1

Hello ASP.NET 2.0!

The evolution of ASP.NET continues! The progression from Active Server Pages 3.0 to ASP.NET 1.0 was revolutionary, to say the least; and we are here to tell you that the evolution from ASP.NET 1.0/1.1 to ASP.NET 2.0 is just as exciting and dramatic.

The introduction of ASP.NET 1.0/1.1 changed the Web programming model; but ASP.NET 2.0 is just as revolutionary in the way it increases productivity. The primary goal of ASP.NET 2.0 is to enable you to build powerful, secure, and dynamic applications using the least possible amount of code. Although this book covers the new features provided by ASP.NET 2.0, it also covers most of what the ASP.NET technology offers.

A Little Bit of History

Before organizations were even thinking about developing applications for the Internet, much of the application development focused on thick desktop applications. These thick-client applications were used for everything from home computing and gaming to office productivity and more. No end was in sight for the popularity of this application model.

During that time, Microsoft developers developed its thick-client applications using mainly Visual Basic (VB).

Visual Basic was not only a programming language; it was tied to an IDE that allowed for easy thick-client application development. In the Visual Basic model, developers could drop controls onto a form, set properties for these controls, and provide code behind them to manipulate the events of the control. For example, when an end user clicked a button on one of the Visual Basic forms, the code behind the form handled the event.

Then, in the mid-1990s, the Internet arrived on the scene. Microsoft was unable to move the Visual Basic model to the development of Internet-based applications. The Internet definitely had a lot of power, and right away the problems facing the thick-client application model were revealed. Internet-based applications created a single instance of the application that everyone could access. Having one instance of an application meant that when the application was upgraded or patched, the changes made to this single instance were immediately available to each and every user visiting the application through a browser.

To participate in the Web application world, Microsoft developed Active Server Pages (ASP). ASP was a quick and easy way to develop Web pages. ASP pages consisted of a single page that contained a mix of markup and languages. The power of ASP was that you could include VBScript or JScript code instructions in the page executed on the Web server before the page was sent to the end user's Web browser. This was an easy way to create dynamic Web pages customized based on instructions dictated by the developer.

ASP used script between brackets and percentage signs — `<% %>` — to control server-side behaviors. A developer could then build an ASP page by starting with a set of static HTML. Any dynamic element needed by the page was defined using a scripting language (such as VBScript or JScript). When a user requested the page from the server by using a browser, the `asp.dll` (an ISAPI application that provided a bridge between the scripting language and the Web server) would take hold of the page and define all the dynamic aspects of the page on-the-fly based on the programming logic specified in the script. After all the dynamic aspects of the page were defined, the result was an HTML page output to the browser of the requesting client.

As the Web application model developed, more and more languages mixed in with the static HTML to help manipulate the behavior and look of the output page. Over time, such a large number of languages, scripts, and plain text could be placed in a typical ASP page that developers began to refer to pages that utilized these features as *spaghetti code*. For example, it was quite possible to have a page that used HTML, VBScript, JavaScript, Cascading Style Sheets, T-SQL, and more. In certain instances, it became a manageability nightmare.

ASP evolved and new versions were released. ASP 2.0 and 3.0 were popular because the technology made it relatively straightforward and easy to create Web pages. Their popularity was enhanced because they appeared in the late 1990s, just as the dotcom era was born. During this time, a mountain of new Web pages and portals were developed, and ASP was one of the leading technologies individuals and companies used to build them. Even today, you can still find a lot of `.asp` pages on the Internet — including some of Microsoft's own Web pages.

But even at the time of the final release of Active Server Pages in late 1998, Microsoft employees Marc Anders and Scott Guthrie had other ideas. Their ideas generated what they called XSP (an abbreviation with no meaning) — a new way of creating Web applications in an object-oriented manner instead of the procedural manner of ASP 3.0. They showed their idea to many different groups within Microsoft, and were well received. In the summer of 2000, the beta of what was then called ASP+ was released at Microsoft's Professional Developers Conference. The attendees eagerly started working with it. When the technology became available (with the final release of the .NET Framework 1.0), it was renamed ASP.NET — receiving the .NET moniker that most of Microsoft's new products were receiving at that time.

Before the introduction of .NET, the model that classic ASP provided and what developed in Visual Basic were so different that few VB developers also developed Web applications — and few Web application developers also developed the thick-client applications of the VB world. There was a great divide. ASP.NET bridged this gap. ASP.NET brought a Visual Basic–style eventing model to Web application development, providing much-needed state management techniques over stateless HTTP. Its model is much like the earlier Visual Basic model in that a developer can drag and drop a control onto a design surface or form, manipulate the control's properties, and even work with the code behind these controls to act on certain events that occur during their lifecycles. What ASP.NET created is really the best of both models, as you will see throughout this book.

I know you'll enjoy working with this latest release of ASP.NET 2.0. Nothing is better than getting your hands on a new technology and seeing what's possible. The following section discusses the goals of ASP.NET 2.0 so you can find out what to expect from this new offering!

The Goals of ASP.NET 2.0

ASP.NET 2.0 is a major release of the product and is an integral part of the .NET Framework 2.0. This release of the Framework was code-named *Whidbey* internally at Microsoft. You might hear others referring to this release of ASP.NET as *ASP.NET Whidbey*. ASP.NET 2.0 heralds a new wave of development that should eliminate any of the remaining barriers to adopting this new way of coding Web applications.

When the ASP.NET team started working on ASP.NET 2.0, it had specific goals to achieve. These goals focused around developer productivity, administration, and management, as well as performance and scalability. These goals are achieved with this milestone product release. The next sections look at each of these goals.

Developer Productivity

Much of the focus of ASP.NET 2.0 is on productivity. Huge productivity gains were made with the release of ASP.NET 1.*x*; could it be possible to expand further on those gains?

One goal the development team had for ASP.NET 2.0 was to eliminate much of the tedious coding that ASP.NET originally required and to make common ASP.NET tasks easier. The ASP.NET team developing ASP.NET 2.0 had the goal of reducing by two-thirds the number of lines of code required for an ASP.NET application! It succeeded in this release; you will be amazed at how quickly you can create your applications in ASP.NET 2.0.

The new developer productivity capabilities are presented throughout this book. Before venturing into these new capabilities, first start by taking a look at the older ASP.NET technology in order to make a comparison to ASP.NET 2.0. Listing 1-1 provides an example of using ASP.NET 1.0 to build a table in a Web page that includes the capability to perform simple paging of the data provided.

Listing 1-1: Showing data in a DataGrid server control with paging enabled (VB only)

```
<%@ Page Language="VB" AutoEventWireup="True" %>
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>

<script runat="server">

    Private Sub Page_Load(ByVal sender As System.Object, _
```

(continued)

Listing 1-1: *(continued)*

```
        ByVal e As System.EventArgs)
          If Not Page.IsPostBack Then
              BindData()
          End If
      End Sub

      Private Sub BindData()
          Dim conn As SqlConnection = New SqlConnection("server='localhost';
             trusted_connection=true; Database='Northwind'")
          Dim cmd As SqlCommand = New SqlCommand("Select * From Customers", conn)
          conn.Open()

          Dim da As SqlDataAdapter = New SqlDataAdapter(cmd)
          Dim ds As New DataSet

          da.Fill(ds, "Customers")

          DataGrid1.DataSource = ds
          DataGrid1.DataBind()
      End Sub

      Private Sub DataGrid1_PageIndexChanged(ByVal source As Object, _
        ByVal e As System.Web.UI.WebControls.DataGridPageChangedEventArgs)
          DataGrid1.CurrentPageIndex = e.NewPageIndex
          BindData()
      End Sub

  </script>
  <html>
  <head>
  </head>
  <body>
      <form runat="server">
          <asp:DataGrid id="DataGrid1" runat="server" AllowPaging="True"
           OnPageIndexChanged="DataGrid1_PageIndexChanged"></asp:DataGrid>
      </form>
  </body>
  </html>
```

Although quite a bit of code is used here, this is a dramatic improvement over the amount of code required to accomplish this task using classic Active Server Pages 3.0. We won't go into the details of this older code; we just want to demonstrate that in order to add any additional common functionality (such as paging) for the data shown in a table, the developer had to create custom code.

This is one area where the new developer productivity gains are most evident. ASP.NET 2.0 now provides a new control called the GridView server control. This control is much like the DataGrid server control that you may already know and love, but the GridView server control (besides offering many other new features) contains the built-in capability to apply paging, sorting, and editing of data with relatively little work on your part. Listing 1-2 shows you an example of the GridView server control. This example builds a table of data from the Customers table in the Northwind database that includes paging.

