

Pro ASP.NET 3.5 Server Controls and AJAX Components



**Rob Cameron and
Dale Michalk**

Pro ASP.NET 3.5 Server Controls and AJAX Components

Copyright © 2008 by Rob Cameron, Dale Michalk

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-59059-865-8

ISBN-10 (pbk): 1-59059-865-2

ISBN-13 (electronic): 978-1-4302-0462-6

ISBN-10 (electronic): 1-4302-0462-1

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Microsoft product screen shot(s) reprinted with permission from Microsoft Corporation.

Lead Editor: Ewan Buckingham

Technical Reviewer: Fabio Claudio Ferracchiati

Editorial Board: Clay Andres, Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell,

Jonathan Gennick, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Frank Pohlmann, Ben Renow-Clarke,

Dominic Shakeshaft, Matt Wade, Tom Welsh

Project Manager: Kylie Johnston

Copy Editor: Heather Lang

Associate Production Director: Kari Brooks-Copony

Production Editor: Ellie Fountain

Compositor: Susan Glinert

Proofreader: Liz Welch

Indexer: Brenda Miller

Artist: Kinetic Publishing Services, LLC

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit <http://www.apress.com>.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales—eBook Licensing web page at <http://www.apress.com/info/bulksales>.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com>.

*To my beautiful wife, Ally, and daughters Amanda and Anna,
who bring so much happiness to my life
—Rob Cameron*

Contents at a Glance

About the Authors	xvii
About the Technical Reviewer	xix
Acknowledgments	xxi
Introduction	xxiii
■ CHAPTER 1 Server Control Basics	1
■ CHAPTER 2 Encapsulating Functionality in ASP.NET	43
■ CHAPTER 3 ASP.NET State Management	85
■ CHAPTER 4 The WebControl Base Class and Control Styles	123
■ CHAPTER 5 Server Control Events	183
■ CHAPTER 6 Server Control Templates	253
■ CHAPTER 7 Server Control Data Binding	281
■ CHAPTER 8 Integrating Client-Side Script	347
■ CHAPTER 9 ASP.NET AJAX Controls and Extenders	413
■ CHAPTER 10 Other Server Controls	441
■ CHAPTER 11 Design-Time Support	523
■ CHAPTER 12 Building a Complex Control	577
■ CHAPTER 13 Packaging and Deployment	657
■ INDEX	713

Contents

About the Authors	xvii
About the Technical Reviewer	xix
Acknowledgments	xxi
Introduction	xxiii
CHAPTER 1 Server Control Basics	1
Source Code	1
The Heart and Soul of ASP.NET	1
A .NET Framework “Hello, World” Web Form	2
Control Properties	7
Control Methods	9
Control Events	9
The Web Page As a Control Tree	11
The Root Controls	13
The System.Web.UI Namespace	14
System.Web.UI.HtmlControls Namespace	14
The System.Web.UI.WebControls Namespace	20
Web Controls vs. HTML Controls	40
Summary	41
CHAPTER 2 Encapsulating Functionality in ASP.NET	43
Packaging Content in ASP.NET	43
Inheritance	44
Encapsulation	45
Comparing the Control-Building Techniques	45
User Controls	45
Custom Server Controls	49
Building a User Control	52
Building a Custom Control	60
ASP.NET AJAX	78
ASP.NET AJAX UpdatePanel Server Control	78
ASP.NET AJAX UpdateProgress Server Control	79

Using Design-Time Attributes	82
What's an Attribute?	82
Common Design-Time Attributes	83
Summary	84

■ CHAPTER 3 **ASP.NET State Management** 85

ASP.NET Request-Processing Architecture	85
HttpHandler	87
ASP.NET and Server-Side State Management	88
The Context Object	88
Server-Side State Considerations	89
ASP.NET and Client-Side State Management	89
URL Strings	90
Cookies	90
HTML Hidden Variables	91
ViewState	93
A Client State Workshop	96
Reading the Client State	100
Getting the URL State	101
ASP.NET Server Controls and State	102
Form Post Data and ASP.NET Controls	108
The IPostBackDataHandler Interface	108
The Textbox Control	109
Using the Textbox Control	111
ASP.NET Control State	115
ViewState Is Now Application User State	115
New TextBox3d Demonstration Web Form	116
Adding Control State to TextBox3D	118
Summary	121

■ CHAPTER 4 **The WebControl Base Class and Control Styles** 123

Customizing the Appearance of Controls	123
HTML: Content and Appearance	124
Styling Using Tags	124
Styling Using Cascading Style Sheets	124
Style Properties and Visual Studio	127

WebControl and Control Styling	130
WebControl's ControlStyle Property	131
WebControl Top-Level Style Properties	132
The Style Property	133
A New Rendering System	134
A Styled Label Control	134
The AddAttributesToRender() Method	135
A Styled TextBox Control	136
The Web Control Style Web Form	139
Styles, HTML 3.2, and Down-Level Browsers	147
Down-Level Browser Style Rendering Behind the Scenes	149
Custom Styling	149
The Styled InputBox Control	149
LabelStyle and TextBoxStyle	152
Customizing ViewState	153
Rendering the Output	154
The InputBox Style Web Form	158
Applying the LabelStyle and TextBoxStyle Settings	163
Creating a Custom Style Class	166
The CursorStyle Enumeration	166
The FancyLabel Control	171
Rendering the FancyLabel Control	172
The FancyLabel Style Web Form	174
The StyleCollection Class	178
Summary	181

