

Microsoft®

Expression® Web 4

Second Edition



Updated for
Service Pack 2:
HTML5,
CSS 3, jQuery

IN DEPTH

que®

Jim Cheshire



Microsoft®

Expression® Web 4

IN DEPTH

Second Edition

Jim Cheshire

que®

800 East 96th Street
Indianapolis, Indiana 46240

MICROSOFT® EXPRESSION® WEB 4 IN DEPTH, SECOND EDITION

Copyright © 2011 by Que Publishing

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-7897-4919-2

ISBN-10: 0-7897-4724-34919-X

Library of Congress Cataloging-in-Publication data is on file.

First Printing: June 2012

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Que Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Microsoft is a registered trademark of Microsoft Corporation.

Expression is a registered trademark of Microsoft Corporation.

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Bulk Sales

Que Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside the United States, please contact

International Sales

international@pearson.com

Editor in Chief

Greg Wiegand

Executive Editor

Loretta Yates

Development Editor

Todd Brakke

Managing Editor

Sandra Schroeder

Project Editor

Seth Kerney

Indexer

Cheryl Lenser

Technical Editor

Kathleen Anderson

Publishing Coordinator

Cindy Teeters

Book Designer

Anne Jones

Compositor

Bronkella Publishing, Inc.

CONTENTS AT A GLANCE

Introduction 1

I An Overview

1 An Overview of Expression Web 4 5

II Creating Content in Expression Web 4

2 Creating, Opening, and Importing Sites 25

3 Creating Pages and Basic Page Editing 47

4 Using Page Views 69

5 Using Tables 87

6 Using Frames 109

7 Editing Tag Properties 123

8 Using the Quick Tag Tools 133

9 Using Graphics and Multimedia 145

10 Using Find and Replace 171

11 Configuring Page Editor Options 187

12 Maintaining Compatibility and Accessibility 201

13 Using SuperPreview 217

III Publishing and Managing Websites

14 Publishing a Site 235

15 Site Management and Reporting 255

16 Using Personal Web Packages 267

IV Using CSS in Expression Web 4

17 Creating Style Sheets 273

18 Managing CSS Styles 293

V Scripting, DHTML, and Other Dynamic Content

19 Using Dynamic Web Templates 317

20 Using Interactive Buttons 333

21 Using Behaviors 343

22 Client Scripting 359

23 Using Layers 379

24 Using Form Controls 395

VI ASP.NET and PHP Development

25 Using Standard ASP.NET Controls 411

26 Using ASP.NET Navigation Controls 441

27 Using ASP.NET Master Pages and User Controls 461

28 Developing a Login System Using ASP.NET 477

29 Form Validation Using ASP.NET 511

30 Using ASP.NET Web Parts 523

31 Using ASP.NET Ajax 541

32 Using PHP 553

33 Using the Microsoft Expression Development Server 579

VII Managing Data with ASP.NET

- 34** Displaying and Editing Database Data with ASP.NET 587
- 35** Sending Email Using ASP.NET 605

VIII Creating Add-Ins Using JavaScript and HTML (Online Only)

- 36** Expression Web 4 Add-In Basics 615
- 37** Creating and Manipulating an Add-In User Interface 661
- 38** Packaging, Testing, and Debugging Add-Ins 689
- 39** Expression Web 4 JavaScript API Reference 701
- Index 757

CONTENTS

Introduction 1

Who Should Read This Book? 1

Why Use Expression Web? 1

How This Book Is Organized 2
Special Elements 3

I An Overview

1 An Overview of Expression Web 4 5

The Expression Web Interface 5

Panels 5

The Folder List 7

The Design Surface 8

The Status Bar 9

Working with Sites 9

Creating Sites 9

Site Reports 10

Publishing Sites 10

Tools for Creating Pages 11

Dynamic Web Templates 11

Find and Replace 11

Editing Tag Properties 12

Quick Tag Tools 13

Powerful CSS Tools 13

Style Builder 13

Manage Styles Panel 14

Apply Styles Panel 14

Site Optimization 16

Accessibility Checker 16

Compatibility Checker 16

SEO Checker 17

SuperPreview 17

CSS Reports 17

Scripting and Dynamic Content 18

Interactive Buttons 19

Behaviors 19

Layers 19

Photoshop Content 21

Deep Zoom Images 21

Video 21

ASP.NET Controls 22

PHP Support 22

Data Access Features 23

Creating Add-Ins 23

II Creating Content in Expression Web 4

2 Creating, Opening, and Importing Sites 25

What Is a Site? 25

How Expression Web Maintains a Site 25

Sites and Subsites 27

Site Templates 27

Types of Sites 29

Disk-Based Sites 29

FTP Sites 32

HTTP Sites 35

Importing Sites with the Import Site Wizard 38

Choosing an Import Method 38

Specifying a Destination Web

Location 44

Finishing the Site Import Process 44

Deciding on a Site Type 45

3 Creating Pages and Basic Page Editing 47

- Creating Pages 47
 - General Page 47
 - ASP.NET Pages 49
 - CSS Layouts 50
 - Style Sheets 50
 - Frames Pages 51
- Importing Files 51
- Formatting Text 52
 - How Expression Web Formats Text 52
 - Font Families 54
 - Font Sizes 55
- Creating Hyperlinks 55
 - Targeting Hyperlinks 56
 - Hyperlink Parameters 56
 - HTML Bookmarks 57
 - Hyperlink Screentips 58
- Spell-Checking 59
- Configuring Page Properties 60
 - General Tab 60
 - Formatting Tab 61
 - Advanced Tab 61
 - Custom Tab 62
 - Language Tab 63
- Using Code Snippets 63
- Configuring File Editors 65
- Making the Most of Code Snippets 68

4 Using Page Views 69

- Introduction to Page Views 69
- Working in Design View 70
 - Visual Aids 70
 - Ruler and Grid 74
 - Tracing Images 76
 - Adjusting Page Size 78

- Working in Code View 79
 - Customizing Code Formatting 81
 - Quick Tag Tools 83
 - IntelliSense 83
 - Context Menu 84
 - Bookmarks 84
- Working in Split View 85
- Taking Advantage of Page Views 86

5 Using Tables 87

- The Origin of Tables 87
- The Makeup of an HTML Table Tag 88
 - The align Attribute 88
 - Table Borders 90
 - The cellpadding Attribute 91
 - The cellspacing Attribute 92
 - The frame Attribute 93
- Rows, Columns, and Cells 93
 - The colspan and rowspan Attributes 94
 - Aligning Content in Cells 95
 - middle 95
- Tables in Expression Web 96
 - Inserting Tables 97
 - Customizing Tables 97
- Designing for Multiple Resolutions Using Tables 106

6 Using Frames 109

- Using Frames in Sites 109
 - When to Use Frames 109
 - When Not to Use Frames 110
- Creating Frames Pages 110
- Configuring Frames 112
 - Splitting Frames 113
 - Deleting Frames 114
- Creating Alternative Content 114