Listing 1-2: Viewing a paged dataset with the new GridView server control

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

<script runat="server">

</script>

<html xmlns=http://www.w3.org/1999/xhtml>
<head runat="server">
    <title>GridView Demo</title>
</head>
<body>
    <form runat="server">
        <asp:GridView ID="GridView1" Runat="server" AllowPaging="True"
         DataSourceId="Sqldatasource1" />
        <asp:SqlDataSource ID="SqlDataSource1" Runat="server"
         SelectCommand="Select * From Customers"
         ProviderName="System.Data.OleDb"
         ConnectionString="Provider=SQLOLEDB;Server=localhost;uid=sa;
         pwd=password;database=Northwind" />
    </form>
</body>
</html>
```

That's it! You can apply paging by using a couple of new server controls. You turn on this capability using a server control attribute, the `AllowPaging` attribute of the GridView control:

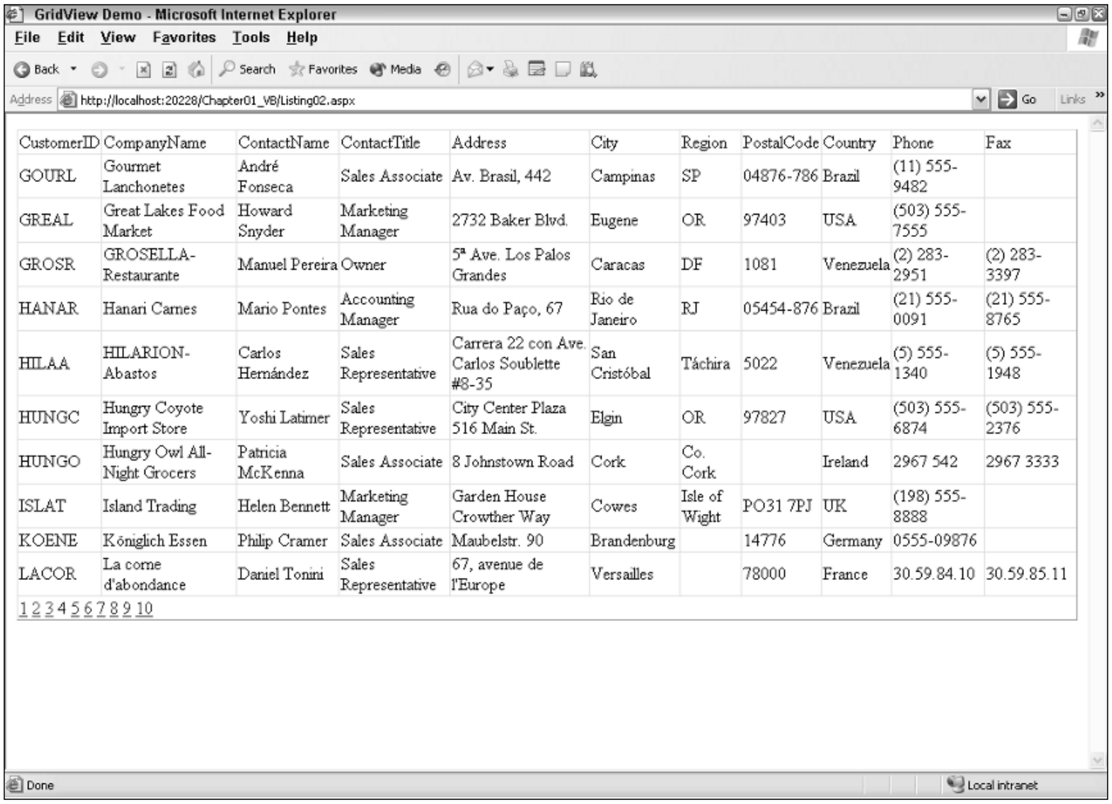
```
<asp:GridView ID="GridView1" Runat="server" AllowPaging="True"
 DataSourceId="SqlDataSource1" />
```

The other interesting event occurs in the code section of the document:

```
<script runat="server">

</script>
```

These two lines of code aren't actually needed to run the file. They are included here to make a point — *you don't need to write any server-side code to make this all work!* You have to include only some server controls: one control to get the data and one control to display the data. Then the controls are wired together. Running this page produces the results shown in Figure 1-1.

This is just one of thousands of possible examples, so at this point you likely can't grasp how much more productive you can be with ASP.NET 2.0. As you work through the book, however, you will see plenty of examples that demonstrate this new level of productivity.


GridView Demo - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address
Go Links
Done
Local intranet

http://localhost:20228/Chapter01_VB/Listing02.aspx

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
GOURL	Gourmet Lanchonetes	André Fonseca	Sales Associate	Av. Brasil, 442	Campinas	SP	04876-786	Brazil	(11) 555-9482	
GREAL	Great Lakes Food Market	Howard Snyder	Marketing Manager	2732 Baker Blvd.	Eugene	OR	97403	USA	(503) 555-7555	
GROSR	GROSELLA-Restaurante	Manuel Pereira	Owner	5ª Ave. Los Palos Grandes	Caracas	DF	1081	Venezuela	(2) 283-2951	(2) 283-3397
HANAR	Hanari Carnes	Mario Pontes	Accounting Manager	Rua do Paço, 67	Rio de Janeiro	RJ	05454-876	Brazil	(21) 555-0091	(21) 555-8765
HILAA	HILARION-Abastos	Carlos Hernández	Sales Representative	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	Táchira	5022	Venezuela	(5) 555-1340	(5) 555-1948
HUNGC	Hungry Coyote Import Store	Yoshi Latimer	Sales Representative	City Center Plaza 516 Main St.	Elgin	OR	97827	USA	(503) 555-6874	(503) 555-2376
HUNGO	Hungry Owl All-Night Grocers	Patricia McKenna	Sales Associate	8 Johnstown Road	Cork	Co. Cork		Ireland	2967 542	2967 3333
ISLAT	Island Trading	Helen Bennett	Marketing Manager	Garden House Crowther Way	Cowes	Isle of Wight	PO31 7PJ	UK	(198) 555-8888	
KOENE	Königlich Essen	Philip Cramer	Sales Associate	Maubelstr. 90	Brandenburg		14776	Germany	0555-09876	
LACOR	La corne d'abondance	Daniel Tonini	Sales Representative	67, avenue de l'Europe	Versailles		78000	France	30.59.84.10	30.59.85.11

1 2 3 4 5 6 7 8 9 10

Figure 1-1

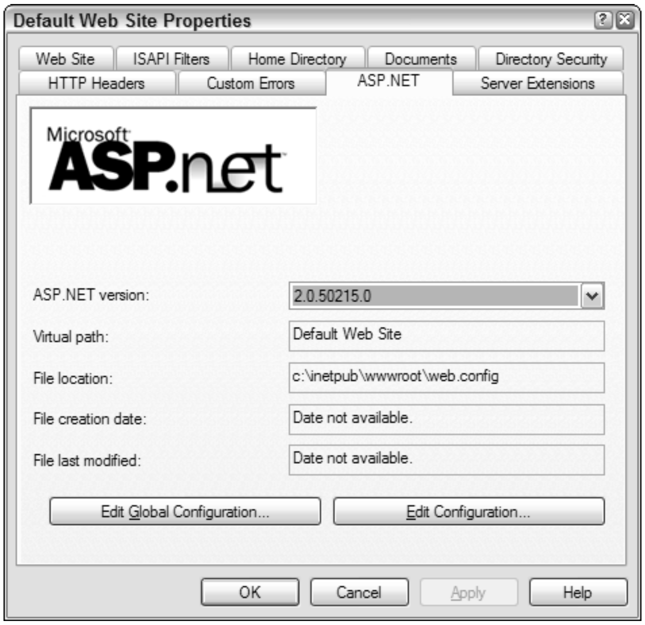
Administration and Management

The initial release of ASP.NET focused on the developer, and little thought was given to the people who had to administer and manage all the ASP.NET applications that were built and deployed. Instead of working with consoles and wizards as they did in the past, administrators and managers of these new applications now had to work with unfamiliar XML configuration files such as `machine.config` and `web.config`.

To remedy this situation, ASP.NET 2.0 now includes a Microsoft Management Console (MMC) snap-in that enables Web application administrators to edit configuration settings easily on the fly through IIS. Figure 1-2 shows the ASP.NET Configuration Settings dialog open on one of the available tabs.

This dialog allows system administrators to edit the contents of the `machine.config` and the `web.config` files directly from the dialog instead of having them examine the contents of an XML file.

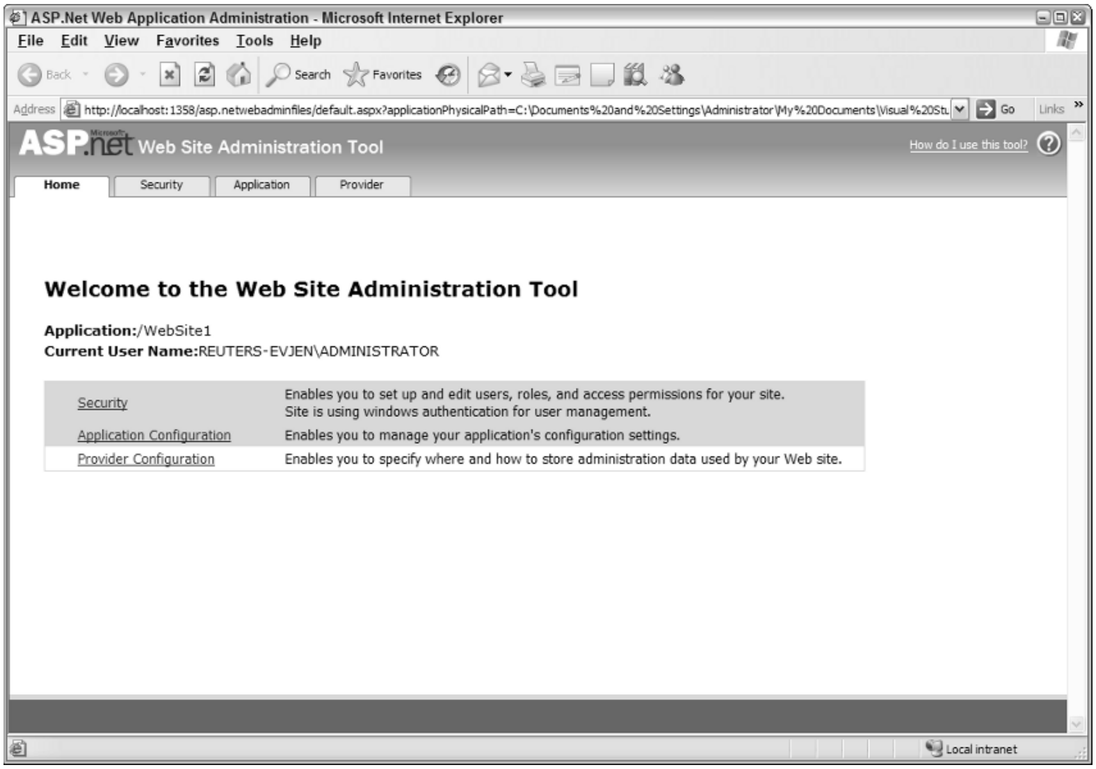
In addition to this dialog, Web or system administrators have a Web-based way to administer their ASP.NET 2.0 applications — using the new Web Administration Tool shown here in Figure 1-3 (this is also covered extensively in Chapter 33 of this book).



Default Web Site Properties

ASP.NET version: 2.0.50215.0
Virtual path: Default Web Site
File location: c:\inetpub\wwwroot\web.config
File creation date: Date not available.
File last modified: Date not available.

Figure 1-2



ASP.Net Web Application Administration - Microsoft Internet Explorer

Address http://localhost:1358/asp.netwebadminfiles/default.aspx?applicationPhysicalPath=C:\Documents%20and%20Settings\Administrator\My%20Documents\Visual%20St

Welcome to the Web Site Administration Tool

Application:/WebSite1
Current User Name:REUTERS-EVJEN\ADMINISTRATOR

Security	Enables you to set up and edit users, roles, and access permissions for your site. Site is using windows authentication for user management.
Application Configuration	Enables you to manage your application's configuration settings.
Provider Configuration	Enables you to specify where and how to store administration data used by your Web site.

Figure 1-3

You might be asking yourself how you can access these new tools programmatically. Well, that's the exciting part. These tools build off new APIs that are now part of the .NET Framework 2.0 and that are open to developers. These new APIs give you programmatic access to many of the configurations of your Web applications such as reading and writing to `.config` files. They enable you to create similar tools or even deployment and management scripts.

In addition to these new capabilities, one exciting new feature allows you to easily encrypt sections of your configuration files. In the past, many programmers stored vital details — such as usernames, passwords, or even their SQL connection strings — directly in the `web.config` file. With the capability to easily encrypt sections of these files, you can now store these items in a more secure manner. As an example, suppose you have a `<connectionStrings>` section in your `web.config` file, like this:

```
<connectionStrings>
   <add name="Northwind"
    connectionString="Server=localhost;Integrated Security=True;Database=Northwind"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

You can now use the new `Configuration` class to encrypt this portion of the `web.config` file. Doing this causes the `<connectionStrings>` section of the `web.config` file to be changed to something similar to the following:

```
<connectionStrings configProtectionProvider="RsaProtectedConfigurationProvider">
   <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
    xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod
       Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
         <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
            <EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
            <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
               <KeyName>Rsa Key</KeyName>
            </KeyInfo>
            <CipherData>
               <CipherValue>
                tnbdGPpmif2LOhhzS/fmzCV798XkhdRnK0IrCUrC5q
                AnK8NMqHHVEVYKjIyXvlR/ns7Oofih6YzI+z
                8f5Dh6HrekLS9yR0AxuSeHda/VJA7bmHdL+18
                30nDK0fbrD22BvHTwpYVYNsE6jhYeGEdUBqE1
                oIK9mV3nQSjzqr7GbkI=
               </CipherValue>
            </CipherData>
         </EncryptedKey>
      </KeyInfo>
      <CipherData>
        <CipherValue>
         gnzaR6BRvj76nUO0nsajgnrLwt72qGY6Sw9
         PkM77vk4YTo3816LWDbiVkUuwEpekwN/EPE
         cXGLUxopKVOK97rCqCoLNOQY16jKpPBTTp8
         bY3WGwVhxxGhezdV+EkWaLN8jXpaBSnGYKH
         mY9l4DoWaH9mugrr2a5Y3JB42XPUKJBtrF0
         VZj48Hwkb/zOD7ggWmJujiFH5xQ/FrIC76I
         16QKSInJiZ8dZU
```

```
            </CipherValue>
         </CipherData>
      </EncryptedData>
   </connectionStrings>
```

Now if some malicious user illegally gets into your machine and gets his hands on your application's web.config file, you could prevent him from getting much of value — such as the connection string of your database.

Performance and Scalability

One of the goals for ASP.NET 2.0 set by the Microsoft team was to provide the world's fastest Web application server. This book also addresses a number of performance enhancements available in ASP.NET 2.0.

One of the most exciting performance enhancements is the new caching capability aimed at exploiting Microsoft's SQL Server. ASP.NET 2.0 now includes a feature called *SQL cache invalidation*. Before ASP.NET 2.0, it was possible to cache the results that came from SQL Server and to update the cache based on a time interval — for example, every 15 seconds or so. This meant that the end user might see stale data if the result set changed sometime during that 15-second period.

In some cases, this time interval result set is unacceptable. In an ideal situation, the result set stored in the cache is destroyed if any underlying change occurs in the source from which the result set is retrieve — in this case, SQL Server. With ASP.NET 2.0, you can make this happen with the use of SQL cache invalidation. This means that when the result set from SQL Server changes, the output cache is triggered to change, and the end user always sees the latest result set. The data presented is never stale.

Another big area of change in ASP.NET is in the area of performance and scalability. ASP.NET 2.0 now provides 64-bit support. This means that you can now run your ASP.NET applications on 64-bit Intel or AMD processors.

Because ASP.NET 2.0 is fully backward compatible with ASP.NET 1.0 and 1.1, you can now take any former ASP.NET application, recompile the application on the .NET Framework 2.0, and run it on a 64-bit processor.

Additional New Features of ASP.NET 2.0

You just learned some of the main goals of the ASP.NET team that built ASP.NET 2.0. To achieve these goals, the team built a mountain of new features into ASP.NET. A few of them are described in the following sections.

New Developer Infrastructures

An exciting advancement in ASP.NET 2.0 is that new infrastructures are in place for you to use in your applications. The ASP.NET team selected some of the most common programming operations performed with ASP.NET 1.0 to be built directly into ASP.NET. This saves you considerable time and coding.

Membership and Role Management

In earlier versions, if you were developing a portal that required users to log in to the application to gain privileged access, invariably you had to create it yourself. It can be tricky to create applications with areas that are accessible only to select individuals.

With ASP.NET 2.0, this capability is now built in. You can now validate users as shown in Listing 1-3.

Listing 1-3: Validating a user in code

VB

```
If (Membership.ValidateUser (Username.Text, Password.Text)) Then
   ' Allow access code here
End If
```

C#

```
if (Membership.ValidateUser (Username.Text, Password.Text)) {
   // Allow access code here
}
```

A new series of APIs, controls, and providers in ASP.NET 2.0 enable you to control an application's user membership and role management. Using these APIs, you can easily manage users and their complex roles — creating, deleting, and editing them. You get all this capability by using the APIs or a built-in Web tool called the Web Site Administration Tool.