CHAPTER 5 **Server Control Events**

Events and ASP.NET Controls	183
The Need for Events in ASP.NET	183
The .NET Framework Event Model	185
Delegates	186
Events	190
System.EventHandler Delegate	190
Invoking an Event in a Control	191
Adding an Event to the TextBox Control	191
Enhancing the TextBox Control with a TextChanged Event	191
Using the TextBox Control on a Web Form	194

Creating a Custom Event	198
Creating a TextChangedEventArgs Class	198
Creating a TextChangedEventHandler Delegate	199
Adding an Event to the CustomEventTextBox Control	200
Using the CustomEventTextBox Control on a Web Form	203
Capturing Postback with the Button Control	207
Rendering the Button	207
Exposing a Click Event and the Events Collection	209
Command Events and Event Bubbling	211
Exposing the Command Event	211
Capturing the Postback via IPostBackEventHandler	213
Using the SuperButton Control on a Web Form	217
Composing the SuperButton Control into a Composite	
Pager Control	224
Building the Pager Child Control Hierarchy	224
Defining the PageCommand Event	226
Exposing the PageCommand Event from the	
Pager Control	227
Capturing the Bubbles via OnBubbleEvent	228
The INamingContainer Interface	229
Using the Pager Control on a Web Form	233
Control Life Cycle	237
Plugging Into the Life Cycle	238
The Lifecycle Server Control	239
Life Cycle and the HTTP Protocols GET and POST	239
HTTP POST Request via Postback	247
Summary	250

■ CHAPTER 6 **Server Control Templates** 253

Customized Control Content	253
Using Control Templates	254
The ParseChildren Attribute	254
A Menu Control with Templates	256
The Template Properties	257
Creating the Header Section	258
Creating the Footer Section	260
Creating the Hyperlink Section	260
Viewing the TemplateMenu Control	266
Checking the Rendered HTML	268

Parsing Data from the Control Tags	268
The TagDataMenu Control	268
The BuilderMenuControl	273
Viewing the Tag Parsing Menu Controls	278
Summary	280
 CHAPTER 7 Server Control Data Binding	281
Customized Control Content	282
Control Data Binding	282
DataBinding Base Class Options	282
The Repeater Control	283
Data Binding with the Repeater Control	312
Advanced Interaction with the Repeater Control	318
Using Dynamic Templates	323
The Dynamic Templates Web Form	323
Implementing the ITemplate Interface	329
CompositeDataBoundControl	334
Summary	344
 CHAPTER 8 Integrating Client-Side Script	347
Client-Side Script Server Control Scenarios	347
Handling Client-Side Events	348
The Click Web Form	350
Handling Mouse Events for Image Rollovers	352
The RolloverImage Web Form	362
Running a Client Script When a Form Is Submitted	366
The FormConfirmation Control	366
The ConfirmedLinkButton Control	367
The Confirm Web Form	369
Integrating Client-Side and Server-Side Events	374
The UpDown Server Control	374
The UpDown Web Form	392
Client Callbacks	395
Client Callbacks API	396
The Callback Web Form	396
The StockNews Callback Control	404
Summary	412

CHAPTER 9	ASP.NET AJAX Controls and Extenders	413
	ASP.NET AJAX	413
	Partial Page Updates	414
	SimpleUserControlAJAX Demonstration	414
	ASP.NET AJAX Extensibility	416
	The GetScriptReferences Method	417
	The GetScriptDescriptors Method	418
	ASP.NET AJAX Client Script	419
	HoverButton Example	419
	ASP.NET AJAX Server Controls	426
	The TextCaseExtender Control	426
	The TextCaseBehavior Client-Side Component	428
	The HighlightedHyperLink ASP.NET AJAX Server Control	432
	The HighlightedHyperlink Client-Side Component	435
	Summary	440
CHAPTER 10	Other Server Controls	441
	Web-Part-Based Web Site Development	441
	Web Part Development	442
	Web Part Infrastructure	442
	Creating Web Parts	443
	Web Part Development Tips	476
	Adaptive Control Behavior	477
	Nonmobile Adaptive Behavior	477
	Mobile Controls Overview	482
	Browsing Mobile Web Forms	487
	Customizing and Implementing Mobile Controls	488
	Templates and Device-Specific Choices	491
	The DeviceSpecific.aspx Mobile Web Page	491
	Templates	492
	The DeviceSpecific and Choice Elements	493
	Filter Attribute and deviceFilters Configuration	494
	MobileCapabilities, browserCaps, and Device Update 2	495
	New Capabilities in MobileCapabilities	497
	User Controls	502
	Mobile User Controls	503
	Miniaturizing the Header and Footer	503
	Hosting the Header and Footer User Controls	504

Custom Controls	504
Rendering the Mobile Control	505
The Mobile Control Life Cycle	507
Inheritance	511
Composition	511
Inheriting from MobileControl	511
Testing MCTextBox	519
Summary	521
 CHAPTER 11 Design-Time Support	523
Professional Quality	523
Design-Time Architecture	523
Environment Services Overview	524
Customizing Component Behavior	526
Attributes	527
The TitledThumbnail Control	527
The TitledThumbnail Control at Design Time	532
Type Converters	538
UI Type Editors	545
The SimpleTextEditor Editor	545
The Collection Editor	548
Component Editors	550
The Component Editor Dialog Box	550
The Component Editor Class	555
Custom Designers	558
The Control Designer and Designer Verbs	560
The Templated Control Designer	564
The Data-Bound Control Designer	568
Miscellaneous Design-Time Items	573
The Toolbox Icon	573
Debugging Design-Time Development	573
Summary	574
 CHAPTER 12 Building a Complex Control	577
The Problem Domain	577
The Live Search Web Service	578
Web Services Description Language and .NET Web Service Proxies	579

