-click it and you will see that it is pointing to your web site. Close your browser so Cassini unlocks the page.

4. Return to VWD and open the Mishaps page. This time, watch the lower-left corner of VWD as you start to run the page. You will see a message that the build has started and a brief display of an error list box. After seeing the Mishaps page in your browser, switch back to VWD. Note that the page (in Design View) is locked while it is served.

How It Works

This section focuses on running pages from VWD. You can start the run by clicking the green arrow. This action starts Cassini. It also opens your browser and sends a request to Cassini for the page. Once running, you can see the icon for the server in the Windows system tray.

Design Surface

The center of the VWD interface is occupied by the large Design Surface. This is the area where you will do most of your work of adding content to ASP.NET 2.0 pages. You can switch between Design View, which displays a simile of the final page in a browser-like display, or you can switch to Source View, which displays code in a text screen (see Figure 1-4). In general, the Design View is easier and faster for most work because it supports more drag-and-drop features. You can switch to Source View when you need to make those minor changes that are beyond the capability of the VWD drag-and-drop interface.

Figure 1-4

When you add a control to a page in Design View, a Common Tasks Menu may pop up. This mini menu contains the most frequently used setup features for the control. Not all controls have smart task panels, but if it is available it can be opened and closed using the small black triangle at the top corner (shown in Figure 1-5) of a control that is selected.

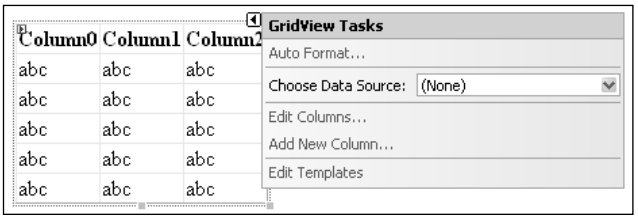

Figure 1-5

Several default settings can be changed in the Design Surface by opening the Tools menu and selecting Options. These options change the way the pages appear to you, as the programmer, when they are opened for editing in VWD. These are not the settings for the appearance of the page to the web site visitor. You can select to start pages in Design View or Source View, as well as the automatic opening of the smart task panel. Being able to revise the number of spaces for tabs and indents helps your projects conform to your company's specifications for web page code.

At the bottom edge of the Design Surface is a navigation tool that is useful in large and complex documents. You can read the navigation tags to find out where the insertion bar (cursor) is currently placed. The current setting will appear to be highlighted, as depicted in Figure 1-6. You can also click a tag and the entire tag will be selected in the Design Surface.

Figure 1-6

The designer is, in many ways, like your word processor. But VWD also offers the two alternatives to viewing a page (Design and Source) as well as enhancements for navigating through the page. The next section discusses how VWD helps you to add features to the page.

Toolbox

VWD offers the set of ASP.NET server-side controls in a Toolbox for easy drag and drop onto the page. Chapter 3 discusses in detail the various server-side controls and how they are used; here you will just get a feel for how to use the Toolbox in general. The Toolbox can be displayed by choosing Menu ➪ View ➪ Toolbox or by pressing Ctrl+Alt+X. Once displayed, the Toolbox can be moved to a new location on the screen by dragging its title bar. As you drag the Toolbox to different areas it will render a compass icon that allows you to drop the toolbar toward the top, bottom, left, or right, as well as on top of other windows. If you are trying to maximize the size of your design surface, you can stack your Solution Explorer and Toolbox on top of each other at one location on the screen.

The Toolbox is organized into several panels that group similar controls. The panels can be expanded to show their tools or collapsed to save space. There is some variation among installations, but a typical set of panels includes the following:

- **Standard** for the majority of ASP.NET 2.0 server-side controls.
- **Data** for data source and data-bound controls.
- **Validation** for controls that reject user input that does not meet your range of acceptable values.
- **Navigation** for menus and breadcrumbs.
- **Login** for the authentication controls.
- **WebParts** for larger components in sites that the user can rearrange or hide.
- **HTML** for generic (non-ASP) tags.
- **General** can be customized.

Figure 1-7 depicts the Toolbox as it will appear on your screen.

Figure 1-7

Clicking the plus icon expands a panel to show its list of available controls. Figure 1-8 shows the Login and Standard panels expanded.