- Targeting Frames 115
- Adding and Configuring Inline Frames 117
- Tips for Frames 119
 - Frame Borders 119
 - Resizable Frames 120
 - Breaking Out of a Frameset 121
- 7 Editing Tag Properties 123**
 - An Introduction to Tag Properties 123
 - Understanding the Tag Properties Panel 123
 - Viewing Tag Properties with the Tag Properties Panel 126
 - Setting Tag Attributes with the Tag Properties Panel 128
 - Creating a Page 128
 - Setting Tag Properties 128
 - Using Events with the Tag Properties Panel 130
 - Tag Properties and Web Standards 131
- 8 Using the Quick Tag Tools 133**
 - Introduction to the Quick Tag Tools 133
 - Locating and Selecting Elements Using the Quick Tag Selector 134
 - Editing Page Content Using the Quick Tag Editor 136
 - Editing a Tag 137
 - Removing a Tag 138
 - Inserting HTML 140
 - Wrapping a Tag 141
 - Controlling Positioning 142
 - Editing Tag Properties 142
 - When to Use the Quick Tag Editor 142
- 9 Using Graphics and Multimedia 145**
 - Web Image Formats 145
 - The GIF Format 145
 - The JPEG Format 146
 - The PNG Format 146
 - Inserting Images 146
 - Formatting Images 148
 - Resizing Images 148
 - Changing Picture Properties 149
 - Converting Images 151
 - Creating Image Thumbnails 153
 - Configuring Auto Thumbnails 155
 - Creating Image Maps 156
 - Inserting Multimedia 157
 - Inserting Flash Movies 157
 - Inserting Silverlight Applications 159
 - Inserting Silverlight Video 160
 - Inserting Deep Zoom Images 161
 - Inserting Windows Media 163
 - Importing Adobe Photoshop Files 167
 - Page Transitions 168
 - Serving Video 169
- 10 Using Find and Replace 171**
 - An Introduction to Find and Replace 171
 - Finding and Replacing Text 172
 - Using Regular Expressions 172
 - Finding Text 173
 - Replacing Text 176
 - Using HTML Rules in Find and Replace 179
 - Finding and Replacing HTML Tags 181
 - Saving Queries 182
 - Editing and Removing Recent Searches 183

11 Configuring Page Editor Options 187

Accessing Page Editor Options 187

Exploring Page Editor Options 187

The General Tab 187

AutoThumbnail Tab 191

Default Fonts Tab 191

Code Formatting Tab 191

CSS Tab 191

Color Coding Tab 193

Authoring Tab 193

Picture Tab 195

Code Snippets Tab 195

Ruler and Grid Tab 195

IntelliSense Tab 197

Font Families Tab 197

Experiment to Learn More 199

12 Maintaining Compatibility and Accessibility 201

An Introduction to Accessibility 201

Designing for Accessibility 202

Accessible Hyperlinks 202

Accessible Tables 203

Accessible Frames 204

Other Accessibility Considerations 204

Using the Accessibility Checker 205

Checking Accessibility 205

Working with the Accessibility Panel 206

Generating Accessibility Reports 208

Designing for Compatibility 209

What Is Browser Compatibility? 209

Compatibility Features in Expression Web 210

Identifying Code Problems 210

Marking Invalid Code 213

Using Reports to Find Problems 214

Seeing Color 216

13 Using SuperPreview 217An Overview of SuperPreview 217
SuperPreview 217

How SuperPreview Generates a Preview 218

The SuperPreview Interface 220

Pointer Modes 220

DOM Highlighting 220

UI Helpers 220

Layout Modes 221

Preview URL 221

Baseline and Comparison Browser Selectors 222

DOM Tab 222

Browser Size Drop-Down 224

Using SuperPreview to Preview Layout 224

Setting Up the Previews 224

Generating Previews 225

Fixing Rendering Problems 225

Using the Snapshot Panel 229

Using Remote Browsers 230

Building Layouts with SuperPreview 232

III Publishing and Managing Websites**14 Publishing a Site 235**

What Is Publishing? 235

Server Options for Publishing 236

FTP 236

FrontPage Server Extensions 237

WebDAV 239

File System 240

Publishing Content 241

Configuring a Publishing Destination and Publishing a Site 241

Publishing Selected Files and

- Synchronizing Files 244
- Optimizing HTML During Publishing 244
- Troubleshooting HTTP Publishing 246

Hosting Your Site 253

15 Site Management and Reporting 255

Site Settings 255

- General Tab 255
- Preview Tab 256
- Advanced Tab 257
- Publishing Tab 258

Site Reports 259

- Configuring Reports 261
- Saving Reports 263

Using SEO Reports to Increase Traffic 263

16 Using Personal Web Packages 267

What Are Web Packages? 267

Creating a Web Package 267

Importing a Web Package 270

Capitalizing on Web Packages 272

IV Using CSS in Expression Web 4

17 Creating Style Sheets 273

An Introduction to CSS 273

The Purpose of CSS 274

How CSS Is Applied to Pages 277

- External Style Sheets 277
- Embedded Style Sheets 277
- Inline Styles 278

Formatting Content with CSS 278

Positioning Content with CSS 284

CSS Classes 287

- Basic Application of a CSS Class 287
- Applying Multiple CSS Classes 289
- Pseudo-Classes 290
- Pseudo-Elements 291

Centering a DIV with CSS 292

18 Managing CSS Styles 293

Expression Web's CSS Tools 293

- Apply Styles Panel 293
- Manage Styles Panel 293
- Link Style Sheet Dialog 295
- CSS Properties Panel 295
- CSS Reports 296
- Style Builder 297

Working with Styles 298

- Using the Manage Styles Panel 299
- Using the Apply Styles Panel 304
- Using the CSS Properties Panel 307
- Using the Style Builder 310
- Using the Attach Style Sheet and Link Style Sheet Dialogs 312

CSS Reports 313

- Checking for CSS Errors 313
- Checking CSS Usage 315

Arranging CSS Styles 315

V Scripting, DHTML, and Other Dynamic Content

19 Using Dynamic Web Templates 317

An Introduction to Dynamic Web Templates 317

Creating a Dynamic Web Template 318

- Creating a Page Layout 319
- Adding Editable Regions 319

Attaching a Dynamic Web Template 322

- Attaching to an Existing Page 322
- Attaching to a New Page 324

- Updating a Site with Dynamic Web Templates **324**
 - Modifying a Dynamic Web Template **324**
 - Modifying an Attached Page in Code View **325**
- Managing Editable Regions **327**
 - Adding a New Editable Region **327**
 - Renaming an Existing Editable Region **328**
 - Resolving Mismatched Editable Regions **329**
- Detaching a Dynamic Web Template **330**
- Under the Hood **331**
- 20 Using Interactive Buttons 333**
 - Overview of Interactive Buttons **333**
 - Inserting and Configuring Interactive Buttons **334**
 - The Button Tab **334**
 - The Font Tab **335**
 - The Image Tab **337**
 - Saving an Interactive Button **339**
 - Editing an Interactive Button **340**
 - Practical Uses for Interactive Buttons **341**
- 21 Using Behaviors 343**
 - Understanding and Working with Behaviors **343**
 - Using the Behaviors Panel **343**
 - How Behaviors Work **344**
 - Adding Behaviors Within a Paragraph **344**
 - Expression Web Behaviors **345**
 - The Call Script Behavior **345**
 - The Change Property Behavior **346**
 - The Change Property Restore Behavior **349**
 - The Go To URL Behavior **349**
 - The Jump Menu Behavior **350**
 - The Jump Menu Go Behavior **350**
 - The Open Browser Window Behavior **351**
 - The Play Sound Behavior **352**
 - The Popup Message Behavior **353**
 - The Preload Images Behavior **353**
 - The Set Text Behavior **354**
 - The Swap Image Behavior **357**
 - The Swap Image Restore Behavior **357**
 - When to Use Behaviors **358**
- 22 Client Scripting 359**
 - A History of Browser Scripting **359**
 - JavaScript Basics **361**
 - Adding JavaScript to a Page **361**
 - Linking to an External Script File **362**
 - Adding Inline JavaScript **363**
 - The Document Object Model **363**
 - The window Object **364**
 - The document Object **365**
 - Writing Simple Scripts **366**
 - Showing and Hiding Page Elements **366**
 - Accessing and Changing Attributes **371**
 - Form Field Validation **373**
 - Debugging **377**
- 23 Using Layers 379**
 - Introduction to Layers **379**
 - Inserting and Configuring Layers **380**
 - Adding Content to a Layer **381**
 - Resizing a Layer **382**
 - Creating and Working with Child Layers **383**
 - Positioning Layers **386**