Creating the Control Library Project	583
Strong-Named Assemblies and Versioning Attributes	584
Bin Directory or Global Assembly Cache Deployment	584
Additional Assembly Attributes	585
Configuring the Search Settings	586
Crafting the Configuration Section XML	586
Registering the Configuration Section	587
Building a Configuration Section Handler Class	589
Wrapping the Web Service Proxy in a Utility Method	591
Designing the Control Architecture	593
The Search Control	595
Handling the Search	596
The Result Control	604
The ResultItem Control	605
Building the Result Control	609
Creating a Control Hierarchy for Data Binding or Postback	611
Creating ResultItem Controls	614
Creating the Child Pager Control	616
Managing Paging	617
Styling the Result Control	618
The Pager Control	643
Creating the Pager Results	644
Creating the Pager's Previous Button	645
Creating the Pager's Bar Pages	646
Creating the Pager's Next Button	647
Ensuring Pager's Style Rendering	648
Summary	655

■ CHAPTER 13 Packaging and Deployment

Designer Support	657
Designers and Dummy Data Source	657
Template Support in the Result Control	666
Toolbox Image Icons	670
Testing the Live Search Controls	671
The Default Look and Feel	671
Customizing the Live Search Controls' Appearance	674

Licensing Support	677
The RsaLicense License	678
License Cryptography	681
Generating the License	683
The RsaLicenseDataAttribute Custom Attribute	685
Adding Licensing to the Search and Result Controls	686
The RsaLicenseProvider Class	688
Globalization and Localization	696
The CultureInfo Class	696
The ResourceManager Class	697
Culture Types and Localizing Resource Files	700
Satellite Assemblies and Resource Fallback	702
Setting Thread Culture in the Global.asax File	704
Viewing a Localized Web Form	705
Code Analysis for Managed Code	709
Documentation	711
Summary	712

INDEX	713
--------------------	------------

About the Authors



■ **ROB CAMERON** is employed with Microsoft Corporation in Atlanta, GA. He has been with Microsoft since 2001 assisting communications sector and media and entertainment companies build solutions on the Microsoft platform. Prior to employment at Microsoft, he worked as an independent consultant developing software on the Microsoft platform for over five years. He has a master's degree in information technology management and a bachelor's degree in computer science. A former naval officer and United States Naval Academy graduate, he enjoys spending his free time with his wife and two daughters.



■ **DALE MICHALK** is employed with Microsoft Corporation in Dallas, Texas. He has been with Microsoft since 2001, where he helps promote .NET as a development platform and assists companies interested in migrating to new technologies such as ASP.NET. He is a former U.S. Army officer and West Point graduate.

About the Technical Reviewer

■ **FABIO CLAUDIO FERRACCHIATI** is a senior consultant and a senior analyst/developer using Microsoft technologies. He works for Brain Force (www.brainforce.com) in its Italian branch (www.brainforce.it). He is a Microsoft Certified Solution Developer for .NET, a Microsoft Certified Application Developer for .NET, a Microsoft Certified Professional, and a prolific author and technical reviewer. Over the past ten years, he's written articles for Italian and international magazines and coauthored more than ten books on a variety of computer topics. You can read his LINQ blog at www.ferracchiati.com.

Acknowledgments

Writing a book is a long and incredible journey that requires the support and care of a lot of people. The first and foremost of those I would like to recognize are my family members. Without their support and patience with all those long hours on the computer, this book would never have come to pass. I would like to thank Dale Michalk for inviting me on this journey, starting with our first book *Building ASP.NET Server Controls*. Dale's contributions to the first book are no doubt a significant part of this effort as well, and that is why Dale's name appears on the front cover of this book.

Apress is a fantastic company to work for as an author, as evidenced by their care and feeding in getting this book into production. This is a publishing house run by those who actually write for a living; they understand the balance in ensuring high quality versus meeting deadlines. Thanks especially to Matthew and Kylie for all the patience in the slipped schedules and author changes. Thanks to the editing folks from Apress—Kylie, Heather, and Ellie—as well as to those who I don't know by name but whose efforts helped to make this book possible. I would also like to thank Fabio Claudio Ferracchiati, who reviewed the book and provided technical assistance and support.

A final thanks is owed to the ASP.NET product team who provided the Microsoft web development community with an awesome product and are busy at work on future versions that will reach new heights.

Rob Cameron

Introduction

With the explosion of the Internet, web development tools evolved as a combination of HTML and a scripting language, such as ASP or Perl, to generate dynamic output. With the advent of Microsoft's .NET Framework, ASP.NET turned web development on its head by combining a design-time interface similar to Visual Basic with an HTML and JavaScript output that requires nothing more than a web browser for rendering. With ASP.NET 3.5, HTML and JavaScript are combined in powerful ways via ASP.NET AJAX technology that helps connect client-side and server-side connection without losing point-and-click design-time support. We wrote this book to document the major improvements since ASP.NET 1.1, while also covering the fundamentals for those new to custom server control development.

At the core of ASP.NET is server control technology. From the `Page` class to the `Label` control to web parts, all objects in ASP.NET are server controls. Server controls combine server-side execution in a well defined life cycle with browser-friendly rendering that includes down-level browsers as well as a plethora of mobile clients. Regardless of the target output, all server controls behave in a similar manner. Understanding this technology and how to leverage it in your own development efforts are the subjects of this book.

Who This Book Is For

The target audience for this book consists of developers with an intermediate to advanced experience level looking to deepen their understanding of ASP.NET and its underlying server control architecture. The example code in this book is written in C#. However, if you are a VB.NET developer, the examples translate pretty easily, as ASP.NET development is language agnostic. The .NET Framework and the ASP.NET object model are what's important, not the language.

