Microsoft

2005 EDITION

MICROSOFT®

ASP.NET 2.0 由入门到精通

Step by Step

George Shepherd

China Edition
Published by Shanghai World Publishing Corporation

世界图书出版公司

2

Microsoft

Microsoft® ASP.NET 2.0 Step By Step

George Shepherd

图书在版编目(CIP)数据

Microsoft ASP. NET 2.0:由入门到精通:英文/(美)史蒂弗雷德著.
—上海:上海世界图书出版公司,2007.6

ISBN 978-7-5062-8334-2

I. M… Ⅱ. 史… Ⅲ. 主页制作-程序设计-英文 Ⅳ. TP393.092

中国版本图书馆 CIP 数据核字(2007)第 083239 号

Microsoft ASP. NET 2.0: **由入门到精通**

[美]乔治·史蒂弗雷德 著

上海世界图书出版公司 重印发行

上海市尚文路 185 号 B 楼

邮政编码 200010

(公司电话:021-63783016 转发行部)

上海市堡港印刷厂印刷

如发现印装质量问题,请与印刷厂联系

(质检科电话:021-68121468)

各地新华书店经销

开本:787×960 1/16 印张:28.25 字数:847 000

2007 年 6 月第 1 版 2007 年 6 月第 1 次印刷

ISBN 978-7-5062-8334-2/T·159

图字:09-2007-443 号

定价:188.00 元

http://www.wpcsh.com

Dedicated to

Ted Gregory Daston Shepherd

Introduction

This book will help you figure out how to write Web applications using Microsoft's most current version of its HTTP request processing framework—ASP.NET 2.0. Web development has come a long way since the earliest sites began popping up in the early 1990s. The world of Web development offers several different choices as far as development tools go. Over the past few years, ASP.NET has evolved to become one of the most consistent, stable, and feature-rich frameworks available for managing HTTP requests.

ASP.NET together with Visual Studio include a number of features to make your life as a Web developer easier. For example, Visual Studio starts you off with several very useful project templates from which to develop your site. Visual Studio also supports a number of development modes, including using Internet Information Services directly to test your site during development, using a built-in Web server, or developing your site over an FTP connection. The debugger in Visual Studio lets you run the site and step through the critical areas of your code to find problems. The Visual Studio designer enables effective user interface development, allowing you to drop control elements onto a canvas to see how they appear visually. These are but a few of the features built into the ASP.NET framework when paired with Visual Studio.

While ASP.NET and Visual Studio offer excellent tools for writing Web applications, Web development on the Microsoft platform hasn't always been this way. The road to ASP.NET 2.0 has been nearly a decade in the making.

The Road to ASP.NET 2.0

Until about 1993, there were very few Web servers in the world. Most of these earliest Web servers lived at universities or other research centers. In the early 1990s, the number of Web sites available began to increase dramatically. If you used the Web back in the early 1990s, chances are you probably came across little more than some HTML pages put together by the earliest Web site pioneers or some photo collections represented by links to GIF or JPEG files. Back then, there was no Google, no Yahoo, and certainly no MSN Search. The only way you could get to someone's site was if you either knew the site's Uniform Resource Locator (URL) or were referred to it through someone else's page.

Typing a URL like this:

```
http://www.somesite.com
```

into a browser's navigation window sent your request through a maze of routers, finally appearing at a server somewhere. The earliest Web servers lived on UNIX boxes. They performed the simple job of loading the HTML file and sending it back to the requestor (perhaps a browser such as Mosaic).

The advent of the Common Gateway Interface (CGI) introduced a standard way to interface with browsers to produce interactive Web applications. While a Web server that serves up plain, static HTML documents is useful in certain contexts (for example, a hyperlinked dictionary), more complex applications require a conversation between the user and end server.

That's where CGI comes in. With the help of HTML tags representing standard GUI controls, CGI applications can respond to requests dynamically. That is, CGI applications vary their output depending upon the state within the request and the application, paving the way for widely interactive applications. For example, a CGI application can examine an incoming request and determine the user is looking for a certain piece of information (perhaps a product code). The CGI application can perform a database lookup for the product and shoot some HTML that describes the product back to the client.