Setting Layer Properties with Behaviors **387**
 Setting the Visibility of Layers **387**
 Adding Layer Interactivity **388**

Z-Order Anomalies **393**

24 Using Form Controls 395

Understanding HTML Forms **395**

Using Form Controls in Expression Web **396**

Creating a Form **397**

Saving Form Results to a File or Email **400**

 File Results Tab **401**

 Email Results Tab **402**

 Confirmation Page Tab **404**

 Saved Fields Tab **406**

Saving Form Results to a Database **407**

 Updating a Database with New Fields **409**

Hidden Form Fields **410**

VI ASP.NET and PHP Development

25 Using Standard ASP.NET Controls 411

ASP.NET: A Brief Introduction **411**

Creating ASP.NET Pages **412**

The Basics of ASP.NET Controls **414**
 Understanding Control Properties **415**

An Overview of the Standard ASP.NET Controls **419**

The AdRotator Control **420**
 Creating a Simple AdRotator Page **422**
 Creating the Advertisement File **422**

The Calendar Control **426**
 Formatting the Calendar Control **426**
 Calendar Control Properties **427**

The Wizard Control **431**
 Wizard Steps **431**
 Creating a Simple Wizard **432**

Making ASP.NET Work for You **439**

Formatting with Styles **439**

26 Using ASP.NET Navigation Controls 441

Overview of Navigation Systems **441**

Creating a Sitemap File **443**

Using the ASP.NET Menu Control **444**

 Creating a Test Site **445**

 Adding a Menu Control **446**

 Formatting the Menu Control **448**

Using the ASP.NET TreeView Control **451**

 Formatting the TreeView Control **453**

Using the ASP.NET SiteMapPath Control **457**

 Formatting the SiteMapPath Control **458**

Improving Navigation with Master Pages **459**

27 Using ASP.NET Master Pages and User Controls 461

The Need for a Common Layout **461**

The Master Page **461**

The Content Page **464**

Developing a Master Page Site **467**
 Creating the Master Page **467**
 Creating the Content Page **473**

Extend Reusability with ASP.NET User Controls **475**

28 Developing a Login System Using ASP.NET 477

Website Login Systems 477

Overview of ASP.NET Login Controls 478

The Login Control 478

LoginStatus Control 482

LoginName Control 482

ChangePassword Control 482

PasswordRecovery Control 487

CreateUserWizard Control 488

LoginView Control 489

Creating a Login Solution 491

Configuring the Website (IIS 5 or IIS 6) 492

Configuring the Website (IIS 7.x) 494

Configuring the Website (Microsoft Expression Development Server) 502

Creating the Web Pages 504

Using Web Deploy to Publish a Membership Database 507

29 Form Validation Using ASP.NET 511

The Need for Form Validation 511

The ASP.NET Validation Controls 511

Common Properties 513

Creating a Validated Form 515

Validation Groups 521

30 Using ASP.NET Web Parts 523

An Introduction to Web Parts 523

Creating ASP.NET User Controls 524

Web Parts Controls in the Toolbox 525

Creating a Web Parts Page 526

Web Parts Page Display Modes 529

Creating a User Control That Sets the Display Mode 530

Adding Code to Change the Display Mode 531

The Web Parts Catalog 534

Editing Web Parts Controls 537

31 Using ASP.NET Ajax 541

What Is Ajax? 541

Microsoft's ASP.NET Ajax 541

Client-Side Ajax 542

Server-Side Ajax 542

Microsoft Ajax Control Toolkit 543

Adding Ajax Functionality to a Web Form 543

Creating a Site and Page 543

Adding Server-Side Code 545

Adding a ScriptManager Control 546

Adding an UpdatePanel Control 548

Using Client-Side Ajax 549

Adding a <div> to the Web Form 549

Creating the Client Library 549

Adding the Client Script to the ScriptManager Control 550

32 Using PHP 553

An Introduction to PHP 553

PHP Syntax 553

Installing PHP 557

Installing IIS 7 and FastCGI 557

Installing PHP 559

Configuring the Microsoft Expression Development Server for PHP 560

Enabling PHP for IIS Using FastCGI 561

Creating PHP Pages 562

Previewing the Page 563

PHP in Design View	566	ASP.NET and Other Web Application Platforms	588
PHP in Code View	568	ASP.NET Data Source Controls	589
PHP Syntax Highlighting	568	AccessDataSource Control	589
Using IntelliSense with PHP	568	SqlDataSource Control	591
Setting PHP-Specific IntelliSense Options	570	SiteMapDataSource Control	591
PHP Script Options	572	XmlDataSource Control	592
Form Variable	572	Displaying Data with ASP.NET	592
URL Variable	573	Displaying Data in Tabular Form	592
Session Variable	573	Sorting the GridView	595
Cookie Variable	574	Editing Data with ASP.NET	597
Include Once	574	Configuring the Data Source	597
Code Block	575	Configuring the GridView	598
Comment	576	Testing the Page	599
Displaying PHP Information	577	Creating a Master/Detail View	600
33 Using the Microsoft Expression Development Server	579	Creating the Master View	600
Introduction to the Microsoft Expression Development Server	579	Creating the Detail View	601
How to Use the Microsoft Expression Development Server	581	35 Sending Email Using ASP.NET	605
Limitations of the Microsoft Expression Development Server	583	A Typical Contact Form	605
Process Identity	583	Creating the Contact Form	606
No Remote Access	584	Adding and Configuring ASP.NET Validation Controls	608
No Support for ASP Pages	584	Adding the Validation Controls	608
Starting the Microsoft Expression Development Server from the Command Prompt	585	Writing ASP.NET Code to Send Email	611
VII Managing Data with ASP.NET		Displaying a Confirmation Page	613
34 Displaying and Editing Database Data with ASP.NET	587	VIII Creating Add-Ins Using JavaScript and HTML (Online Only)	
A History of Data Access	587	36 Expression Web 4 Add-in Basics	615
Data Access Technologies in Expression Web	588	Add-ins in Expression Web	615
		Expression Web 4 JavaScript Add-ins	616
		The Makeup of Expression Web Add- ins	616