If you are a developer in need of learning a particular technique, each major facet of control development is presented with simple example code to highlight that particular topic. For example, if you are looking for information on how to add events to your server controls, or how to understand how events work in ASP.NET, you can drill into that chapter to get the details.

If you are a developer looking for full-featured example code, you'll find that here too. One example shows how to implement data binding and templates that can connect to a database backend. The rich example in the last part of the book pulls techniques described throughout this book into a holistic demonstration of how to build a rich, complex server control that is fully localized and includes licensing support.

How This Book Is Structured

This book is about server control technology as the underlying foundation of ASP.NET. It will provide you with a deep understanding of how server control technology works, as well as

explaining how to build your own custom server controls as part of a web development project or for resale in the component marketplace.

The first section of the book provides an introduction to server control technology. We also discuss the different ways to build a server control including inheritance from a base control (such as `Control` or `WebControl`) encapsulation, or composite controls, as well as inheritance from an existing or rich control, like the `TextBox` server control.

The second section of the book dives into deep a discussion on critical topics such as state management, server-side event handling, templates, data binding, and integrating client-side script, as well as considering advanced base classes such as `CompositeControl` and `DataBoundControl`. A common theme for all of these discussions is how the topic relates to the control life cycle. Understanding the control life cycle is critical to server control development as well as to ASP.NET development in general. Of course, there are copious amounts of code to support our discussions as well.

The third section of the book covers advanced development techniques such as building ASP.NET AJAX controls and extenders. We also cover web part development for ASP.NET or SharePoint. We round out the section with a discussion of control adapters for modifying an existing server control's HTML output and device adapters for mobile control development.

The last section of the book covers design-time support in detail. Many of the controls built in earlier chapters include design-time support; however, we centralize discussion of the design-time support capabilities in ASP.NET and server controls to facilitate understanding without cluttering up the earlier chapters. We finish up this last section of the book by walking through how to create a professional-quality server control with a discussion on licensing, globalization, and localization.

Prerequisites

The following applications would be helpful in working through the examples in this book, but access to them isn't required:

- Visual Studio 2008, Express edition
- SQL Server 2005 Express (for a couple of the database samples)
- Internet Information Services (for the mobile web project)

Downloading the Code

The source code for this book is available to readers at www.apress.com in the Source Code section of this book's home page. Please feel free to visit the Apress web site and download all the code there. You can also check for errata and find related titles from Apress.

Contacting the Authors

You can contact Rob Cameron via <http://blogs.msdn.com/robcamer>; there is a contact link to send Rob an e-mail there.



Server Control Basics

To create server controls, you need to understand how they work. This chapter provides a very high-level run-through of the various server control namespaces to set the scene for the rest of this book. To begin our journey, we'll start by reviewing what a server control provides to clients and taking a look at some of the prebuilt controls supplied by ASP.NET. We'll study the controls' inheritance bloodlines for the HTML and web controls, examining how the namespaces are organized, so that you become familiar with what is available for immediate use in ASP.NET. Because inheritance and composition of existing server controls are important timesaving control-building techniques available in ASP.NET, this rapid journey through the object model is well worth the effort.

To begin this chapter, we start out with a "Hello, World" form to demonstrate master pages. The `MasterPage` class can trace its inheritance back to the user control functionality introduced in ASP.NET 1.0. We next discuss the basic server control construction, as well as how server controls are organized in an ASP.NET web form. Finally, we cover the root server control namespaces with an example of the types of server controls found in the different namespaces.

Source Code

The source code for this book is available for download from the Apress web site for those who want to follow along by running the code in Visual Studio 2008. The web site project is file based, so having IIS installed and configured isn't required. There is a main solution file titled `ControlsBook2Solution.sln` that, when opened, will load all of the projects. Please refer to the read-me file included with the source code download for detailed instructions on how to get the code running. The full source code is also printed in this book, so those who want to read while not in front of a computer can still enjoy reading the source code.

The Heart and Soul of ASP.NET

Each piece of HTML delivered by an ASP.NET page, whether a `` tag without server-side interactivity, a complex list control such as the `DataGrid` that supports templates, or the web form itself that hosts the HTML tags, is generated by an object that inherits from the `System.Web.UI.Control` base class. These objects, or server controls, are the engine that drives the ASP.NET page-rendering process. The fact that every snippet of rendered HTML exists as a server control allows for a consistent page parsing process that permits easy control configuration and manipulation to create dynamic and powerful content. The clean, consistent object

model provided by ASP.NET also facilitates extension through custom server controls that share a common object model.

A .NET Framework “Hello, World” Web Form

The first step on our journey through the ASP.NET server controls is construction of a “Hello, World” web form. Before actually creating the “Hello, World” web form, we need to create a master page to provide a consistent UI for the book web site. A master page, one of the many new features in ASP.NET 2.0 and later versions, has a `@Master` directive at the top of the code instead of the `@Page` directive on a standard web form.

Note ASP.NET 3.5 includes additional master page item templates to support AJAX functionality and nested master pages called AJAX Master Page and Nested Master Page respectively.

The `@Master` directive takes most of the same options as the `@Control` directive. If you have not migrated to ASP.NET 2.0 or later, master pages are a welcome addition in ASP.NET and should often be used for page layout and template purposes in situations where ASP.NET user controls were in ASP.NET 1.1 but came up short. Figure 1-1 shows the master page used in this book’s sample web site.