As far as storing users and their roles, ASP.NET 2.0 uses an `.mdb` file (the file type for the new SQL Server Express Edition, not to be confused with Microsoft Access) for storing all users and roles. You are in no way limited to just this data store, however. You can expand everything offered to you by ASP.NET and build your own providers using whatever you fancy as a data store. For example, if you want to build your user store in LDAP or within an Oracle database, you can do so quite easily.

Personalization

One advanced feature that portals love to offer their membership base is the capability to personalize their offerings so that end users can make the site look and function however they want. The capability to personalize an application and store the personalization settings is now completely built into the ASP.NET framework.

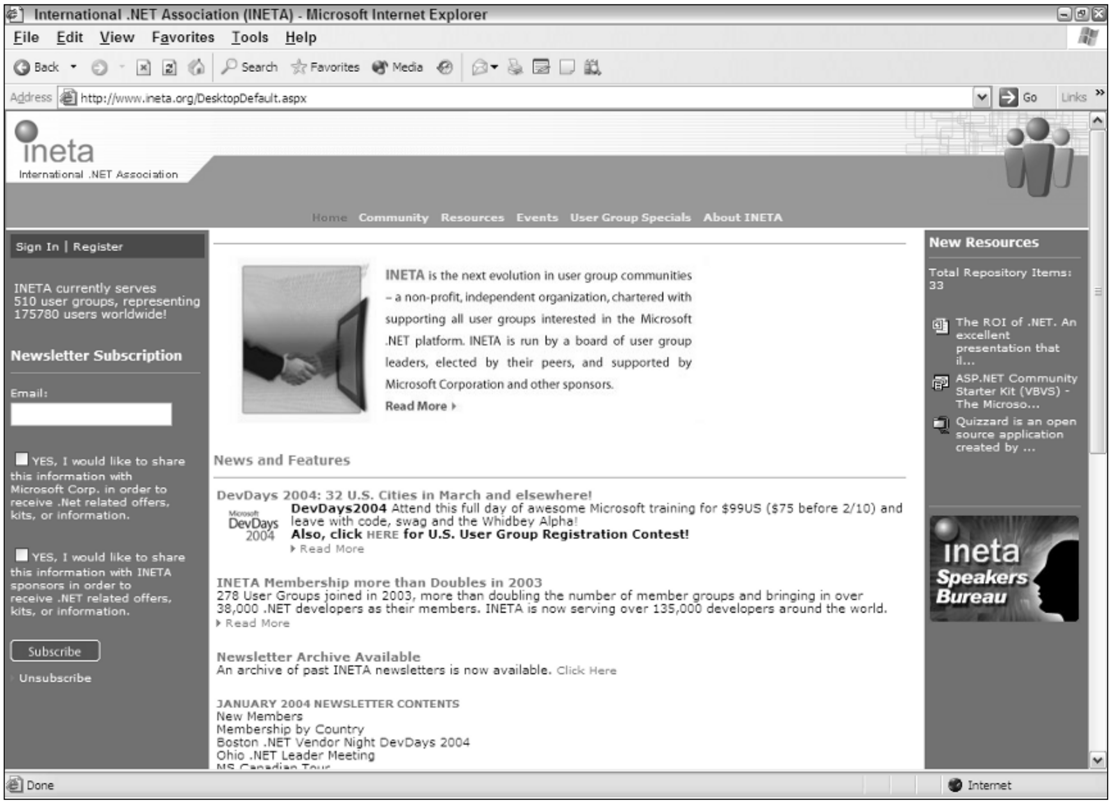
Because personalization usually revolves around a user and possibly a role that this user participates in, the personalization architecture can be closely tied to the membership and role infrastructures. You have a couple of options for storing the created personalization settings. The capability to store these settings in either Microsoft Access or in SQL Server is built into ASP.NET 2.0. As with the capabilities of the membership and role APIs, you can use the flexible provider model, and then either change how the built-in provider uses the available data store or build your own custom data provider to work with a completely new data store. The personalization API also supports a union of data stores, meaning that you can use more than one data store if you want.

Because it is so easy to create a site for customization using these new APIs, this feature is quite a value-add for any application you build.

The ASP.NET Portal Framework

During the days of ASP.NET 1.0, developers could go to the ASP.NET team's site (found at `http://www.asp.net`) and download some Web application demos such as IBuySpy. These demos are known as Developer Solution Kits and are used as the basis for many of the Web sites on the Internet today. Some were even extended into Open Source frameworks such as DotNetNuke.

The nice thing about IBuySpy was that you could use the code it provided as a basis to build either a Web store or a portal. You simply took the base code as a starting point and extended it. For example, you could change the look and feel of the presentation part of the code or introduce advanced functionality into its modular architecture. Developer Solution Kits are quite popular because they make performing these types of operations so easy. Figure 1-4 shows the INETA (International .NET Association) Web site, which builds on the IBuySpy portal framework.


International .NET Association (INETA) - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.ineta.org/DesktopDefault.aspx
Go Links
ineta
International .NET Association
Home Community Resources Events User Group Specials About INETA
Sign In | Register
Newsletter Subscription
Email:
Subscribe
Unsubscribe
New Resources
ineta Speakers Bureau
Done
Internet

INETA currently serves 510 user groups, representing 175780 users worldwide!

YES, I would like to share this information with Microsoft Corp. in order to receive .Net related offers, kits, or information.

YES, I would like to share this information with INETA sponsors in order to receive .NET related offers, kits, or information.

INETA is the next evolution in user group communities – a non-profit, independent organization, chartered with supporting all user groups interested in the Microsoft .NET platform. INETA is run by a board of user group leaders, elected by their peers, and supported by Microsoft Corporation and other sponsors.
Read More

News and Features

DevDays 2004: 32 U.S. Cities in March and elsewhere!
Microsoft DevDays 2004
DevDays2004 Attend this full day of awesome Microsoft training for $99US ($75 before 2/10) and leave with code, swag and the Whidbey Alpha!
Also, click HERE for U.S. User Group Registration Contest!
Read More

INETA Membership more than Doubles in 2003
278 User Groups joined in 2003, more than doubling the number of member groups and bringing in over 38,000 .NET developers as their members. INETA is now serving over 135,000 developers around the world.
Read More

Newsletter Archive Available
An archive of past INETA newsletters is now available. Click Here

JANUARY 2004 NEWSLETTER CONTENTS
New Members
Membership by Country
Boston .NET Vendor Night DevDays 2004
Ohio .NET Leader Meeting

Total Repository Items: 33

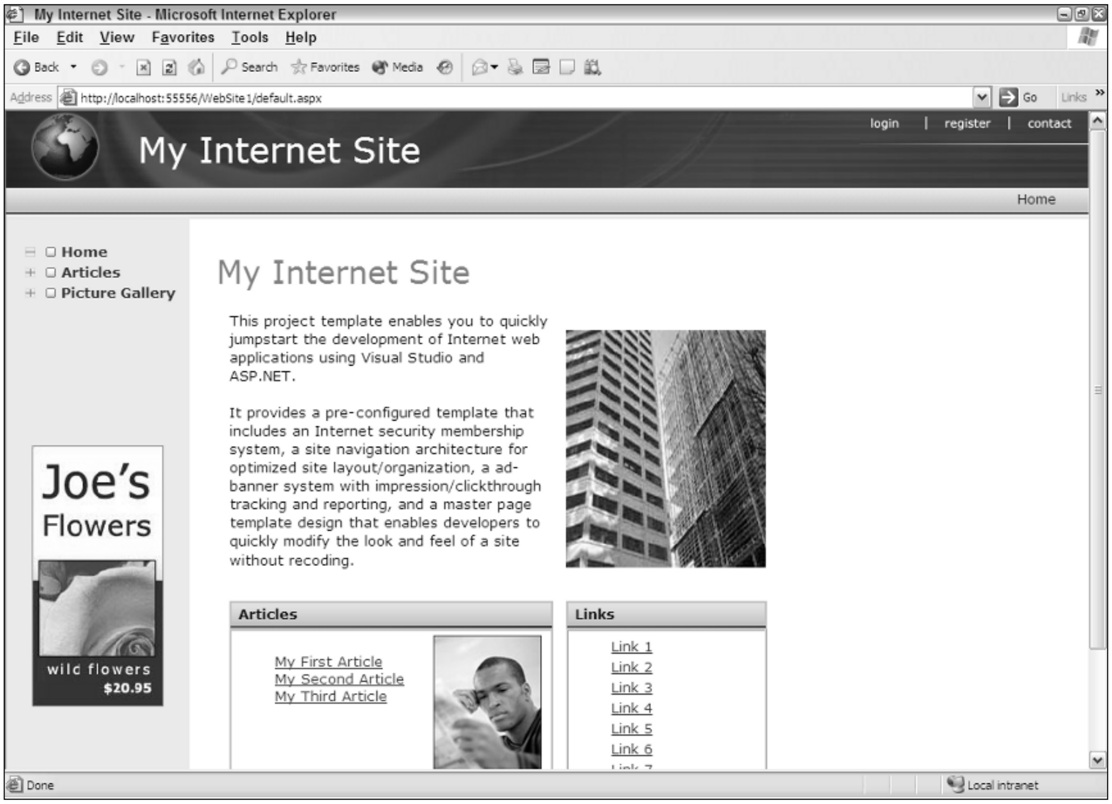
The ROI of .NET. An excellent presentation that il...