- XML Basics 617
- General Manifest Elements and Attributes 618
 - src (optional) 618
 - legacy (optional) 619
 - developer (optional) 619
 - navigationalallowed (optional) 619
 - <name> (required) 620
 - <description> (optional) 620
 - <author> (optional) 620
 - <version> (optional) 621
 - <homepage> (optional) 621
 - <minversion> (optional) 621
 - <guid> (optional) 621
 - <load> (optional) 622
- Commands and Dialog Boxes 622
 - id (required) 622
 - filetype (optional) 622
 - onclick (optional) 623
- Menus and Toolbars 624
 - <menuitem> (optional) 624
 - <toolbaritem> (optional) 626
- Panels 627
 - <panel> (optional) 628
- Menu and Command Bar Reference 630
 - Menus 631
 - Toolbars 650
- 37 Creating and Manipulating an Add-in User Interface 661**
 - Planning an Add-in 661
 - Creating the Manifest with the Add-in Builder 662
 - Creating the Manifest 662
 - Editing the Manifest 667
 - Creating the User Interfaces 667
 - Creating a Custom Page Size for Panels 668
 - Creating the Panel's Interface 669
 - Creating the Options Dialog Interface 672
 - Adding Functionality with JavaScript 673
 - JavaScript for panel.htm 673
 - JavaScript for options.htm 679
 - The Set Page Title Dialog 680
 - Accessing Managed Classes from JavaScript 682
 - Creating a Managed Class 683
 - Editing the Add-in Manifest to Load the Managed Class 686
 - Calling the Managed Class 687
 - Summary 687
- 38 Packaging, Testing, and Debugging Add-ins 689**
 - Creating an Add-in Installation Package 689
 - Testing and Debugging Add-ins 690
 - Testing Add-ins 690
 - Debugging Add-ins Using Expression Web 691
 - Debugging Add-ins Using Visual Studio 695
 - Summary 699
- 39 Expression Web 4 JavaScript API Reference 701**
 - Conventions Used in this Reference 701
 - xweb.application Object 702
 - xweb.application.version Property 702
 - xweb.application.chooseFile Method 703
 - xweb.application.endDialog Method 704
 - xweb.application.handleEvent Method 705

- xweb.application.newDocument Method **706**
- xweb.application.openDocument Method **708**
- xweb.application.refreshFileListing Method **708**
- xweb.application.setActiveDocument Method **709**
- xweb.application.setPanelVisibility Method **710**
- xweb.application.showModalDialog Method **711**
- xweb.application.settings Object **712**
 - xweb.application.settings.read Method **713**
 - xweb.application.settings.write Method **713**
- xweb.developer Object **714**
 - xweb.developer.write Method **714**
 - xweb.developer.writeLine Method **715**
- xweb.document Object **715**
 - xweb.document.anchors Property **717**
 - xweb.document.applets Property **717**
 - xweb.document.embeds Property **718**
 - xweb.document.filename Property **718**
 - xweb.document.forms Property **719**
 - xweb.document.frames Property **719**
 - xweb.document.images Property **720**
 - xweb.document.isXHTML Property **720**
 - xweb.document.links Property **720**
 - xweb.document.location Property **721**
 - xweb.document.name Property **722**
 - xweb.document.pathFromSiteRoot Property **722**
 - xweb.document.scripts Property **723**
 - xweb.document.selection Property **723**
 - xweb.document.appendScriptReference Method **724**
 - xweb.document.appendStyleReference Method **725**
 - xweb.document.close Method **726**
 - xweb.document.getElementById Method **726**
 - xweb.document.getElementsByAttribute Name Method **727**
 - xweb.document.getElementsByTagName Method **728**
 - xweb.document.getScriptElementByCode Method **728**
 - xweb.document.getScriptElementByFile Method **729**
 - xweb.document.getStyleElementByCode Method **730**
 - xweb.document.getStyleElementByFile Method **730**
 - xweb.document.insertBeforeHtml Method **731**
 - xweb.document.save Method **732**
 - xweb.document.saveAs Method **732**
 - xweb.document.synchronizeViews Method **733**
- xweb.file Object **733**
 - xweb.file.copy Method **734**
 - xweb.file.createFile Method **735**
 - xweb.file.createFolder Method **735**
 - xweb.file.deleteFile Method **736**
 - xweb.file.exists Method **737**
 - xweb.file.getAttributes Method **737**
 - xweb.file.getCreationDate Method **738**
 - xweb.file.getModificationDate Method **738**
 - xweb.file.getSize Method **739**
 - xweb.file.listFolder Method **739**
 - xweb.file.read Method **740**
 - xweb.file.setAttributes Method **741**
 - xweb.file.write Method **742**
- htmlElement Object **742**
 - htmlElement.childNodes Property **742**

htmlElement.className Property	743
htmlElement.id Property	743
htmlElement.innerHTML Property	744
htmlElement.innerText Property	745
htmlElement.nextSibling Property	746
htmlElement.outerHtml Property	747
htmlElement.parentNode Property	748
htmlElement.previousSibling Property	748
htmlElement.tagName Property	749
htmlElement.getAttribute Method	749
htmlElement.removeAttribute Method	750
htmlElement.setAttribute Method	750
xweb.document.selection Object	751
selection.end Property	751
selection.start Property	751
selection.text Property	752
selection.append Method	752
selection.insert Method	753
selection.set Method	753
selection.remove Method	754
selection.replace Method	754
selection.wrap Method	755

Index 757

ABOUT THE AUTHOR

Jim Cheshire is the owner of Jimco Software and Books and is the author of several design books and books on the Amazon Kindle and Barnes and Noble Nook. In his real job, Jim works as a senior escalation engineer at Microsoft on the ASP.NET, IIS, and Expression Web teams. He has worked on the FrontPage, Visual Basic, ASP, IIS, and ASP.NET teams at Microsoft for almost 15 years.

You can reach Jim by visiting one of his websites: www.jimcobook.com or www.jimcosoftware.com. You can also email him at jwc@jimcobook.com.

DEDICATION

This book is dedicated to my lovely wife, Becky, and my two children. I love you all very much.

ACKNOWLEDGMENTS

I owe a debt of gratitude to my editors at Que Publishing. Loretta, it's been such a pleasure to work with you over the years. Todd, thanks for your consistent work to improve what comes out of my mind. To Kathleen and Ian, thank you for your hard work in ensuring that this book is technically accurate and easy to understand. To Seth, thanks for your commitment to a quality book. Although the cover of this book bears my name only, the book would have not been possible without the commitment of all of you, and I am sincerely thankful for all your hard work.

Thanks to Anna Ullrich, Paul Bartholomew, Justin Harrison, Steve Guttman, Erik Saltwell, Marc Kapke, Mike Calvo, and Erik Mikkelson at Microsoft, all of whom were of great help in answering questions that arose during the writing of this book. I also owe John Dixon at Microsoft a special thank you for always being available for questions about add-in development. Without John's assistance, the last part of this book simply wouldn't have been possible.

—Jim

WE WANT TO HEAR FROM YOU!

As the reader of this book, *you* are our most important critic and commentator. We value your opinion and want to know what we're doing right, what we could do better, what areas you'd like to see us publish in, and any other words of wisdom you're willing to pass our way.

As an editor in chief for Que Publishing, I welcome your comments. You can email or write me directly to let me know what you did or didn't like about this book—as well as what we can do to make our books better.

Please note that I cannot help you with technical problems related to the topic of this book. We do have a User Services group, however, where I will forward specific technical questions related to the book.