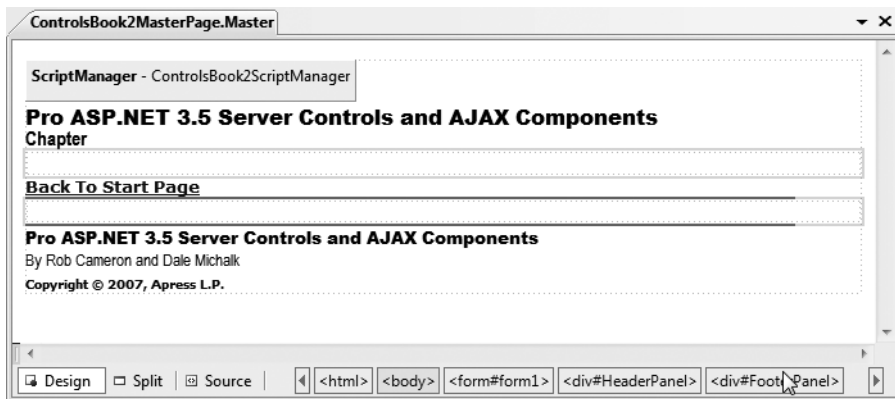


Figure 1-1. *The Controls Book 2 web site’s master page*

Web forms added to the project can be configured to use the master page rendering at design time, like Figure 1-2.

Notice in Figure 1-2 that the master page area is grayed out (and cannot be edited) at design time in a web content form. The design-time view displays the master page HTML and the web content form HTML, providing a more accurate view of the rendered web form. Listings 1-1 and 1-2 show the master page source page and code-behind file.

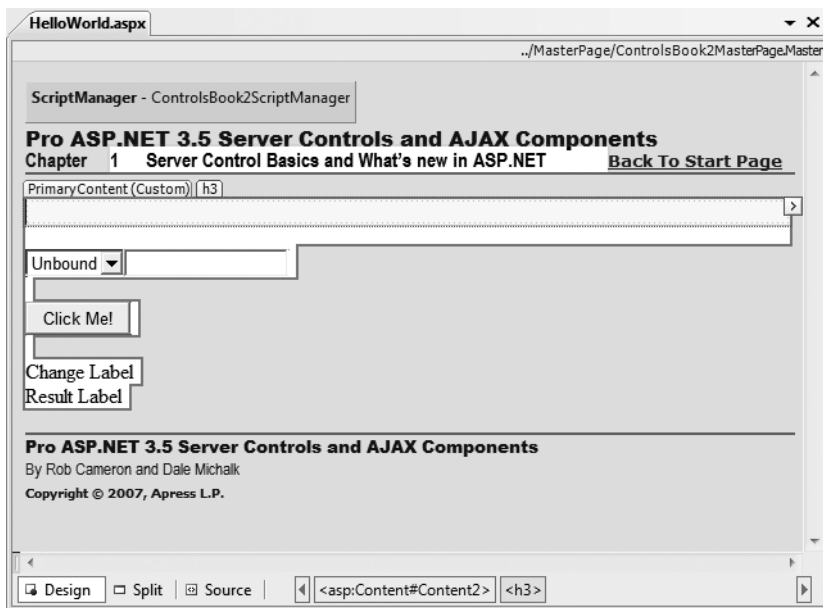


Figure 1-2. The Controls Book 2 web site's master page displayed in a web content form

Listing 1-1. The ControlsBook2 Master Page File

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="ControlsBook2MasterPage.master.cs"
Inherits="ControlsBook2Web.MasterPage.ControlsBook2MasterPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title>Master Page</title>
  <link href="../css/ControlsBook2Master.css" rel="stylesheet" type="text/css" />
  <link href="../css/SkinnedControl.css" rel="stylesheet" type="text/css" />
  <asp:ContentPlaceHolder ID="HeadSection" runat="server">
  </asp:ContentPlaceHolder>
</head>
<body>
  <form id="form1" runat="server">
    <div id="HeaderPanel">
      <asp:ScriptManager ID="ControlsBook2ScriptManager" runat="server">
        <Scripts>
          <asp:ScriptReference Path="../ch09/hoverbutton.js" />
        </Scripts>
      </asp:ScriptManager>
      <asp:Label ID="Label2" CssClass="TitleHeader" runat="server" Height="18px"
Width="604px">Pro ASP.NET 3.5 Server Controls and AJAX Components</asp:Label>
      <br />
    </div>
  </form>
</body>
</html>
```

</body>

Listing 1-2. *The ControlsBook2MasterPage Master Page Code-Behind Class File*

```
using System;
```

```
{
    public partial class ControlsBook2MasterPage : System.Web.UI.MasterPage
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }
    }
}
```

In the master page for the Controls Book 2 web site, the chapter number and chapter title have ContentPlaceHolder placeholder tags to allow content pages to update the chapter number and title.

Each web form sets values for the chapter title and number by simply placing the value in the corresponding Content tag in the content page. This is a simple example of providing a consistent user interface in a web site, but still allowing customization.

Tip ASP.NET User Controls are still present in ASP.NET 3.5. In fact the `MasterPage` class inherits from the `UserControl` class.

The resulting arrangement is shown in Figure 1-3 with a DropDownList control, a TextBox control, two Label controls, and a Button control. The resulting source code generated by Visual Studio 2008 is shown in Listings 1-3 and 1-4.

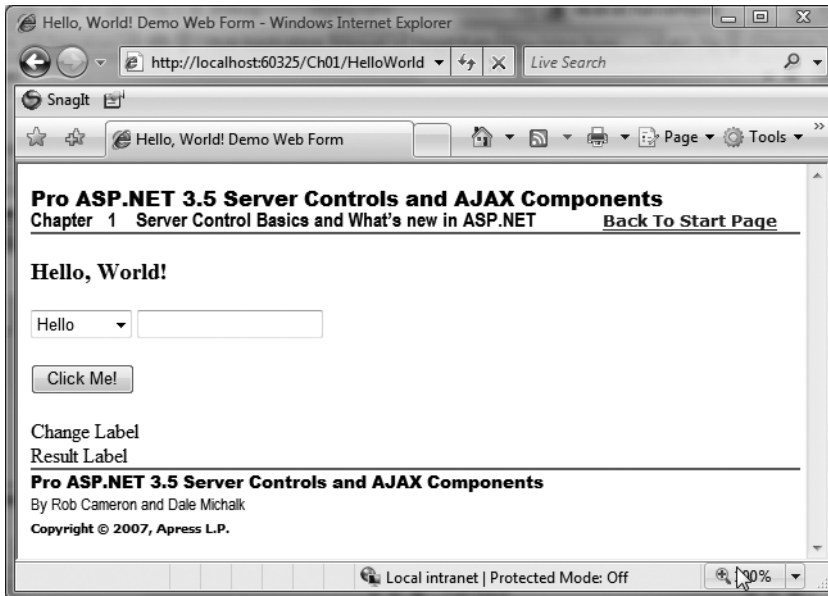


Figure 1-3. *The HelloWorld server control web form*

Listing 1-3. *The HelloWorld Demo Web Form .aspx File*