When it became clear that the Web was an important aspect of information technology, Microsoft entered the fray by introducing the Internet Services API (ISAPI) and a program to listen for HTTP requests: Internet Information Services (IIS). While the first UNIX Web servers started a new process to handle each HTTP new request (in keeping with the classical UNIX model), that model is very expensive. The Microsoft Web strategy is based on DLLs. It's much faster to load a DLL to respond to an HTTP request than it is to start a whole new process.

When programming to the Microsoft platform, IIS listens to port 80 for HTTP requests. IIS handles some requests directly, while delegating other requests to specific ISAPI extension DLLs to execute the request. In other cases, IIS will map a file extension to a specific ISAPI DLL. A number of ISAPI DLLs come preinstalled with Windows. However, IIS is extensible, and you may map different extensions to any ISAPI DLL—even one you wrote. To make a Web site work using IIS and ISAPI, developers employ ISAPI DLLs. These DLLs intercept the request, decompose it, and respond by sending back something to the client (usually some HTML).

While the IIS/ISAPI platform represents a very flexible and functional way to create Web applications, it's not without its downside. Specifically, ISAPI DLLs are traditionally written in C++ and are subject to the pitfalls of C++ programming (including such foibles as de-referencing bad pointers, forgetting to free memory, and traditionally lengthy development cycles). The other problem with ISAPI DLLs is that it's becoming increasingly more difficult to find C++ programmers. Enter Active Server Pages, or classic ASP.

Classic ASP

In an effort to make Web development more accessible on the Microsoft platform, Microsoft introduced Active Server Pages (ASP). The idea behind classic ASP is that a single ISAPI DLL named ASP.DLL interprets files with the extension ASP (for example, MYSITE.asp). ASP files include some HTML and perhaps some script code to be executed on the server. The ASP ISAPI DLL executes the script code as necessary and sends the HTML contained in the ASP file back to the client. The script code usually calls COM objects that do the dirty work (for example, looking up items in a database and tailoring the output based upon its findings) while the look and feel of the page is defined by the HTML in the ASP file.

While ASP opened the doors to a whole host of new programmers by catering to a much more widely used programming language (Visual Basic and VBScript), it wasn't the silver bullet. Among the downsides of classic ASP are:

- Mixing of user interface code and programming logic
- Performance issues due to *IDispatch*
- Inconsistent means of managing state (session state and application state)
- An ad-hoc security model

This isn't an exhaustive list by any means, but it highlights the most important issues with classic ASP. That's why ASP.NET exists.

ASP.NET 1.0 and 1.1

Microsoft's .NET framework introduces a whole new way of programming the Microsoft platform. Microsoft developers are primarily concerned with threads and memory (that's basically the API programming model). This model carried over to all areas of development, including Web development, placing a heavy burden upon programmers.

.NET is built upon the notion of managed types. Developers writing classic Windows code (and Web code) wrote classes using C++ or Visual Basic. In many ways, types are similar to the notion of the C++ class in that types are units of state with functionality attached to them. However, the similarity ends there. Whereas it was incumbent upon the developer to manage instances of classes, types are managed completely by the .NET runtime services–the Common Language Runtime (CLR). Because the CLR takes over managing memory and threads, developers are much more at liberty to concentrate on the actual application (rather than chasing down errant pointers, memory leaks, and unexplained crashes).

ASP.NET introduces runtime services and a well-engineered class library for greatly enhancing Web development. In a way, classic ASP was sort of "taped onto" the IIS/ISAPI architecture without any real organic thought as to how early design decisions would affect developers later on. Well, now it's later on and classic ASP.NET's warts have become fairly obvious.