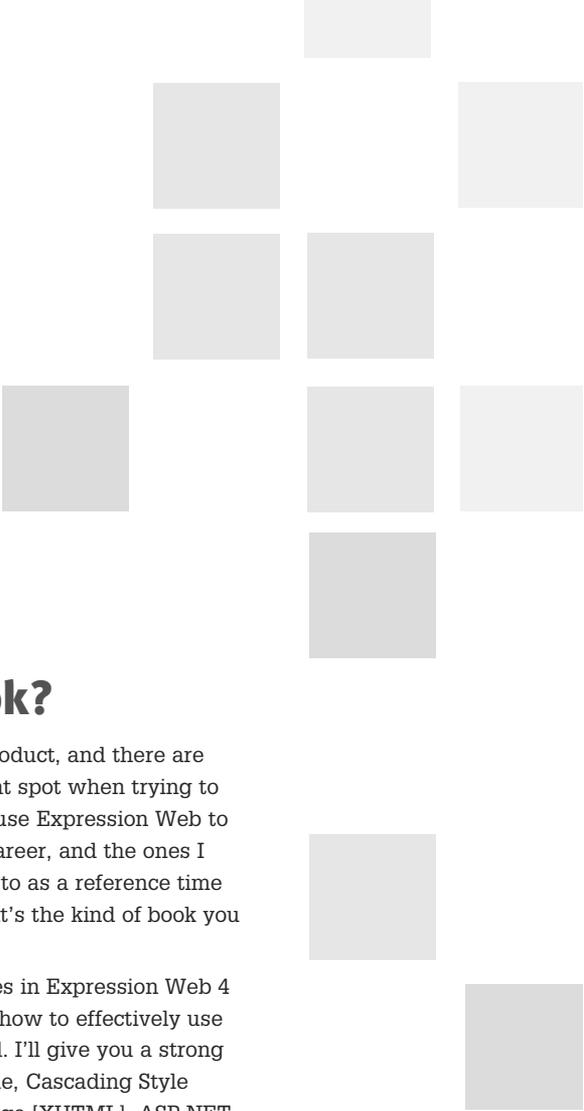
When you write, please be sure to include this book's title and author as well as your name, email address, and phone number. I will carefully review your comments and share them with the author and editors who worked on the book.

Email: feedback@quepublishing.com

Mail: Greg Wiegand
Editor in Chief
Que Publishing
800 East 96th Street
Indianapolis, IN 46240 USA

Reader Services

Visit our website and register this book at informit.com/register for convenient access to any updates, downloads, or errata that might be available for this book.



INTRODUCTION

Who Should Read This Book?

Expression Web is a powerful web development product, and there are plenty of books about it. That puts readers in a tight spot when trying to figure out which book to purchase to learn how to use Expression Web to its fullest. I've read a lot of technical books in my career, and the ones I value the most are the ones that I keep going back to as a reference time and again as I use the technology they teach. If that's the kind of book you appreciate, then this is the book for you.

In this book, I will show you how to use the features in Expression Web 4 for real-world web design. I'll give you pointers on how to effectively use several features together to achieve a common goal. I'll give you a strong foundation in the technologies involved (for example, Cascading Style Sheets [CSS], Extensible Hypertext Markup Language [XHTML], ASP.NET, PHP, and more) so you aren't working in a vacuum.

I'll wrap up the book by showing you how to use the new extensibility features in Expression Web 4 to develop add-ins using skills you likely already have as a web designer.

The goal of this book is to equip you to be a better web designer using Expression Web as one of the tools in your arsenal. If that sounds like something you're interested in, keep reading!

Why Use Expression Web?

Ask any professional web designer which features are important for a web design application and he will tell you that it must adhere to current web standards and make creating and maintaining a standards-driven site easy. Not only that, but a solid web design tool should be created from

the ground up to support dynamic content such as ASP.NET and PHP from a designer's perspective. Expression Web meets all those needs.

You'll not only be able to create dynamic, standards-compliant sites with Expression Web, but you'll also be able to do so in a way that enables you to take advantage of your creativity. It doesn't drag you down with complex dialogs and frustrating code changes. After using Expression Web for a while, you'll never consider going back to your previous web design tool.

How This Book Is Organized

This book is broken up into multiple parts so you can quickly find the information you need. Here is the rundown on all the sections:

- **Part I, “An Overview”**—Part I comprises an overview of Expression Web. You'll receive a complete tour of the Expression Web feature set, along with some tips and tricks on using the interface.
- **Part II, “Creating Content in Expression Web 4”**—The chapters in this part teach you how to create content and work with features in Expression Web that help you manage your page content and edit your pages.
- **Part III, “Publishing and Managing Websites”**—In this part, you'll learn about publishing sites, managing your sites, and generating reports on your sites.
- **Part IV, “Using CSS in Expression Web 4”**—Instead of including CSS as an afterthought, I've dedicated two complete chapters to using this important design concept. You'll learn not only the details of CSS in general, but also how to use the powerful CSS tools in Expression Web.
- **Part V, “Scripting, DHTML, and Other Dynamic Content”**—In this part, you'll learn tricks to differentiate your site from the status quo. You'll learn how to add dynamic components automatically in Expression Web. You'll also learn how to read and write client-side JavaScript so you can understand what goes on under the covers when Expression Web adds code to your page.
- **Part VI, “ASP.NET and PHP Development”**—Expression Web fully supports Microsoft ASP.NET and provides some support for PHP. In this part, you'll find out what ASP.NET gives you as a web designer and how to use Expression Web to create some pretty powerful site features, all without writing any code. You'll also learn how to take advantage of the PHP support included in Expression Web.
- **Part VII, “Managing Data with ASP.NET”**—In this part, you'll learn how to use ASP.NET to display and edit database data without having to write a bunch of code.
- **Part VIII, “Creating Add-Ins Using JavaScript and HTML”**—In this online-only section, you'll see how you can easily create add-ins to extend the functionality of Expression Web 4 using JavaScript and HTML.

As a purchaser of this book, you are also entitled to a free copy of my e-book *Creating Microsoft Expression Web 4 Add-ins*. After the e-book is released, you'll be able to find it, along with the code samples and sample files, on the website that accompanies this book at www.informit.com/title/9780789749192.

Special Elements

Throughout the book, you'll find some special elements that are designed to make it easy to locate important information or special tips to help you get the most out of Expression Web.

When an important term is used for the first time, it is printed in italics and defined close by. When instructions require you to enter text or values into a dialog, the data you are to enter appears bolded.

Cross-References

Nothing's worse than a technical book that assumes you will read it from beginning to end like a novel. Most folks use technical books like reference materials, so this book makes generous use of cross-references.

If a feature is mentioned that is covered elsewhere in the book, there is a cross-reference directing you to where you can find details on that topic.

Notes, Tips, Cautions, and Sidebars

You'll find numerous bits of information in these special elements.

Read Sidebars for the Big Picture

You won't find sidebars in every chapter; they're designed to give you more insight into a particular topic. If you're the kind of person who wants to know all the details, you'll find sidebars extremely valuable.



note

Notes include additional technical information or links to important information.



tip

Tips provide information to make using a feature easier or provide information you might not have considered.



caution

Caution elements prevent you from shooting yourself in the foot. They point out problems so you can avoid them and save yourself time and possible headaches.



Where applicable, I've included troubleshooting notes so that if you get into trouble, you can resolve any problems easily.

I sincerely hope you enjoy this book and find it to be an invaluable resource as you build sites with Expression Web 4.

Thank you for purchasing this book, and I hope you enjoy reading it as much as I enjoyed writing it for you!

—Jim

This page intentionally left blank

AN OVERVIEW OF EXPRESSION WEB 4

The Expression Web Interface

The true measure of any design tool is how well the interface contributes to your productivity. The Expression Web interface is designed specifically to make access to tools easy while maintaining maximum real estate for the design surface. The first thing you'll likely notice when opening Expression Web is that the interface is a dark color. This is due to an architectural change introduced in Expression Web 3. The interface color proved not to be popular with many Expression Web users, so Microsoft added the ability to use your Windows color scheme instead of the dark interface.

Major changes in Expression Web involve changes in publishing, search engine optimization (SEO) enhancements, remote browser capabilities in SuperPreview, and a new extensibility system that makes it easy to develop add-ins for Expression Web using JavaScript and HTML.

The next sections go over some of the major interface elements and features of Expression Web.



tip

To change Expression Web's interface to your Windows color scheme, select Tools, Application Options, and check the Use Your Current Windows Color Scheme check box.

Panels

Most of the tools you will use in Expression Web are available in panels, as shown in Figure 1.1.



Figure 1.1
Panels are a flexible and convenient way to access tools in Expression Web.