ASP.NET Community Starter Kit (VBVS) - The Microso...

Quizzard is an open source application created by ...

Figure 1-4

Because of the popularity of frameworks such as IBuySpy, ASP.NET 2.0 offers built-in capability for using Web Parts to easily build portals. The possibilities for what you can build using the new Portal Framework is astounding. The power of building using Web Parts is that it easily enables end users to completely customize the portal for their own preferences. Figure 1-5 shows an example application built using Web Parts.



My Internet Site

This project template enables you to quickly jumpstart the development of Internet web applications using Visual Studio and ASP.NET.

It provides a pre-configured template that includes an Internet security membership system, a site navigation architecture for optimized site layout/organization, a ad-banner system with impression/clickthrough tracking and reporting, and a master page template design that enables developers to quickly modify the look and feel of a site without recoding.

Figure 1-5

Site Navigation

The ASP.NET team members realize that end users want to navigate through applications with ease. The mechanics to make this work in a logical manner are sometimes hard to code. The team solved the problem in ASP.NET 2.0 with a series of navigation-based server controls.

First, you can build a site map for your application in an XML file that specific controls can inherently work from. Listing 1-4 shows a sample site map file.

Listing 1-4: An example of a site map file

```
<?xml version="1.0" encoding="utf-8" ?>

<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
   <siteMapNode title="Home" description="Home Page" url="default.aspx">
      <siteMapNode title="News" description="The Latest News" url="News.aspx">
         <siteMapNode title="U.S." description="U.S. News"
          url="News.aspx?cat=us" />
         <siteMapNode title="World" description="World News"
```

```
             url="News.aspx?cat=world" />
            <siteMapNode title="Technology" description="Technology News"
             url="News.aspx?cat=tech" />
            <siteMapNode title="Sports" description="Sports News"
             url="News.aspx?cat=sport" />
         </siteMapNode>
         <siteMapNode title="Finance" description="The Latest Financial Information"
           url="Finance.aspx">
            <siteMapNode title="Quotes" description="Get the Latest Quotes"
             url="Quotes.aspx" />
            <siteMapNode title="Markets" description="The Latest Market Information"
             url="Markets.aspx">
               <siteMapNode title="U.S. Market Report"
                description="Looking at the U.S. Market" url="MarketsUS.aspx" />
               <siteMapNode title="NYSE"
                description="The New York Stock Exchange" url="NYSE.aspx" />
            </siteMapNode>
            <siteMapNode title="Funds" description="Mutual Funds"
             url="Funds.aspx" />
         </siteMapNode>
         <siteMapNode title="Weather" description="The Latest Weather"
          url="Weather.aspx" />
      </siteMapNode>
   </siteMap>
```

After you have a site map in place, you can use this file as the data source behind a couple of new site navigation server controls, such as the TreeView and the SiteMapPath server controls. The TreeView server control enables you to place an expandable site navigation system in your application. Figure 1-6 shows you an example of one of the many looks you can give the TreeView server control.

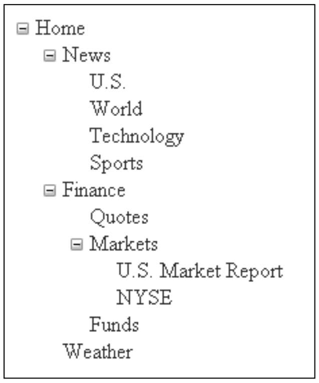


Figure 1-6

The SiteMapPath is a control that provides the capability to place what some call *breadcrumb navigation* in your application so that the end user can see the path that he has taken in the application and can easily navigate to higher levels in the tree. Figure 1-7 shows you an example of the SiteMapPath server control at work.

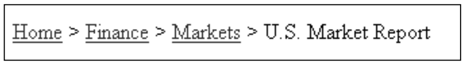

Home > Finance > Markets > U.S. Market Report

Figure 1-7

These new site navigation capabilities provide a great way to get programmatic access to the site layout and even to take into account things like end-user roles to determine which parts of the site to show.

New Compilation System

In ASP.NET 2.0, the code is constructed and compiled in a new way. Compilation in ASP.NET 1.0 was always a tricky scenario. With ASP.NET 1.0, you could build an application's code-behind files using ASP.NET and Visual Studio, deploy it, and then watch as the `.aspx` files were compiled page by page as each page was requested. If you made any changes to the code-behind file in ASP.NET 1.0, it was not reflected in your application until the entire application was rebuilt. That meant that the same page-by-page request had to be done again before the entire application was recompiled.

Everything about how ASP.NET 1.0 worked with classes and compilation changed with the release of ASP.NET 2.0. The mechanics of the new compilation system actually begin with how a page is structured in ASP.NET 2.0. In ASP.NET 1.0, you either constructed your pages using the code-behind model or by placing all the server code inline between `<script>` tags on your `.aspx` page. Most pages were constructed using the code-behind model because this was the default when using Visual Studio .NET 2002 or 2003. It was quite difficult to create your page using the inline style in these IDEs. If you did, you were deprived of the use of IntelliSense, which can be quite the lifesaver when working with the tremendously large collection of classes that the .NET Framework offers.

ASP.NET 2.0 offers a new code-behind model because the .NET Framework 2.0 offers the capability to work with *partial classes* (also called partial types). Upon compilation, the separate files are combined into a single offering. This gives you much cleaner code-behind pages. The code that was part of the `Web Form Designer Generated` section of your classes is separated from the code-behind classes that you create yourself. Contrast this with the ASP.NET 1.0 `.aspx` file's need to derive from its own code-behind file to represent a single logical page.

ASP.NET 2.0 applications can include an `\App_Code` directory where you place your class's source. Any class placed here is dynamically compiled and reflected in the application. You do not use a separate build process when you make changes as you did with ASP.NET 1.0. This is a *just save and hit* deployment model like the one in classic ASP 3.0. Visual Studio Web Developer also automatically provides IntelliSense for any objects that are placed in the `\App_Code` directory, whether you are working with the code-behind model or are coding inline.

ASP.NET 2.0 also provides you with tools that enable you to precompile your ASP.NET applications — both `.aspx` pages and code behind — so that no page within your application has latency when it is retrieved for the first time. Doing this is also a great way to discover any errors in the pages without invoking every page. Precompiling your ASP.NET 2.0 applications is as simple as using `aspnet_compiler.exe` and employing some of the available flags. As you precompile your entire application, you also receive error notifications if any errors are found anywhere within it. Precompilation also enables you to deliver only the created assembly to the deployment server, thereby protecting your code from snooping, unwanted changes, and tampering after deployment. You see examples of both of these scenarios later in this book.

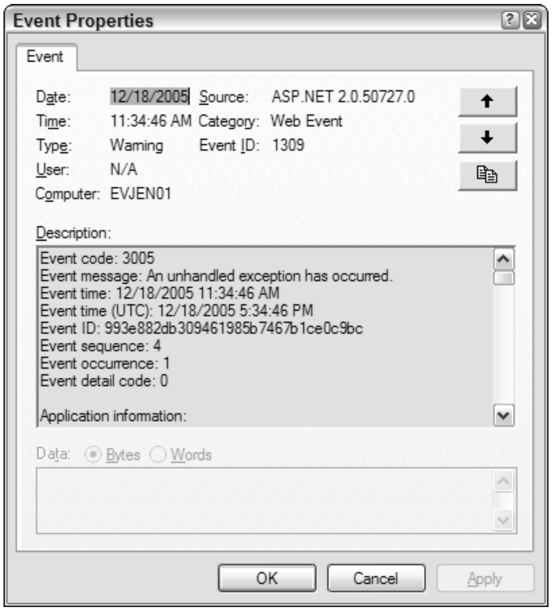
Health Monitoring for Your ASP.NET Applications

The built-in health monitoring capabilities are new and rather significant features designed to make it easier to manage a deployed ASP.NET application. Health monitoring provides what the term implies — the capability to monitor the health and performance of your deployed ASP.NET applications.

ASP.NET health monitoring is built around various health monitoring events (which are referred to as *Web events*) occurring in your application. Using the new health monitoring system enables you to perform event logging for Web events such as failed logins, application starts and stops, or any unhandled exceptions. The event logging can occur in more than one place; therefore, you can log to the event log or even back to a database. In addition to performing this disk-based logging, you can also use the system to e-mail health monitoring information.