ASP.NET is built from the ground up to be an extensible, feature-rich way to handle HTTP requests. ASP.NET leverages IIS in that requests for ASP.NET services are mapped to an ISAPI DLL. The DLL is named ASPNET_ISAPI.DLL. From there, processing is passed into a worker process provided by ASP.NET (ASPNET_WP.EXE in IIS 5 or W3WP.EXE in IIS 6). The fundamental request processing is handled by managed types within the worker process. Control passes between a number of classes plugged into the pipeline–some provided by Microsoft and/or third parties, and some provided by the developer. What's more, ASP.NET is built from the ground up to be a comprehensive framework for writing Web applications. All the parts of the framework execute together to handle requests. By contrast, classic ASP.NET script code had no structure to it, and code logic within applications tended to be ad hoc.

ASP.NET 1.0 and 1.1 provided a significant number of features, including:

- An object-oriented framework for defining applications
- Separation of user interface declarations (HTML) and application logic
- Compiled code for executing application logic
- Configurable session state management
- Built-in data caching
- Built-in content caching
- A well-defined UI componentization architecture
- High-level components for managing data formatting (grids, lists, text boxes)
- Built-in program tracing and diagnostics
- Built-in user input validation
- An easy-to-use custom authentication mechanism
- Solid integration with ADO.NET (the .NET database story)
- Excellent support for Web Services
- Zero reliance on the Component Object Model
- An extensible pipeline with many places in which a request can be intercepted

ASP.NET 1.0 set the stage for many developers both moving into Web development and moving to the Microsoft Platform.

ASP.NET 2.0

Which brings us to ASP.NET 2.0. ASP.NET 2.0 builds upon ASP.NET 1.0 and 1.1 by providing a number of new features in addition to what already existed with ASP.NET 1.0. These features include

- Master Pages and Skins
- Declarative databinding
- Provider pattern model
- New cache features

- Membership controls
- Personalization controls
- Support for Web Parts
- Programmable configuration
- Administration tools
- New compilation model

All the features of ASP.NET 1.0/1.1 are still there. However, these new features make ASP.NET an even more compelling platform for creating Web sites. We'll visit all these features as we tour ASP.NET 2.0.

A Word About the .NET Runtime

ASP.NET 2.0 is built upon Microsoft's Common Language Runtime. In its earliest days, programming Windows involved interacting with the operating system at a very intimate level. For example, getting a Window to show up on a screen took many lines of C code. In addition, Windows included a rudimentary component technology–raw Dynamic Link Libraries. Dynamic Link Libraries (DLLs) represent a necessary technology to enable composing systems dynamically–that is, to assemble applications from several disparate *binary* components. However, DLLs by themselves are not sufficient for composing systems reliably–primarily because it's very difficult to manage multiple versions of a component (a DLL).

During the mid 90's, the Component Object Model (COM) emerged as a way to help manage multiple versions of a component. By stipulating strict rules about how clients and components may interact, COM represented a technology sufficient for composing applications from different binary components. However, COM faced a few dead ends which became apparent as developers began building larger systems.

First, COM relied on humans following rules to get things to interoperate. For example, COM stipulates a rule that once a programmatic interface is published, it must never change. Changing a published COM interface after clients begin coding against it will almost certainly bring a system to its knees. In addition, COM relied on sometimes obtuse rules as far as managing resources. However, the *coup de grace* for COM was probably the disparate type systems involved. That is, COM represented many data types differently for three separate classes of developers: C++ developers, Visual Basic developers, and scripting developers. The different data type systems made it extremely inconvenient to build systems built from different languages. It could be done, but developers had to be very wary when making calls across such component boundaries.

.NET and the Common Language Runtime (the CLR) were developed to solve the dead ends appearing in COM near the end of the last century. When you choose to buy into the .NET runtime, it's like putting your code in a nice hotel room when it runs. For example, the .NET runtime loads and manages your code as it runs. Pure memory leaks are a thing of the past because the runtime collects garbage when necessary. The problem of overrunning array boundaries disappears because the .NET runtime keeps careful watch over memory and knows when anything is out of place. In addition, the .NET runtime includes a new security model making it more difficult to hack into .NET-based software. Finally, the .NET runtime introduces a new packaging and deployment model, .NET Assemblies, which helps enforce versioning components.