Panels can be hidden by clicking the AutoHide button, as shown in Figure 1.2. When you click it, a tab for the panel appears on the left or right side of the Expression Web interface. You can display the panel again by hovering over this tab. If you want to unhide the panel, click the pin icon again.

AutoHide button

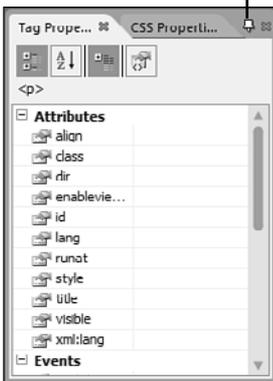


Figure 1.2
Clicking the AutoHide button hides the panel.

You can also drag a panel to any edge of the Expression Web interface to dock it on that edge or drag it away from an edge to make it a floating panel that can be positioned anywhere within the user interface.

Panels can contain tabs so that multiple panels can be present within the same window. Figure 1.3 shows the Manage Styles panel with several tabs inside it for other panels. To activate one of the other panels, simply click the tab. Arrow buttons are provided when there isn't enough room to display all tabs.

**tip**

You can double-click the top edge of a panel to toggle it between a floating panel and a docked panel.

Figure 1.3

Panels are extremely flexible. In this case, several panels are combined into one window. Tabs are provided for each panel.



To add a panel as a tab within another panel, simply drag one panel on top of another.

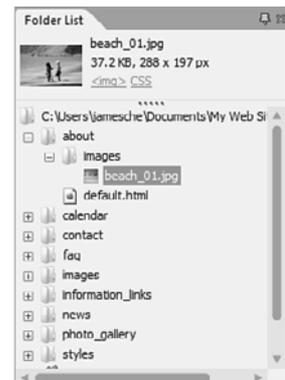
In Service Pack 2 for Expression Web 4, Microsoft added the capability to save your panel configuration in a *workspace*. You can then easily restore your panel layout by selecting it from the Panels menu. Expression Web 4 includes four pre-defined layouts, but you can save as many of your own layouts as you want. For example, you might want to create a layout that has all of Expression Web 4's CSS tools displayed so that you can focus on adding styles to your page.

The Folder List

Expression Web displays all the files and folders in the current site inside the Folder List panel, as shown in Figure 1.4. You can quickly locate any file or double-click a file to edit it.

Figure 1.4

The Folder List provides a view of all the files and folders in your site.



The Folder List has many of the same features you would expect in a file manager, such as creating new folders, adding files, deleting content, and so on. Right-click a file or folder in the Folder List for a menu of options.

The Folder List is similar to the Site View tab in the main Expression Web window, but the Site View tab allows for sorting and more customized views than the rudimentary view provided by the Folder List.

The Design Surface

The design surface (shown in Figure 1.5) is the area in Expression Web where you create your pages. The view of your site matches what you will see when you view the page in a browser.

Contrary to what some people believe, the design surface in Expression Web does not use Internet Explorer for rendering. It is a browser-independent rendering of your page, and it's designed to come as close as possible to what your page will look like in any browser. However, it's only a close approximation. You should always preview your page in a browser (or in SuperPreview) in order to verify its appearance.



caution

When deleting files from the Folder List, keep in mind that deleting some files (such as CSS files, templates, and script files) can affect many other files. There is no Recycle Bin in Expression Web, so be careful! When you delete a file from the Folder List, it's permanently deleted.



Figure 1.5

The design surface in Expression Web provides a close approximation of how a page will appear in your browser.

The Status Bar

The status bar in Expression Web is extremely informative (see Figure 1.6). It contains tools that warn you when incompatible or invalid code is detected in the page that is open in Expression Web. It also provides information at a glance concerning the schema of the page and so on.

Figure 1.6

The status bar is an important tool. It lets you know whether you have bad code in your page and informs you of many properties of a page.



➔ For more information on the status bar, see Chapter 4, “Using Page Views.”

Working with Sites

Expression Web can easily work with single pages, but to get the most out of it, you need to create a site, which is to say, a collection of pages.

Creating Sites

Expression Web can create websites on your local file system or on a remote file system or server. You can use the FrontPage Server Extensions if you want (although they are not required), and you can use FTP to create a website or open an existing one.

The Managed Sites dialog, shown in Figure 1.7, makes it easy to keep track of all your sites and provides for better management of local sites.



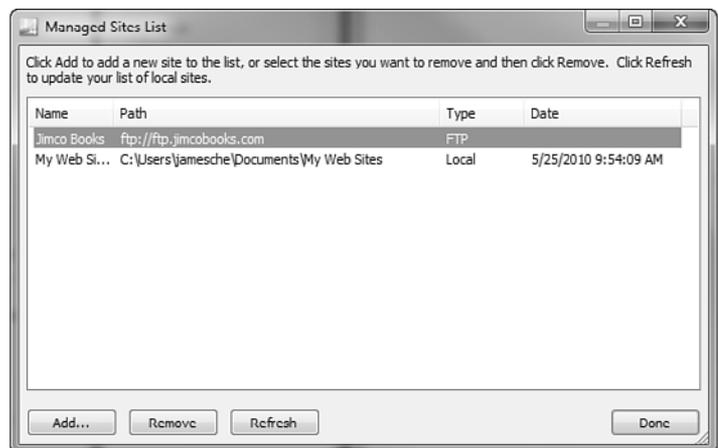
caution

Microsoft considers FrontPage Server Extensions to be deprecated technology. Therefore, you should seriously consider using FTP or secure FTP instead of the FrontPage Server Extensions.

See support.microsoft.com/kb/2610850 for more information.

Figure 1.7

The Managed Sites dialog lets you easily keep up with your sites.



➔ For more information on creating sites, see Chapter 2, “Creating, Opening, and Importing Sites.”

Site Reports

Expression Web has 17 reports that can be displayed for a site. Using these reports, you can easily identify problems such as slow pages, unlinked files, broken hyperlinks, and so on. You also can get a good view of your site in its entirety, as shown in Figure 1.8. You can access some reports by clicking the Reports tab at the bottom of the main Expression Web window when Site View is selected. Other reports are available on the Tools menu.

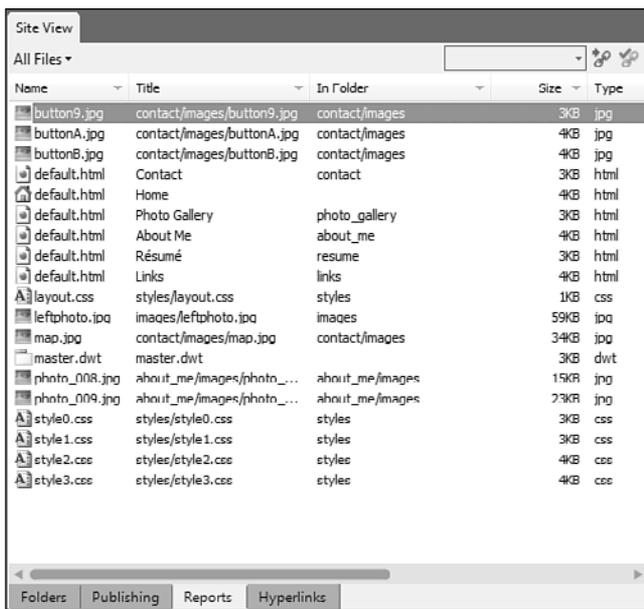


Figure 1.8

Expression Web can display many reports on your website. The All Files report is shown here.

➔ For more information on site reporting, see Chapter 15, “Site Management and Reporting.”