Figure 1-8

The General panel starts out empty. After you have created part of a page, you can select that page and drag it into the General panel to create your own reusable tool. This is useful if you want to duplicate a set of a couple controls with formatting onto several pages.

On the right side of the Toolbox title bar there is a pushpin icon, shown in Figure 1-9. When clicked, the pushpin turns horizontal, meaning that the Toolbox will automatically hide when not in use, leaving only its title bar exposed.

Figure 1-9

The appearance of the Toolbox changes as it is used. For example, the titles of each panel will change as they are selected.

When your mouse moves over the Toolbox title bar, the Toolbox will expand out for your employment, as shown in Figure 1-10.

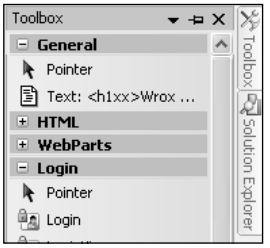

Figure 1-10

In this Try It Out you practice using the Design and Source Views and Toolbox features of VWD.

Try It Out — Using the Views and Toolbox

1. Continue working in VWD with your Mishaps page.
2. Switch to Design View. In the Toolbox, expand the HTML panel and scroll down to the bottom of the panel. Drag a Horizontal Rule from the Toolbox onto the page (anywhere between paragraphs).
3. Your next objective is to add a calendar to the bottom of the History page. Open the Standard panel of the Toolbox and find the ASP.NET calendar control. Drag it to the page. (Double-clicking performs the same operation.) Select the calendar with a single click and notice the small right-facing arrow in the top-right corner. Click it to expose the smart task panel. Click Auto Format, select a format, and click Apply. Observe how easy it is to modify many rendering criteria at once using VWD's dialog interface.
4. Open your History page in VWD. View it in DesignView. Move your insertion bar up to the first line to the tag that begins with `<%@ Page...>`. Note that the navigation guide (at the bottom of

the design panel) shows you that you are in the `<Page>` tag. Click the `<Page>` tag. You will see that the entire tag is selected in the design panel.

How It Works

As you saw, by adding a simple HTML Horizontal Rule, the Toolbox offers the ability to drag and drop elements to the page rather than typing out their tags. Even complex constructs like a calendar are added with just a drag and drop. Once on the page, you can modify an element by using the smart task panel.

Properties Window

An object, such as a web page, a `ListBox` server-side control, or a connection to a data source, has *properties*. Properties are settings that determine how the object appears and behaves. In earlier versions of ASP, many goals were achieved by writing lengthy and complex code. In ASP.NET 2.0, however, most of that code has been pre-written by Microsoft and encapsulated within the server-side controls. Properties determine how that behavior will be exercised. Properties can be very simple, such as BackColor, or very complex, such as EnablePaging. Likewise, the values assigned to a property can be as simple as BLUE or as complex as a multiple-line SQL statement. Properties values can be set by typing them directly into the Code View or by using the Properties window, which is shown in Figure 1-11.

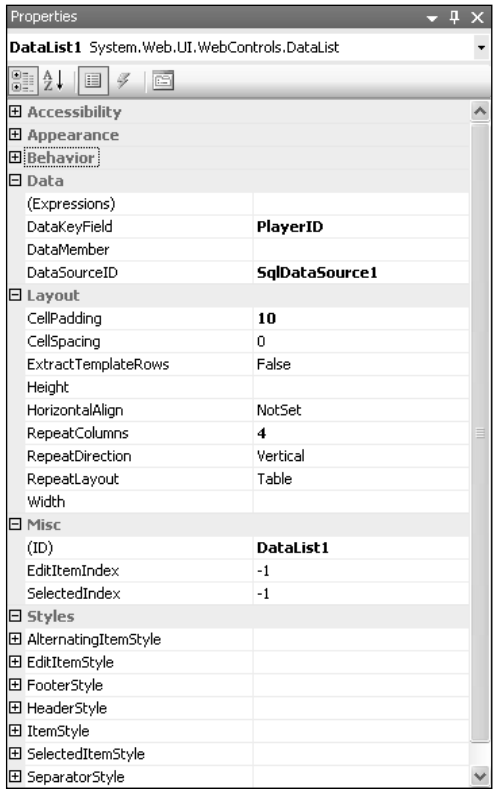

Figure 1-11

Display the Properties window by pressing F4 or by choosing Menu ➪ View ➪ Properties. The properties are organized into panels that can be collapsed or expanded (similar to the Toolbox). For example, in Figure 1-11 the top three panels are collapsed. At the top of the Properties window is a drop-down list containing the names of all of the controls on the page currently open. Below that are icons to arrange the list of properties categorically or alphabetically. The lightning icon will change the Properties window so that it displays events (a topic covered in Chapter 6) rather than properties. The body of the window displays property names on the left and their current values on the right. At the bottom sits a box that gives some help on the currently selected property.