ASP.NET is founded on the .NET runtime. As we'll see in the following chapters, ASP.NET runs completely under the auspices of the CLR. After IIS hands an HTTP request off to ASP.NET, it runs through the ASP.NET pipeline. The request may be intercepted at various places along the way, and you have ample opportunity to interrogate the request and modify the response before it finally leaves the ASP.NET runtime. Gone is the COM layer between the HTTP request processing machinery and a business's domain objects. Domain objects running under .NET can be linked into the request processing pipeline for high performance and tight security. In addition, because all .NET components agree upon the data types being passed between them, there are no more bizarre data conversions (as there used to be in classic ASP).

In the process of building ASP.NET applications you will be developing .NET assemblies—most of the time implicitly, but sometimes explicitly. While you'll be focusing on ASP.NET as a Web application framework, you'll develop a strong familiarity with the .NET runtime as well. Very often, the classes you use in an ASP.NET application are the same or very similar to those you'd use in a console application, a Windows application, or even a component library.

Using This Book

The purpose of this book is to weave the story of ASP.NET 2.0 development for you. Each section presents a specific ASP.NET feature in a digestible format *with* examples. The step-wise instructions should yield working results for you immediately. You'll find most of the main features within ASP.NET illustrated here with succinct, easily duplicated examples. I made the examples rich to illustrate the feature without being overbearing. In addition to showing off ASP.NET features by example, you'll find practical applications of each feature so you can take these techniques into the real world.

Who Is This Book For?

This book is targeted to several developers:

- **Those starting out completely new with ASP.NET** The text includes enough back story to explain the Web development saga even if you've developed only desktop applications.

- **Those migrating from either ASP.NET 1.x or even classic ASP** The text explains how ASP.NET 2.0 is different from ASP.NET 1.*x*. The text also includes references explaining differences between ASP.NET and classic ASP.
- **Those wanting to consume ASP.NET how-to knowledge in digestible pieces** Most chapters stand independently. You don't have to read the chapters in any particular order to find the book valuable. Each chapter stands more or less on its own (with the exception of the first chapter detailing the fundamentals of Web applications–you may want to read it first if you've never ventured beyond desktop application development). You may find it useful to study the chapters about server-side controls together (Chapters 3, 4, and 5), but it's not completely necessary to do so.

Organization of This Book

This book is organized so that each chapter may be read independently, for the most part. With the exception of Chapter 1 about Web application essentials and the three server-side control chapters–Chapters 3, 4, and 5–which make sense to tackle together, each chapter serves as a self-contained block of information about a particular ASP.NET feature.

Getting Started

If you've gotten this far, you're probably ready to begin writing some code. Before beginning, make sure that Visual Studio 2005 is installed on your machine. As long as you've installed the development environment, you can be sure the .NET runtime support is installed as well.

The first few examples will require nothing but a text editor and a working installation of Internet Information Services. To start, we'll begin with some basic examples to illustrate ASP.NET's object-oriented nature and compilation model. In addition to letting you see exactly how ASP.NET works when handling a request, this is a good time to lay out ASP.NET's architecture from a high level. We'll progress to Web form programming and soon begin using Visual Studio to write code (which makes things much easier!).

After learning the fundamentals of Web form development, we'll break apart the rest of ASP.NET, using examples to understand ASP.NET's features such as server-side controls, content caching, writing custom handlers, caching output and data, and debugging and diagnostics, all the way to ASP.NET's support for Web Services.

Finding Your Best Starting Point in This Book

This book is designed to help you build skills in a number of essential areas. You can use this book whether you are new to Web programming or you are switching from another Web development platform. Use the following table to find your best starting point in this book.