Publishing Sites

Expression Web has a robust file publishing system that can use the file system, FTP, or HTTP via the FrontPage Server Extensions. You can choose to publish only those files that have changed since the last time you published, or you can publish all files.

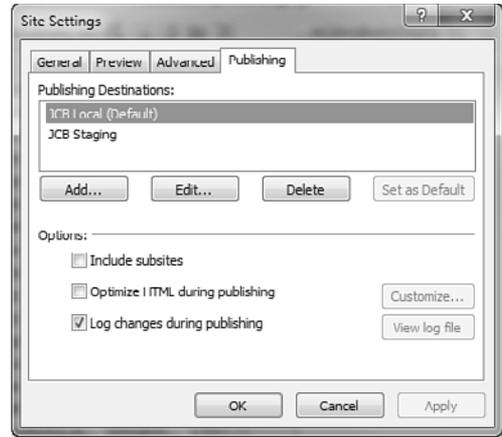
The Publishing tab in the Site Settings dialog (shown in Figure 1.9) allows you to save common publishing destinations and to easily specify how Expression Web chooses what to publish. Expression Web also allows you to publish securely using SFTP and FTPS.

note

Expression Web adds several publishing menu items to make publishing changed pages, selected files, or the current file easier.

Figure 1.9

The Publishing tab in the Site Settings dialog adds flexibility to publishing.



➔ For more information on publishing a site, see Chapter 14, “Publishing a Site.”

Tools for Creating Pages

Expression Web helps you create standards-compliant sites with minimal effort. Plenty of features are available to help make that possible.

Dynamic Web Templates

To help you create a site with a consistent look and feel across all pages, Expression Web offers Dynamic Web Templates. You can create any number of Dynamic Web Templates and attach pages of your choice to each.

Dynamic Web Templates also help you easily make site-wide changes. Simply modify a Dynamic Web Template and save it. When you do, pages attached to that template are updated automatically.

➔ For more information on using Dynamic Web Templates, see Chapter 19, “Using Dynamic Web Templates.”

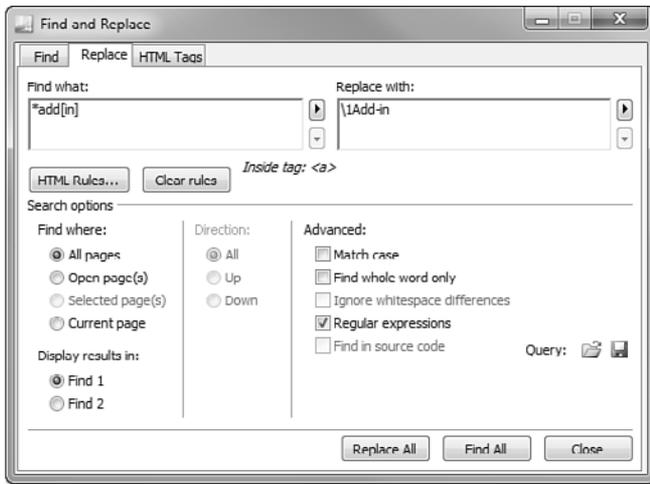
Find and Replace

A robust find and replace tool is an important tool for any web designer. Expression Web not only includes a flexible and powerful tool for find and replace (see Figure 1.10), but also incorporates regular expressions so you can create complex queries.

➔ For more information on using the Find and Replace tool, see Chapter 10, “Using Find and Replace.”

note

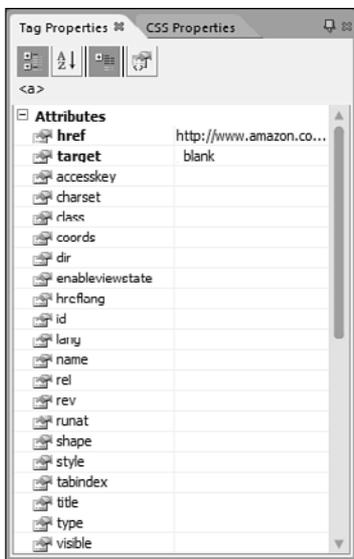
Expression Web isn't technically a WYSIWYG tool. The view you see in Design view is a close approximation of what you'll see in the browser, but it's not perfect.

**Figure 1.10**

The Find and Replace tool meets simple needs as well as complex ones. Regular expressions allow for complex searches.

Editing Tag Properties

It's never been easier to edit tag properties. The Tag Properties panel (see Figure 1.11) shows you all the properties of a selected HTML element in one location. By bolding properties that have been set, this tool lets you see how tags have been configured.

**Figure 1.11**

Tag properties are easy to examine and modify using the Tag Properties panel.

➔ For more information on using the Tag Properties panel, see Chapter 7, “Editing Tag Properties.”

Quick Tag Tools

The Quick Tag Tools are an easy way to select a specific page element (see Figure 1.12). You also can access properties of the selected element or modify HTML code directly.

Figure 1.12

Quick Tag Tools are the most convenient way to locate specific HTML elements. You can also modify properties of an element.



➔ For more information on using the Quick Tag Tools, see Chapter 8, “Using the Quick Tag Tools.”

Powerful CSS Tools

Expression Web contains some of the most powerful and flexible CSS tools available in any web design application. You can easily and quickly create CSS code and apply it to your pages using the Style Builder and the Manage Styles and Apply Styles panels. For analyzing how your CSS affects specific page elements, the CSS Properties panel is indispensable.

➔ For more information on using CSS in Expression Web, see Chapter 17, “Creating Style Sheets.”

Style Builder

The Style Builder is an easy-to-use interface for styling page elements using CSS (see Figure 1.13).

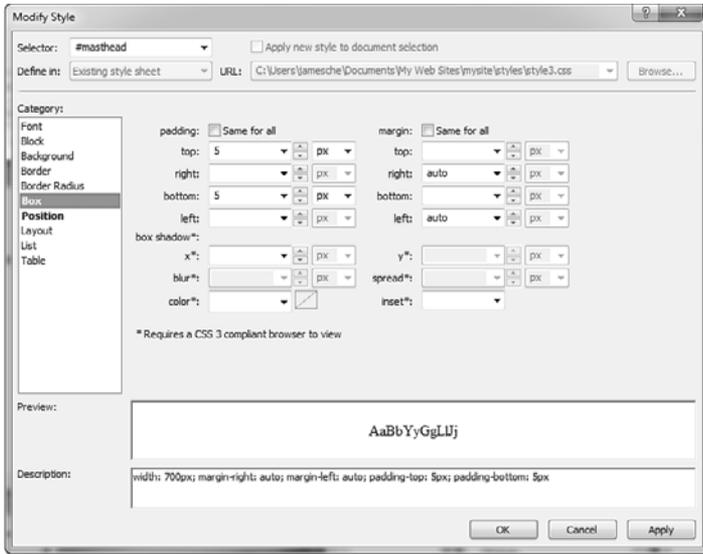


Figure 1.13
Creating and modifying styles is surprisingly easy with the Style Builder.

Manage Styles Panel

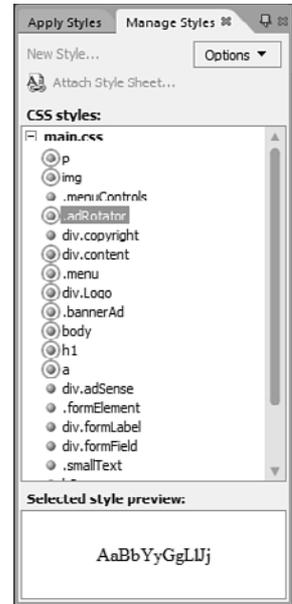
The Manage Styles panel, shown in Figure 1.14, shows CSS styles in one location. You can modify styles, move styles around, and apply styles using this panel.