Besides working with specific events in your application, you can also use the health monitoring system to take health snapshots of a running application. As you can with most systems that are built into ASP.NET 2.0, you are able to extend the health monitoring system and create your own events for recording application information.

Health monitoring is already enabled by default in the system `.config` files. The default setup for health monitoring logs all errors and failure audits to the event log. For instance, throwing an error in your application results in an error notification in the Application log. An example of this is presented in Figure 1-8.



Date: 12/18/2005 Source: ASP.NET 2.0.50727.0
Time: 11:34:46 AM Category: Web Event
Type: Warning Event ID: 1309
User: N/A
Computer: EVJEN01

Description:

```
Event code: 3005
Event message: An unhandled exception has occurred.
Event time: 12/18/2005 11:34:46 AM
Event time (UTC): 12/18/2005 5:34:46 PM
Event ID: 993e882db309461985b7467b1ce0c9bc
Event sequence: 4
Event occurrence: 1
Event detail code: 0

Application information:
```

Data: (● Bytes) ○ Words

Figure 1-8

You can change the default event logging behaviors simply by making some minor changes to your application's `web.config` file. For instance, suppose that you want to store this error event information in a SQL Express file contained within the application. This change can be made by adding a `<healthMonitoring>` node to your `web.config` file as presented in Listing 1-5.

Listing 1-5: Defining health monitoring in the web.config file

```
<healthMonitoring enabled="true">
   <providers>
      <clear />
      <add name="SqlWebEventProvider" connectionStringName="LocalSqlServer"
       maxEventDetailsLength="1073741823" buffer="false" bufferMode="Notification"
       type="System.Web.Management.SqlWebEventProvider,
          System.Web,Version=2.0.0.0,Culture=neutral,
          PublicKeyToken=b03f5f7f11d50a3a"/>
   </providers>
   <rules>
      <clear />
      <add name="All Errors Default" eventName="All Errors"
       provider="SqlWebEventProvider"
       profile="Default" minInstances="1" maxLimit="Infinite"
       minInterval="00:01:00" custom="" />
      <add name="Failure Audits Default" eventName="Failure Audits"
       provider="SqlWebEventProvider" profile="Default" minInstances="1"
       maxLimit="Infinite" minInterval="00:01:00" custom="" />
   </rules>
</healthMonitoring>
```

After this change, events are logged in the `ASPNETDB.MDF` file which is automatically created for you on your behalf if it doesn't already exist in your project.

Opening up this SQL Express file, you find an `aspnet_WebEvent_Events` table. An example of this table with a few entries is presented in Figure 1-9.

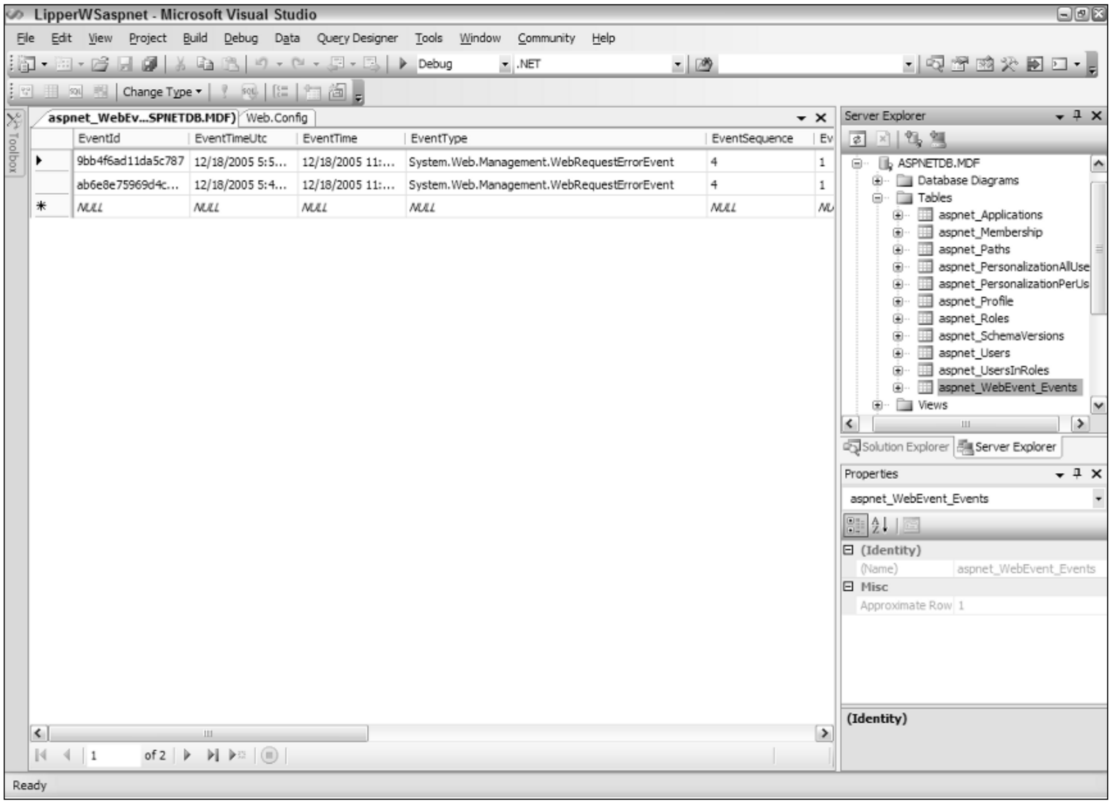
You will learn much more about the health monitoring capabilities provided with ASP.NET 2.0 in Chapter 32.

Reading and Writing Configuration Settings

Using the `WebConfigurationManager` class, you have the capability to read and write to the server or application configuration files. This means that you can write and read settings in the `machine.config` or the `web.config` files that your application uses.

The capability to read and write to configuration files is not limited to working with the local machine in which your application resides. You can also perform these operations on remote servers and applications.

Of course, a GUI-way exists in which you can perform these read or change operations on the configuration files at your disposal. The exciting thing, however, is that the built-in GUI tools that provide this functionality (such as the new ASP.NET MMC snap-in) use the `WebConfigurationManager` class that is also available for building custom administration tools.



	EventId	EventTimeUtc	EventTime	EventType	EventSequence	Ev
▶	9bb4f6ad11da5c787	12/18/2005 5:5...	12/18/2005 11:...	System.Web.Management.WebRequestErrorEvent	4	1
	ab6e8e75969d4c...	12/18/2005 5:4...	12/18/2005 11:...	System.Web.Management.WebRequestErrorEvent	4	1
*	NULL	NULL	NULL	NULL	NULL	NU

(Name)	aspnet_WebEvent_Events
Approximate Row	1

Figure 1-9

Listing 1-6 shows an example of reading a connection string from an application's `Web.config` file.

Listing 1-6: Reading a connection string from the applicationÕs Web.config file

VB

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
   Try
      Dim connectionString As String = _
         ConfigurationManager.ConnectionStrings("Northwind").
            ConnectionString.ToString()
      Label1.Text = connectionString
   Catch ex As Exception
      Label1.Text = "No connection string found."
   End Try
End Sub
```

(continued)

Listing 1-6: *(continued)*

C#

```
protected void Page_Load(object sender, EventArgs e)
{
   try
   {
      string connectionString =
         ConfigurationManager.ConnectionStrings["Northwind"].
            ConnectionString.ToString();
      Label1.Text = connectionString;
   }
   catch (Exception)
   {
      Label1.Text = "No connection string found.";
   }
}
```

This little bit of code writes the Northwind connection string found in the `web.config` file to the screen using a Label control. As you can see, it is rather simple to grab items from the configuration file.

Localization

ASP.NET 2.0 improves upon the previous versions of ASP.NET by making it easier to localize applications. In addition to using Visual Studio, you can create resource files (`.resx`) that allow you to dynamically change the pages you create based upon the culture settings of the requestor.

ASP.NET 2.0 also provides the capability to provide resources application-wide or just to particular pages in your application through the use of two new application folders — App_GlobalResources and App_LocalResources.

The items defined in any `.resx` files you create are then accessible directly in the ASP.NET server controls or programmatically using expressions such as:

```
<%= Resources.Resource.Question %>
```

This new system is straightforward and simple to implement. This topic is covered in greater detail in Chapter 30.

Additions to the Page Framework

The ASP.NET page framework has some dramatic new additions that you can include in your applications. One of the most striking ones is the capability to build ASP.NET pages based upon visual inheritance. This was possible in the Windows Forms world, but it was harder to achieve with ASP.NET. You also gain the capability to easily apply a consistent look and feel to the pages of your application by using themes. Many of the difficulties in working with ADO.NET in the past have now been removed with the addition of a new series of data source controls that take care of accessing and retrieving data from a large collection of data stores. Although these are not the only new controls, the many new server controls create a larger ASP.NET page framework.

Master Pages

With the introduction of *master pages* in ASP.NET 2.0, you can now use visual inheritance within your ASP.NET applications. Because many ASP.NET applications have a similar structure throughout their pages, it is logical to build a page template once and use that same template throughout the application.

In ASP.NET 2.0, you do this by creating a `.master` page, as shown in Figure 1-10.

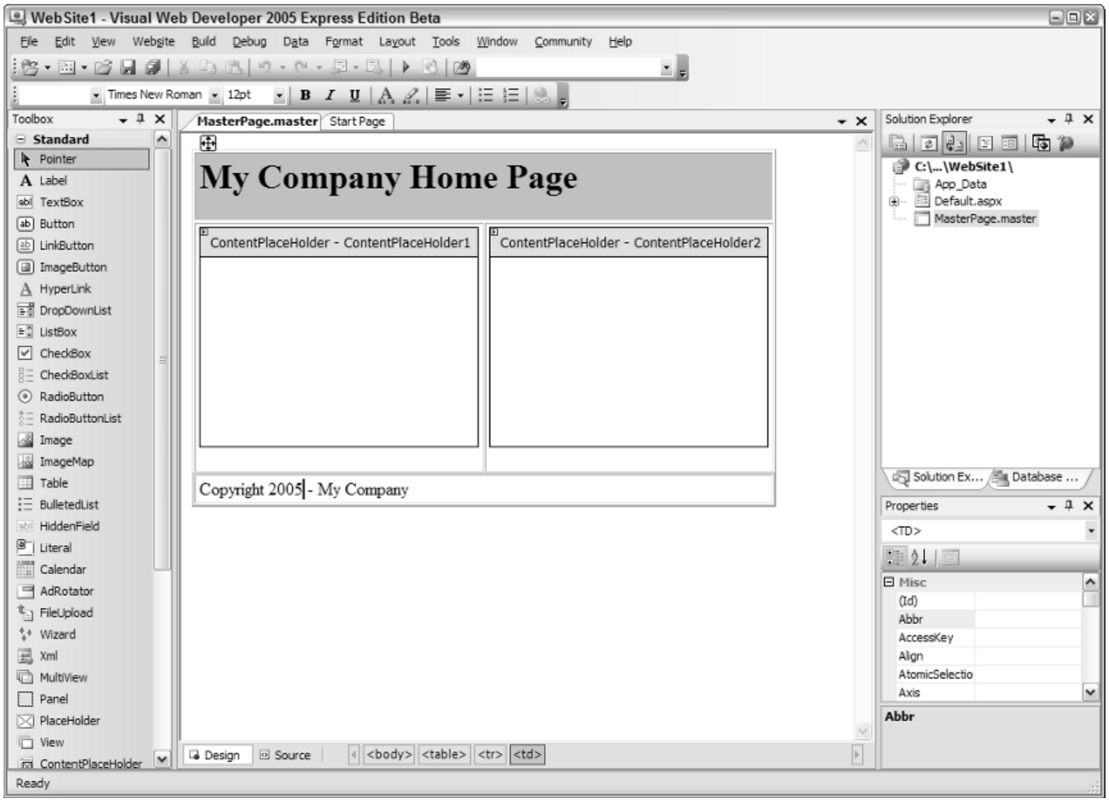
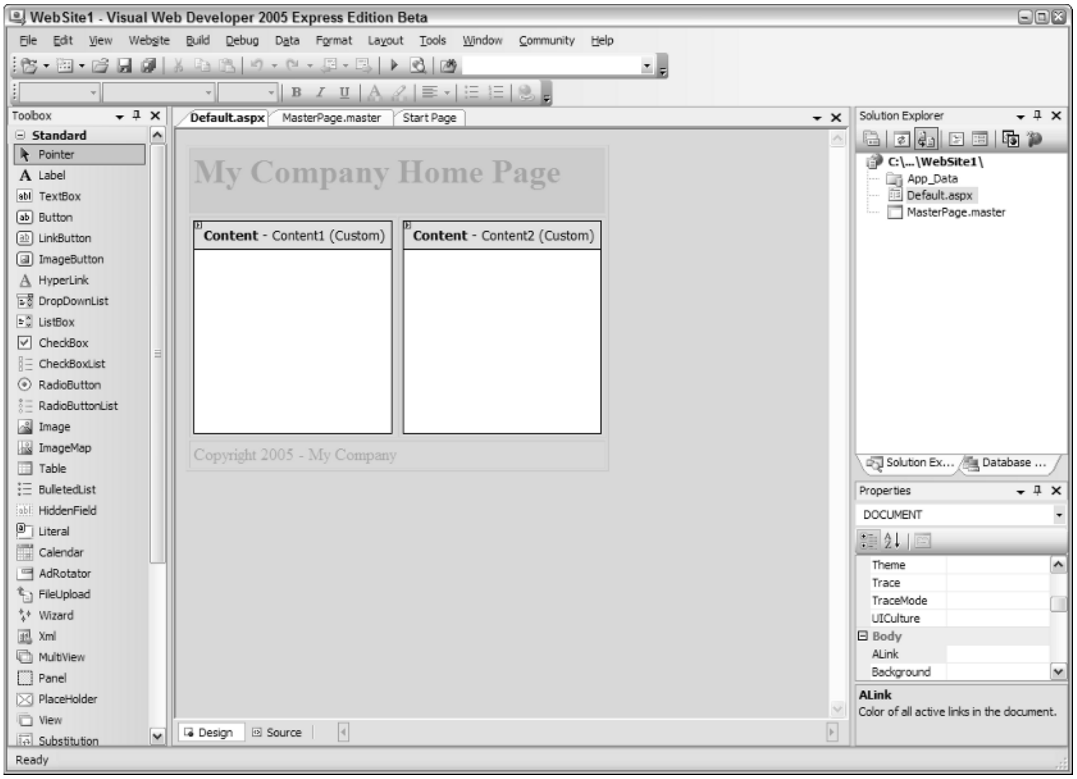


Figure 1-10

An example master page might include a header, footer, and any other elements that all the pages can share. Besides these core elements, which you might want on every page that inherits and uses this template, you can place `<asp:ContentPlaceHolder>` server controls within the master page itself for the subpages (or content pages) to use in order to change specific regions of the master page template. The editing of the subpage is shown in Figure 1-11.

When an end user invokes one of the subpages, he is actually looking at a single page compiled from both the subpage and the master page that the particular subpage inherited from. This also means that the server and client code from both pages are enabled on the new single page.

The nice thing about master pages is that you now have a single place to make any changes that affect the entire site. This eliminates making changes to each and every page within an application.



ALink
Color of all active links in the document.

Figure 1-11

Themes

The introduction of themes in ASP.NET 2.0 has made it quite simple to provide a consistent look and feel across your entire site. Themes are simple text files where you define the appearance of server controls that can be applied across the site, to a single page, or to a specific server control. You can also easily incorporate graphics and Cascading Style Sheets, in addition to server control definitions.

Themes are stored in the `/App_Theme` directory within the application root for use within that particular application. One cool capability of themes is that you can dynamically apply them based on settings that use the new personalization service provided by ASP.NET 2.0. Each unique user of your portal or application can have her own personalized look and feel that she has chosen from your offerings.

New Objects for Accessing Data

One of the more code-intensive tasks in ASP.NET 1.0 was the retrieval of data. In many cases, this meant working with a number of objects. If you have been working with ASP.NET for a while, you know that it was an involved process to display data from a Microsoft SQL Server table within a DataGrid server control. For instance, you first had to create a number of new objects. They included a `SqlConnection` object followed by a `SqlCommand` object. When those objects were in place, you then created a `SqlDataReader` to populate your DataGrid by binding the result to the DataGrid. In the end, a table appeared containing the contents of the data you were retrieving (such as the Customers table from the Northwind database).

ASP.NET 2.0 eliminates this intensive procedure with the introduction of a new set of objects that work specifically with data access and retrieval. These new data controls are so easy to use that you access and retrieve data to populate your ASP.NET server controls without writing any code. You saw an example of this in Listing 1-2, where an `<asp:SqlDataSource>` server control retrieved rows of data from the Customers table in the Northwind database from SQL Server. This SqlDataSource server control was then bound to the new GridView server control via the use of simple attributes within the GridView control itself. It really couldn't be any easier!

The great news about this new functionality is that it is not limited to just Microsoft's SQL Server. In fact, several data source server controls are at your disposal. You also have the capability to create your own. In addition to the SqlDataSource server control, ASP.NET 2.0 introduces the AccessDataSource, XmlDataSource, ObjectDataSource, and SiteMapDataSource server controls. You use all these new data controls later in this book.

New Server Controls

So far, you have seen a number of new server controls that you can use when building your ASP.NET 2.0 pages. For example, the preceding section talked about all the new data source server controls that you can use to access different kinds of data stores. You also saw the use of the new GridView server control, which is an enhanced version of the previous DataGrid control that you used in ASP.NET 1.0.

Besides the controls presented thus far in this chapter, ASP.NET 2.0 provides more than 50 additional new server controls! In fact, so many new server controls have been introduced that the next IDE for building ASP.NET applications, Visual Studio 2005, had to reorganize the Toolbox where all the server controls are stored. They are now separated into categories instead of being displayed in a straight listing as they were in Visual Studio .NET or the ASP.NET Web Matrix. The new Visual Studio 2005 Toolbox is shown in Figure 1-12.

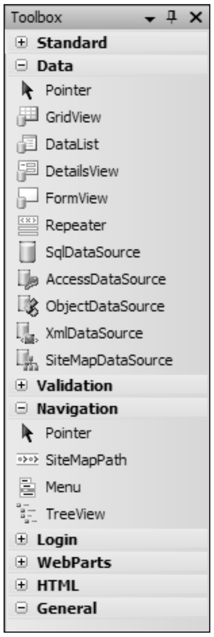
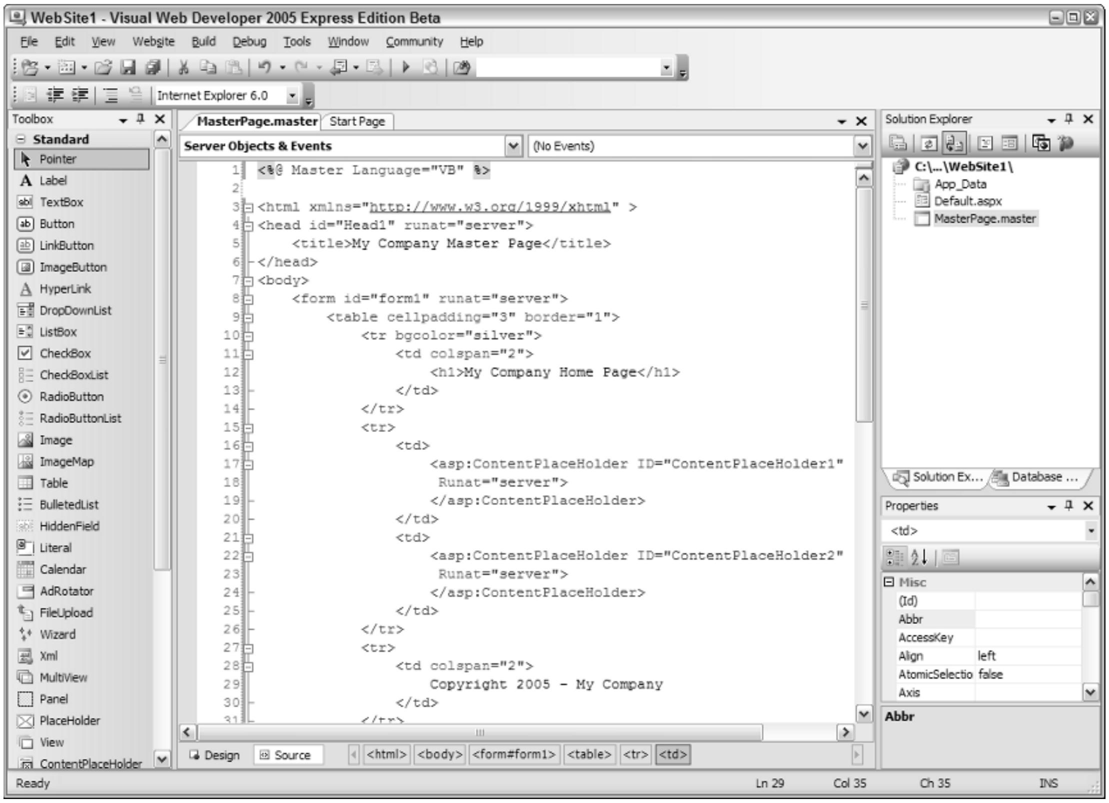


Figure 1-12

A New IDE for Building ASP.NET 2.0 Pages

With ASP.NET 1.0/1.1, you can build your ASP.NET application using Notepad, Visual Studio .NET 2002 and 2003, as well as the hobbyist-focused ASP.NET Web Matrix. ASP.NET 2.0 comes with another IDE to the Visual Studio family — Visual Studio 2005.

Visual Studio 2005 offers some dramatic enhancements that completely change the way in which you build your ASP.NET applications. Figure 1-13 shows you a screen shot of the new Visual Studio 2005.



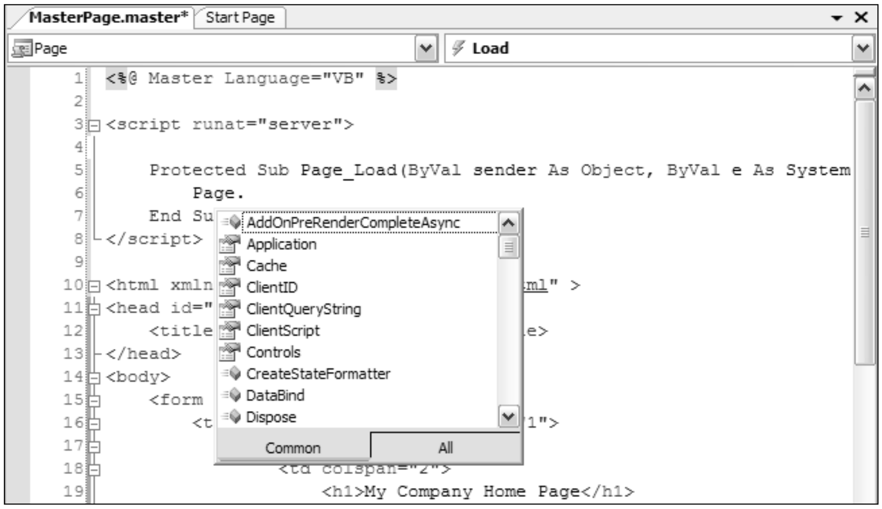
```
<%@ Master Language="VB" %>