The fundamental technique used to change properties is to select an object, usually a control, and then find the property of interest and set it. You can select the object with a single click on the object in Design View or by locating the insertion bar in the object in Source View. Alternatively, you can select an object from a drop-down list at the top of the Properties window. One common mistake arises when you attempt to change the properties but have not first actually selected the object you want to change. You end up changing an object that was still selected from earlier work.

You have several options for setting the value of a property. If the property has a limited number of allowed values (such as true and false), you can double-click the property name to toggle through the values. If there are more than a few options, but still a finite amount, the values will be in a drop-down list. Some properties have many options, and their value box offers an ellipses button that takes you to a dialog box. Last, some properties can accept strings, so their values are set in text boxes. It is always better to select or to toggle a value rather than type it. Once a value has been changed, you must press Enter or Tab or click another property in order to set the value. At that time the Design View will render the change.

You can also set property values by typing into the Source View. Locate the insertion bar within a tag and press the space bar to display an IntelliSense list of all the properties that can be inserted at the location of the insertion bar. Type the first letter or two then an equals sign. IntelliSense will then display all of the legal values (if the list is finite), and again type the first letter or two of the value you desire. Finish by typing a space. Note that there will be no value list if the range of possible property values is not finite. For example, if the value is a SQL statement you will have to type the statement without IntelliSense.

Error List Window

When problems arise you have two major paths for obtaining clues about the problem. First is an error report on the page delivered to the browser, and second is the Error List window within VWD.

ASP.NET 2.0 will give an error report (shown in Figure 1-12) on the page sent to the browser if the following shaded line is in your `Web.config` file:

```
   <system.web>
            <compilation debug="true">
             </compilation>
```

Note that the first time you run (F5) a page you will get a default of <compilation debug=”true” strict=”false” explicit=”true”/>.

Because the default is `true`, you will have debug turned on if there is no specific attribute. So `debug=true` for the following case as well where there are no changes to the default. Of course, having it literally set to `true` helps other programmers on your team that may be looking through your settings:

```
<system.web>
        <compilation></compilation>
```

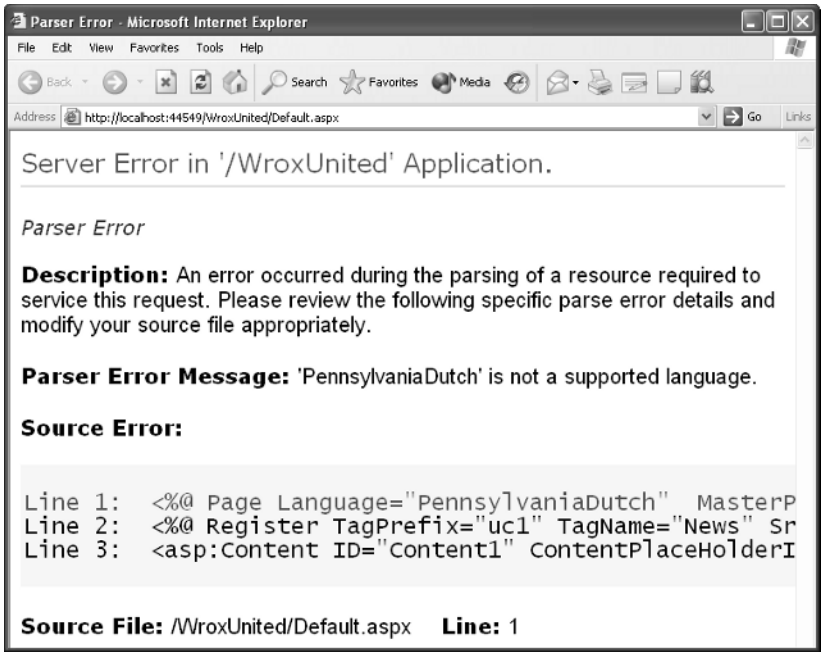

Figure 1-12

A fatal error on the page produces a characteristic white and yellow page on your browser with an error message. Just below the yellow block is the name of the offending file and the line containing the failure. As with all errors, the error may actually be related to the line number displayed, but this should give you a good clue. Note that when you deploy a site the `debug` command in `Web.config` should be set to `false` to improve performance and to reduce information given to hackers. Error handling is covered in greater depth in Chapter 15.