If you are	Follow these steps
New	
To Web development	1. Install the code samples. 2. Work through the examples in Chapters 1 and 2 sequentially. They will ground you in the ways of Web development. They will also familiarize you with ASP.NET and Visual Studio. 3. Complete the rest of the book as your requirements dictate.
New	
To ASP.NET and Visual Studio	1. Install the code samples. 2. Work through the examples in Chapter 2. They provide a foundation for working with ASP.NET and Visual Studio. 3. Complete the rest of the book as your requirements dictate.
Migrating	
From ASP.NET 1.*x* or from classic ASP	1. Install the code samples. 2. Skim the first two chapters to get an overview of Web development on the Microsoft platform and Visual Studio 2005. 3. Concentrate on Chapters 3 through 20 as necessary. You may already be familiar with some topics and may only need to see how a particular feature differs between ASP.NET 1.*x* and ASP.NET 2.0. In other cases, you may need to explore a feature that's completely new for ASP.NET 2.0.
Referencing	
The book after working through the exercises	1. Use the Index or the Table of Contents to find information about particular subjects. 2. Read the Quick Reference sections at the end of each chapter to find a brief review of the syntax and techniques presented in the chapter.

Conventions and Features in This Book

This book presents information using conventions designed to make the information readable and easy to follow. Before you start the book, read the following list, which explains conventions you'll see throughout the book and points out helpful features in the book that you might want to use.

Conventions

- Each chapter includes a summary of objectives near the beginning.
- Each exercise is a series of tasks. Each task is presented as a series of steps to be followed sequentially.
- Notes labeled "Tip" provide additional information or alternative methods for completing a step successfully.

- Text that you type appears in bold, like so:

```
class foo
{
   System.Console.WriteLine("HelloWorld");
}
```

- The directions often include alternate ways of accomplishing a single result. For example, adding a new item to a Visual Studio project may be done from either the main menu, or by right mouse clicking in the Solution Explorer.
- Most of the examples in this book are written using C#. However a few chapters have examples in both C# and Visual Basic so you may see how the same programming idioms are expressed in different languages.

Other Features

- Some text includes sidebars and notes to provide more in-depth information about the particular topic. The sidebars might contain background information, design tips, or features related to the information being discussed. They may also inform you about how a particular feature may differ in this version of ASP.NET.
- Each chapter ends with a Conclusion and a Quick Reference section. The Quick Reference section contains concise reminders of how to perform the tasks you learned in the chapter.

System Requirements

You'll need the following hardware and software to complete the practice exercises in this book:

Note The Visual Studio 2005 software is *not* included with this book! The CD-ROM packaged in the back of this book contains the codes samples needed to complete the exercises. The Visual Studio 2005 software must be purchased separately.

- Microsoft Windows XP Professional with Service Pack 2, Microsoft Windows Server 2003 with Service Pack 1, or Microsoft Windows 2000 with Service Pack 4
- Microsoft Internet Information Services (IIS) (included with Windows)
- Microsoft Visual Studio 2005 Standard Edition or Microsoft Visual Studio 2005 Professional Edition
- Microsoft SQL Server 2005 Express Edition (included with Visual Studio 2005) or Microsoft SQL Server 2005
- 600 MHz Pentium or compatible processor (1 GHz Pentium recommended)

- 192 MB RAM (256 MB or more recommended)
- Video (800 × 600 or higher resolution) monitor with at least 256 colors (1024 × 768 High Color 16-bit recommended)
- CD-ROM or DVD-ROM drive
- Microsoft Mouse or compatible pointing device

You will also need to have Administrator access to your computer to configure SQL Server 2005 Express Edition.

Using Microsoft Access

Chapter 13 on databinding and Chapter 14 on application data caching both use Microsoft Access. If you want to look at the databases and modify them, you need to have installed Microsoft Access on your machine. If you have Microsoft Office, you probably already have it. There is nothing special you need to do to set it up, and there is nothing special you need to do to use the databases within the ASP.NET applications.

Code Samples

The companion CD inside this book contains the code samples, written in C#, that you'll use as you perform the exercises in the book. By using the code samples, you won't waste time creating files that aren't relevant to the exercise. The files and the step-by-step instructions in the lessons also let you learn by doing, which is an easy and effective way to acquire and remember new skills.

Note If you prefer to use code samples written in Visual Basic, you can download a Visual Basic version of the code samples. See the "Installing the Visual Basic Code Samples" section for more information.

Installing the C# Code Samples

Follow these steps to install the C# code samples on your computer so that you can use them with the exercises in this book.