<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>My Company Master Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <table cellpadding="3" border="1">
            <tr bgcolor="silver">
                <td colspan="2">
                    <h1>My Company Home Page</h1>
                </td>
            </tr>
            <tr>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
                     Runat="server">
                    </asp:ContentPlaceHolder>
                </td>
                <td>
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder2"
                     Runat="server">
                    </asp:ContentPlaceHolder>
                </td>
            </tr>
            <tr>
                <td colspan="2">
                    Copyright 2005 - My Company
                </td>
            </tr>
```

(Id)	
Abbr	
AccessKey	
Align	left
AtomicSelectio	false
Axis	

Figure 1-13

One rather dramatic change to the IDE is that Visual Studio 2005 builds applications using a file-based system, not the project-based system used by Visual Studio .NET. When using Visual Studio .NET in the past, you had to create new projects (for example, an ASP.NET Web Application project). This process created a number of project files in your application in addition to the `Default.aspx` page. Because everything was based on a singular project, it became very difficult to develop applications in a team environment.

Web projects in Visual Studio 2005, on the other hand, are based on a file system approach. No project files are included in your project, and this makes it very easy for multiple developers to work on a single application together without bumping into each other. Other changes are those to the compilation system discussed earlier. You can now build your ASP.NET pages using the inline model or the new code-behind model. Whether you build pages inline or with the new code-behind model, you have full IntelliSense capabilities. This, in itself, is powerful and innovative. Figure 1-14 shows IntelliSense running from an ASP.NET page that is being built using the inline model.



```
<%@ Master Language="VB" %>

<script runat="server">

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System
        Page.
    End Su
</script>

<html xmln                                             ml" >
<head id="
    <title                                             e>
</head>
<body>
    <form
        <t                                             1">

                <td colspan="2">
                    <h1>My Company Home Page</h1>
```

Figure 1-14

Another feature of Visual Studio 2005 that has come over from the ASP.NET Web Matrix is that you don't need IIS on your development machine. Visual Studio 2005 has a built-in Web server that enables you to launch pages from any folder in your system with relative ease. Chapter 2 discusses the new Visual Studio 2005 in detail.

Summary

This whirlwind tour briefly introduced some of the new features in ASP.NET 2.0. This release offers so much that we can't come close to covering it all in this chapter. The new ways of working with data and presentation and the new infrastructure provide effective means to create powerful and secure applications. But this book also gets down and dirty in the underlying architecture and features that have been included in ASP.NET since it was initially released.

ASP.NET 2.0 is so powerful and has so much capability built in that its tremendous benefits to productivity really shine through. Pull up your keyboard and have some fun as you take the journey through this book and this powerful technology.