The second way to identify errors comes from within VWD itself via the Error List window, depicted in Figure 1-13. By default, the window is hidden until you run a page. You can force it to be displayed by choosing Menu ➪ View ➪ Error List. The window displays all of the errors encountered during the conversion of the page into the MSIL.

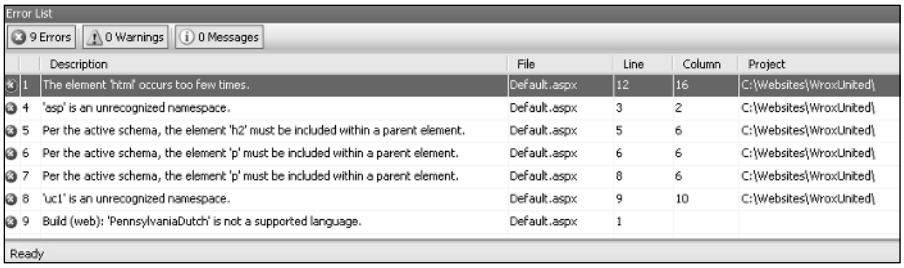

Figure 1-13

Note that the top of the Error List window has three clickable icons: Errors, Warnings, and Messages, which display different lists of items created when the page was built. Hiding a type of item does not remove it from the list; rather, doing so only hides the item so the list is shorter. The second column from the left identifies the order in which the errors occurred.

Errors cause the page as a whole, or some portion of it, to fail. This includes, for example, references to objects that do not exist.

Warnings are problems that could be solved by VWD while building the page (for example, the lack of a closing tag).

Messages are sets of texts that the programmer can include in the code to appear when IIS is building the page.

When viewing the list of items you can sort by clicking a column heading. If you hold Shift you can click a second column for tiebreakers (to determine the order for records with the same value in the primary column). You can also resize the columns by dragging their dividers, or re-order them by dragging the column name left or right. Double-clicking an item allows you to open the offending file and jump the cursor to the offending line.

In this Try It Out you practice changing properties of an image control on the History page. Then you create some errors and observe the results.

Try It Out — Using the Properties and Error Lists Windows

1. Open the History page in Design View. In the Solutions window, open the Images folder and drag the jpeg named logo-yellow to the top of the History page. VWD will automatically create an image with its source set to the jpeg.
2. Click the image once to select it, and then switch to the Properties window (or open it with F4). Change the height from 447 to 100 and press Enter to set the value. As you can see, changes are immediately visible in the design panel.
3. Now modify properties from Source View. Find the `<img>` control and within it the property (attribute) for height. Change it from 100 to 300. Click the green arrow (or press F5) to run the page to see the result.

4. Close your browser and return to Source View. Locate your insertion bar in the `<img>` tag immediately after the closing quote of `height="300"` and press the space bar. This opens the IntelliSense list with all of the properties that are suitable at this point in your page. Press the t key and then the i key to move down the list to the Title property. Press the equal (=) key to close the IntelliSense and type in **"Wrox Logo In Yellow"** including the start and end quotes. Run the page, and in your browser note that when you mouse over the image you see the title you created. Close the browser.
5. Next, introduce a non-fatal error. Open the History page in Source View and find the `<h1>` tag near the top of the page. Change the `<h1>` tag to `<H1xx>` and run the page. The browser opens and you can see that the text "Wrox United — a potted history" fails to render in heading-one style. Close the browser and switch back to VWD to observe the Errors List window. If not already visible, choose Menu ➪ View ➪ Error List to view the Errors List window. Note that two errors were entered in the list. First was a note that "h1xx" on line 3 is not supported. Second, the closing tag of `</h1>` on line 4 no longer matches an opening tag.
6. Your last experiment is to introduce a fatal error. In Source View go to the top of the page and change the first line from `Language = "VB"` to `Language="Esperanto"` and run the page.
7. You will deal with solving many kinds of errors in each chapter of this book. For now, return your page to its original form by deleting the `<img>` tag, restoring the goofy `<h1xx>` to the proper `<h1>`, and changing the language back to VB.
8. Save the page.