Note The code sample installer modifies IIS, so you must have Administrator permissions on your computer to install the code samples.

1. Remove the companion CD from the package inside this book and insert it into your CD-ROM drive.

Note An end user license agreement should open automatically. If this agreement does not appear, open My Computer on the desktop or Start menu, double-click the icon for your CD-ROM drive, and then double-click StartCD.exe.

2. Review the end user license agreement. If you accept the terms, select the accept option and then click Next.

   A menu will appear with options related to the book.

3. Click Install Code Samples.
4. Follow the instructions that appear.

Note If IIS is not installed and running, a message will appear indicating that the installer cannot connect to IIS. You can choose to ignore the message and install the code sample files, however, the code samples that require IIS will not run properly.

The code samples will be installed to the following location on your computer: C:\Microsoft Press\ASP.NET 2.0 Step by Step\

The installer will create a virtual directory named aspnet2sbs under the Default Web Site. Below the aspnet2sbs virtual directory, various Web applications are created. To view these settings, open the Internet Information Services console.

Installing the Visual Basic Code Samples

Follow these steps to download and install the Visual Basic code samples on your computer so that you use them with the exercises in this book.

Note The code sample installer modifies IIS, so you must have Administrator permissions on your computer to install the code samples.

1. Download the Visual Basic code samples installer from the book's online companion content page:
   http://www.microsoft.com/mspress/companion/0-7356-2201-9/
2. Run the installer.
3. Follow the instructions that appear.

Note If IIS is not installed and running, a message will appear indicating that the installer can not connect to IIS. You can choose to ignore the message and install the code sample files, however, the code samples that require IIS will not run properly.

The code samples will be installed to the following location on your computer: C:\Microsoft Press\ASP.NET 2.0 Step by Step\

The installer will create a virtual directory named aspnet2sbs under the Default Web Site. Below the aspnet2sbs virtual directory, various Web applications are created. To view these settings, open the Internet Information Services console.

Using the Code Samples

Each chapter in this book explains when and how to use any code samples for that chapter. When it's time to use a code sample, the book will list the instructions for how to open the files. Many chapters begin projects completely from scratch so you can grok the whole development process. Some examples borrow bits of code from previous examples.

Here's a comprehensive list of the code sample projects.

Project	Description
Chapter 1	
HelloWorld.asp, Selectnoform.asp, Selectfeature.htm, Selectfeature2.htm, Selectfeature.asp	Several Web resources illustrating different examples of raw HTTP requests.
WebRequestor	A simple application that issues a raw HTTP Request.
Chapter 2	
HelloWorld, HelloWorld2, HelloWorld3, HelloWorld4, HelloWorld5, partial1.cs, partial2.cs	Web resources illustrating ASP.NET's compilation models and partial classes.
Chapter 3	
BunchOfControls.htm, BunchOf-Controls.asp, BunchOfControls.aspx	Web resources illustrating rendering control tags.
ControlORama	Visual Studio-based project illustrating Visual Studio and server-side controls.
Chapter 4	
ControlORama	Illustrates creating and using rendered server-side con-trols.
Chapter 5	
ControlORama	Illustrates creating and using composite server-side controls and *User* controls.
Chapter 6	
ControlPotpourri	Illustrates control validation, the *TreeView*, and the *MultiView / View* controls.
Chapter 7	
UseWebParts	Illustrates using Web Parts within a Web application.
Chapter 8	
MasterPageSite	Illustrates developing a common look and feel through-out multiple pages within a single Web application using Master Pages, Themes, and Skins.
Chapter 9	
ConfigORama	Illustrates configuration within ASP.NET. Shows how to manage the Web.Config file, how to add new configuration elements and how to retrieve those configuration elements.
Chapter 10	
SecureSite	Illustrates Forms Authentication and authorization within a Web site.
Login.aspx, OptionalLogin.aspx, Web.Config, Web.ConfigForceAuthenti-cation, Web.ConfigForOptionalLogin	Web resources for illustrating Forms Authentication at the very barest level.