```
<%@ Page Language="C#"
MasterPageFile="~/MasterPage/ControlsBook2MasterPage.Master"
    AutoEventWireup="true" CodeBehind="HelloWorld.aspx.cs"
Inherits="ControlsBook2Web.Ch01.HelloWorld"
    Title="Hello, World! Demo Web Form" %>

<asp:Content ID="Content1" ContentPlaceHolderID="ChapterNumAndTitle" runat="server">
    <asp:Label ID="ChapterNumberLabel" runat="server"
Width="14px">1</asp:Label>&nbsp;&nbsp;&nbsp;<asp:Label
    ID="ChapterTitleLabel" runat="server" Width="360px">
Server Control Basics and What's new in ASP.NET</asp:Label>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="PrimaryContent" runat="server">
    <h3><asp:Label ID="Label1" runat="server" Text=
```

```

"Hello, World!"></asp:Label></h3>
<asp:DropDownList ID="Greeting" runat="server" ToolTip="Select a greeting">
</asp:DropDownList>
<asp:TextBox ID="Name" runat="server" Font-Italic="True" ToolTip="Enter your name"
    OnTextChanged="Name_TextChanged"></asp:TextBox><br />
<br />
<asp:Button ID="ClickMe" runat="server" Text="Click Me!"
OnClick="ClickMe_Click"></asp:Button><br />
<br />
<asp:Label ID="Changelabel" runat="server">Change Label</asp:Label><br />
<asp:Label ID="Resultlabel" runat="server">Result Label</asp:Label>
<br />
</asp:Content>

```

Listing 1-4. *The HelloWorld Server Control Demo Code-Behind Class File*

```

using System;
using System.Collections;

namespace ControlsBook2Web.Ch01
{
    public partial class HelloWorld : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            ArrayList list = new ArrayList();
            list.Add("Hello");
            list.Add("Goodbye");

            Greeting.DataSource = list;
            Greeting.DataBind();
        }

        protected void ClickMe_Click(object sender, EventArgs e)
        {
            Resultlabel.Text = "Your new message: " + Greeting.SelectedItem.Value +
            "&nbsp;" + Name.Text + "!";
        }

        protected void Name_TextChanged(object sender, EventArgs e)
        {
            Changelabel.Text = "Textbox changed to " + Name.Text;
        }
    }
}

```

The server controls on our “Hello, World” web form (specifically, the `Label`, `TextBox`, and `DropDownList` objects) render as HTML and, for the `TextBox` control, remember what is typed in the control between postback cycles. The HTML rendered to the browser is backed by powerful objects that can be wired up to programming logic to perform useful work on the web server. During server-side processing, the object-oriented nature of server controls provides us with three main constructs to interact with controls as objects: properties, methods, and events. We discuss these constructs in the sections that follow.

Control Properties

The most common means of working with a server control is through the properties it exposes. Properties allow the control to take information from the web form to configure its output or modify its behavior in the HTML-generation process.

Note Properties are different and more powerful than public data members. Properties provide an additional layer of abstraction through the use of `get` and `set` methods; `get` and `set` methods or function calls provide a convenient location for programming logic, such as displaying an error if a value is out of range or otherwise invalid, enforcing read-only access (implementing a `get` method only), and so on. Properties can be declared as `public`, `protected`, or `private`.

Properties are easily viewable in the Properties window available when you select a control in the Visual Studio Design view of the `.aspx` page. Figure 1-4 shows the Properties window when the `Name` `TextBox` is selected. Notice that the `Font` property has been configured to show the `TextBox`’s `Text` property text in italics.

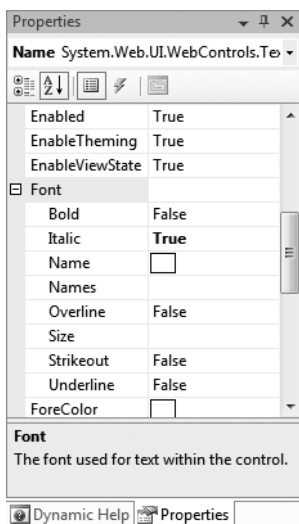


Figure 1-4. *The Properties window for the `TextBox` control*

The Visual Studio Designer translates the entries in the Properties window into attribute values on the HTML view of the .aspx page. To see this, set a property for a control in the Properties tool window and then switch to HTML view. Likewise, if you modify attribute values in the HTML view of the .aspx page, these changes will be reflected in the Designer, assuming you typed in the values correctly. This behavior can be very handy for quickly duplicating attributes between controls. Simply copy the HTML version of the attributes and then paste the HTML into the target control that you want to match the original. You can think of the Designer as a code generator that allows you to declaratively work with the look and feel of the ASP.NET application without having to write the code. As an example, the Font settings set in the Properties window for the TextBox control described previously map directly to Font attributes:

```
<asp:TextBox id="Name" runat="server" Font-Italic="True"
            ToolTip="Enter your name" OnTextChanged="Name_TextChanged">
</asp:TextBox>
```

The Label and TextBox controls work a little differently than most, in that the content between the opening and closing tags is controlled by the Text property:

```
<asp:Label id="Resultlabel" runat="server">Result Label</asp:Label>
```

You can also set a control's properties programmatically in the code-behind class file. The "Hello, World" demonstration sets the Text property for Label1 to a blank string each time the web form is loaded, to overwrite the Label value that is declaratively set in the .aspx page. The activity happens in a method named Page_Load that is mapped to the Page object's Load event:

```
protected void Page_Load(object sender, EventArgs e)
{
    Resultlabel.Text = "";
    ChangeLabel.Text = "";

    if (!Page.IsPostBack)
    {
        UpdateMaster();
        LoadDropDownList();
    }
    DataBind();
}
```

You can also use the properties exposed by the control to read input from the client browser during postback on the server side. The Button click event handling routine in the "Hello, World" web form reads the Text property of the TextBox control and the Value property of the SelectedItem property on the DropDownList control to display the greeting to the client of the web browser:

```
protected void ClickMe_Click(object sender, EventArgs e)
{
    Resultlabel.Text =
        "Your new message: " + Greeting.SelectedItem.Value + "&nbsp;" + Name.Text + "!";
}
```

Control Methods

The second feature exposed by a server control is a collection of object methods. Functionality implemented using methods typically goes beyond the features of a property's set or get method; they usually perform a more complex action against the control. One of the best examples in ASP.NET of using methods for a server control is the data-binding process that links a control with a data source.

In the “Hello, World” web form example, the `Page_Load` event checks to see if the page is requested via a form postback or if it was called for the first time using HTTP GET so that the page can generate the initial HTML for the browser, creating the option list. In the postback scenario, the code to create the option list is not necessary for the `DropDownList` control via the `LoadDropDownList()` method, because the server control `DropDownList1` maintains its internal option list via the web form `ViewState` mechanism for subsequent postback operations to the server. We cover `ViewState` extensively in Chapter 3.

The page's `LoadDropDownList()` method's first task is to create an `ArrayList` collection and load it with the string values “Hello” and “Goodbye”. It also links the `ArrayList` to the `DropDownList` by setting the `DataSource` property to the `ArrayList`:

```
private void LoadDropDownList()
{
    ArrayList list = new ArrayList();
    list.Add("Hello");
    list.Add("Goodbye");

    Greeting.DataSource = list;
}
```

Note that we do not call the `DataBind()` method directly for `DropDownList`. Instead, we call the `DataBind()` method on the `Page_Load` handler itself. The `DataBind()` method of the `Page` class recursively calls the `DataBind()` methods for all its child controls that have references to a data source. In this case, when the `Page` class's `DataBind()` method is invoked, the `DropDownList` control data binds to the `ArrayList` object as shown previously.

Control Events

Events are the final constructs used for interacting with controls that we discuss in this chapter. Events provide a mechanism to notify clients of state changes inside the control. In ASP.NET, events always coincide with an HTTP POST submission back to the web server. Through the automatic postback mechanism, events in ASP.NET appear to behave very much like their counterparts in a Windows Forms application.

Note Events provide an object-oriented mechanism for a control to communicate with other controls that care to know about state changes within that control. If events did not exist, objects would have to resort to polling to know about state changes in other objects. The asynchronous nature of events provides an elegant means for communicating between objects. Event handler methods are generally protected to the control class (the event subscriber), as it would not make sense to call event handlers outside the consuming class.

The Page class in the “Hello, World” example consumes the Click event raised by the Button to read values and sets the first Label control. The Button Click event is easy to map in the Designer by simply double-clicking the button. Double-clicking a control in Visual Studio automatically generates the default event handler for the control. In the case of the Button, it is the Click event. In addition, Visual Studio performs other housekeeping tasks, such as wiring up the event delegate exposed by the Button control to the generated method (in this case, Button1_Click) in the Page class.

Note In the .NET Framework 2.0 and later, the concept of a partial class exists where a class can be split across multiple files. This allows Visual Studio or similar non-Microsoft tools to provide better design-time support.

Events in ASP.NET take advantage of delegates as the infrastructure for this communication among objects. In Chapter 5, we discuss how to work with events in detail.

The Properties window in the Design view of the Visual Studio Designer can help map the events from a control that don’t result from double-clicking the control.

Note Click the yellow lightning bolt icon at the top of the Properties window to filter the view to show only events exposed by a particular control.

Each available event for a control is listed on a separate line, and creating a wired up event handler is as simple as either double-clicking the blank area next to the event name to generate an event with the default naming scheme (ControlName_EventName) or typing a name and pressing the Enter key. Figure 1-5 illustrates creating the event handler for the TextBox control.

The end result of using the Properties window to add the protected event handler to the Page class is a method named TextBox_TextChanged that is wired to the TextChanged event of the TextBox control. You can add code to this handling routine to announce the state change of the TextBox control by setting the Text property of the Label2 control on the web form:

```
protected void Name_TextChanged(object sender, EventArgs e)
{
    ChangeLabel1.Text = "Textbox changed to " + Name.Text;
}
```

Visual Studio 2008 provides much cleaner code generation when compared to Visual Studio .NET 2003. There is no longer a code region named “Web Form Designer generated code” present in the code file. Much of the boilerplate code that existed in ASP.NET 1.1 is no longer present, which makes developers’ lives a bit simpler.

The result of all the not-so-hard work to this point is the browser view in Figure 1-6, which shows what happens when Rob enters his name and selects a polite greeting.

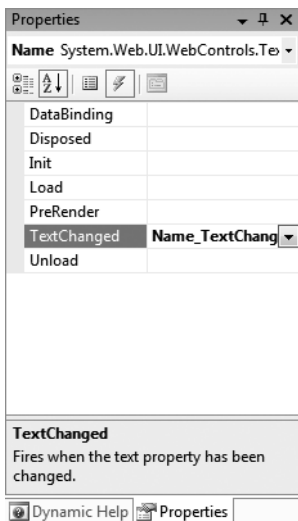


Figure 1-5. Adding an event handler to the TextChanged event of the TextBox control

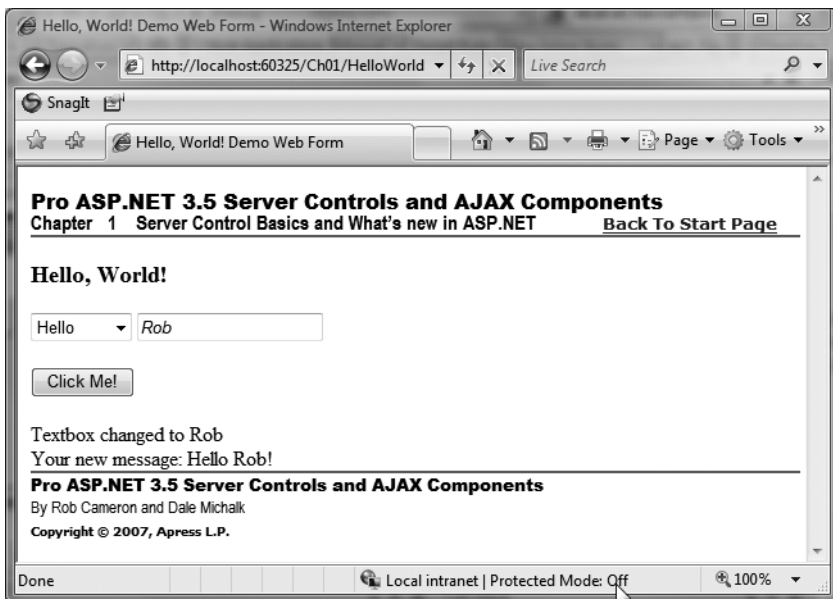


Figure 1-6. The completed “Hello, World” demonstration web form

The Web Page As a Control Tree

ASP.NET provides full programmatic access to the tags on an HTML page in an object-oriented way. The architecture in ASP.NET that provides this capability is the .aspx page control tree. In this section, we discuss the control tree as it relates to the “Hello, World” example.

At first glance, the “Hello, World” web form would seem to contain only a few visible server controls that were explicitly placed on the form. The reality is that the entire display surface of the .aspx page becomes a cornucopia of controls during processing. Any HTML content in the web form that is not part of the server controls laid out in the Visual Studio Designer is packaged into a server control that renders the HTML. The control structure of the web form can be seen by turning on the trace features of ASP.NET through setting the `Trace=True` attribute on the `Page` directive:

```
<%@ Page Language="C#" Trace="true"
    MasterPageFile="~/Master Page/ControlsBook2MasterPage.master"
    AutoEventWireup="true" CodeFile="HelloWorld.aspx.cs"
    Inherits="Ch01_HelloWorld" Title="Ch01 Hello World!" %>
```

You no longer need to make sure that tracing is enabled in the `<trace>` XML element inside of the `web.config` configuration file for the web application with .NET Framework 2.0 and later. However, if you wish to enable and customize the trace functionality, you have to paste the element within the `<system.web>` element of the `web.config` file for the application:

```
<trace
enabled="true"
requestLimit="10"
pageOutput="false"
traceMode="SortByTime"
localOnly="true"
/>
```

Figure 1-7 shows the portion of the trace output that displays the control tree for the web form.

Control UniqueID	Type	Render Size Bytes (including children)	ViewState Size Bytes (excluding children)	Control State Size Bytes (excluding children)
Page	ASP.ch01_helloworld_aspx	4610	0	0
ctl00	ASP.masterpage_controlsbook2masterpage_master	4610	0	0
ctl00\$ctl04	System.Web.UI.LiteralControl	172	0	0
ctl00\$ctl00	System.Web.UI.HtmlControls.HtmlHead	217	0	0
ctl00\$ctl01	System.Web.UI.HtmlControls.HtmlTitle	47	0	0
ctl00\$ctl02	System.Web.UI.HtmlControls.HtmlLink	79	0	0
ctl00\$ctl03	System.Web.UI.HtmlControls.HtmlLink	74	0	0
ctl00\$HeadSection	System.Web.UI.WebControls.ContentPlaceholder	4	0	0

Figure 1-7. Tracing the control tree of the “Hello, World” web form

The X-ray vision into ASP.NET provided by the trace feature dissects the web form in gory detail. At the top is the `Page` control that represents the web form of type `ASP.ch01_helloworld_aspx`. Below it are the server controls that you would expect to be there: `DropDownList`, `TextBox`, `Button`, and `Label`. What you might not expect to see are the `HtmlForm`, `DataBoundLiteralControl`, and `LiteralControl` objects in the control tree trace.