How It Works

You experimented with three ways to change a property. First you worked in Design View and changed a property by typing its new value into the Properties window. Next you made a change by hand-typing a new value into the source code. Last, you used IntelliSense to guide you through adding a property to an existing control.

You observed the results of two types of errors: fatal and non-fatal. In the first case, ASP.NET 2.0 could still render the page even though the faulty tag of `<h1xx>` left your text as default, not heading one. Although the page rendered, back in VWD an error message was logged on to the Error List window. You introduced a more serious error when you changed the language value to a non-supported language. ASP.NET 2.0 could not overcome this error and so you see two results: In the browser you got the error page with troubleshooting information, and back in VWD you got entries to the error list.

VWD's Database Explorer

When you begin to work with data (in Chapter 7 and following) you can use tools in VWD to gain knowledge about your data sources. This information includes the exact names of tables and columns. In fact, as you see later in this book, you can drag columns to the designer and VWD will do all the work of setting up the proper controls to display data from those columns. For now, understand that in the Solution Explorer you can double-click the name of an Access MDB to open that file in Access (assuming Access is installed). For SQL Server databases (as used in this book) you can use a tool named the Database Explorer to do even more exploration of a database and change its data and properties. These features are discussed in detail in Appendix D.

Summary

Microsoft has revised large parts of ASP.NET in version 2.0. The overall biggest benefit is that tasks, which formerly required custom coding, can now be implemented by dragging pre-built controls to the page. These pre-built controls include tools for logging on users, navigation, connecting to stores of data, displaying data, creating a consistent look to the site, and offering customization options to the user. The result is both faster and more robust development of dynamic web pages. On top of this Microsoft has made version 2.0 easier to deploy and faster in performance. As with earlier versions of ASP, the execution of code (that is, the building of dynamic pages) occurs on the server and only standard HTML is sent to the browser. Thus, ASP.NET 2.0 is compatible with all browsers.

Three tools are available from Microsoft for creating ASP.NET 2.0 pages. The one used in this book, Visual Web Developer Express (VWD). VWD displays the organization of pages on your site, and helps you to build new pages or modify existing pages. VWD also comes with a lightweight web server named Cassini for testing your pages. After building a page you can click Run and VWD will start Cassini, start your browser, build the page, and serve it to the browser. This chapter also covered the following topics:

- VWD offers many options for the way that you view and work with pages during their development. Tabs allow switching between Design View (which displays a good facsimile of how the browser will render the page) and Source View (which shows the tags and code that generate the page).
- When creating a new site or adding pages, VWD offers wizards and templates that walk you through the most common setups. In this chapter you looked at how to create a new page based on one of several dozen templates, followed by working with the Toolbox. This source of pre-built objects is a focus of building pages in all the exercises of this book. To organize the large Toolbox, the tools are divided into groups.
- Another window displays properties of whichever object is currently selected. You can, for example, select a text box and see its size, background color, and dozens of other properties. The remainder of this book goes into the details of many properties of objects that ASP.NET 2.0 supports on a page.
- When a page is built as a result of the VWD Run command, you get some feedback on how the process fared. Fatal errors are listed, as well as warnings about potential problems with the page. Double-clicking any of those errors will lead you to the offending line in the site.

This first chapter focused on an introduction to ASP.NET 2.0 and how to build your first pages using VWD. Chapter 2 moves on to understanding some of the ASP.NET 2.0 features that govern the look and feel of all the pages on a site.

Exercises

1. Explain the differences among the .NET 2.0 Framework, ASP.NET 2.0, VWD, and IIS.
2. List some differences between Cassini and IIS.
3. When you drag the title bar of the toolbar it will only go to certain locations and certain sizes. How can you put the title bar where you want it?

4. How can you copy a .jpg file in `C:\MyPhotos` into your site for display on a page?
5. You want to add a subfolder to your site, but Folder is not one of the items listed in Add Items. Why?
6. Microsoft has written extensive code to make it easier for programmers to create Web pages. How does a programmer actually use that code?
7. Why are there no tools in the General panel of the Toolbox?