Apply Styles Panel

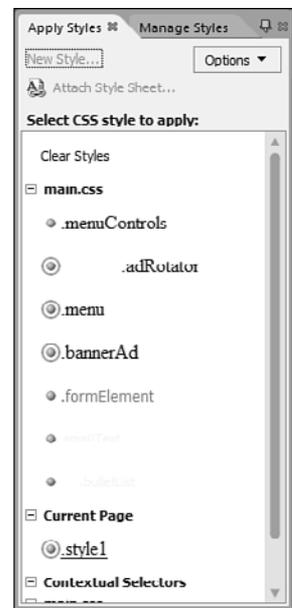
The Apply Styles panel complements the Manage Styles panel nicely (see Figure 1.15). One of the greatest features of the Apply Styles panel is its capability to easily help you determine which CSS styles apply to particular page elements.

Figure 1.14

The Manage Styles panel shows you all the styles available to you.

**Figure 1.15**

The Apply Styles panel is a complementary tool to the Manage Styles panel.



Site Optimization

Site optimization is an important part of developing any website. After you have developed your site, you need to clean up any CSS code problems, fix any browser incompatibilities, and ensure that your site is compliant with the latest accessibility standards. Expression Web offers tools for all those tasks.

Accessibility Checker

There are two standards for website compliancy: Section 508 and the Web Content Accessibility Guidelines (WCAG). Learning all the ins and outs of each would be a challenging task. Fortunately, Expression Web can run reports that show you how your site fares with both standards.

The Accessibility Checker feature (see Figure 1.16) highlights pages in your site that don't meet standards and enables you to correct problems before you deploy your site.



note

Expression Web will check your site only against WCAG 1.0 Priority 1 and 2. It will not check your compliance with WCAG 2.0.

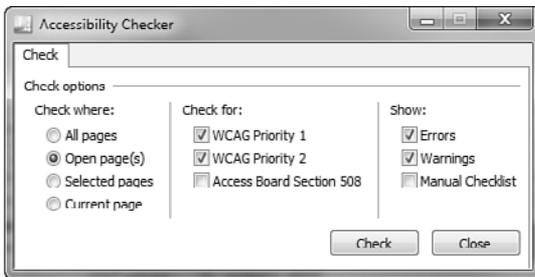


Figure 1.16

The Accessibility Checker ensures that your site meets the latest accessibility standards.

➔ For more information on accessibility features in Expression Web, see Chapter 12, “Maintaining Compatibility and Accessibility.”

Compatibility Checker

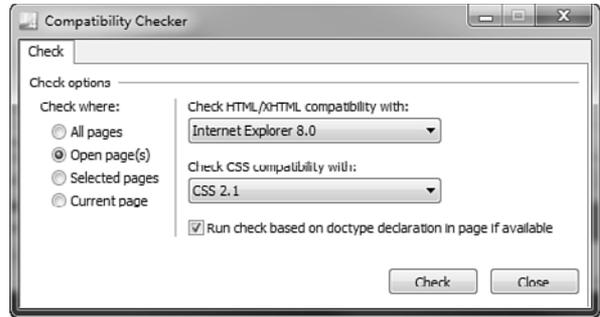
Ensuring that your site renders properly in all browsers is a difficult task. Expression Web offers the Compatibility Checker feature so you can identify rendering problems before they affect your site visitors (see Figure 1.17).

The Compatibility Checker allows you to check against a browser version, document type, and CSS standard.

➔ For more information on using Compatibility Checker, see Chapter 12, “Maintaining Compatibility and Accessibility.”

Figure 1.17

The Compatibility Checker can help you avoid any rendering problems.

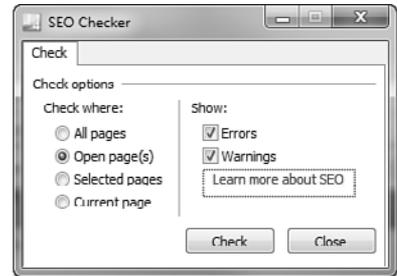


SEO Checker

The SEO Checker feature is new to Expression Web and makes it easy to optimize your site for search engines. The Learn More About SEO link (shown in Figure 1.18) is a great way to learn more about optimizing your site for maximum traffic.

Figure 1.18

SEO Checker is a great addition to Expression Web.



SuperPreview

Microsoft Expression SuperPreview (shown in Figure 1.19) solves browser compatibility problems by allowing designers to quickly see how a page will render in multiple browsers. Expression Web adds a remote browser service so that you can see what your site looks like on Safari running on a Mac.

SuperPreview also offers tools for identifying and correcting problems with position and other rendering problems.

➔ *For more information on using SuperPreview, see Chapter 13, "Using SuperPreview."*

CSS Reports

To aid in cleaning up unused CSS or to simply run a report containing details on CSS usage, Expression Web offers the CSS Reports feature (see Figure 1.20).

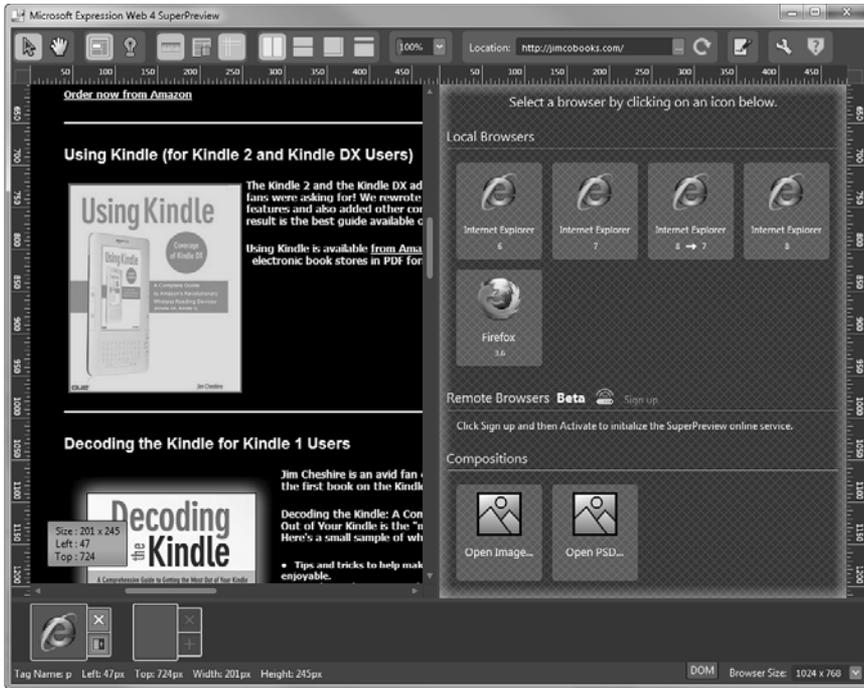


Figure 1.19 SuperPreview is a powerful way to preview a page in multiple browsers.

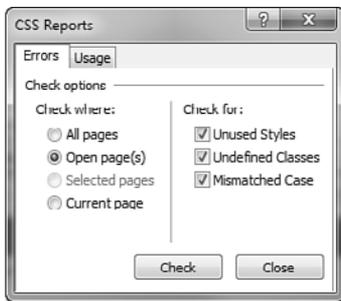


Figure 1.20 CSS Reports can help you clean up your CSS before it causes a problem.

➔ For more information on CSS Reports, see Chapter 18, "Managing CSS Styles."

Scripting and Dynamic Content

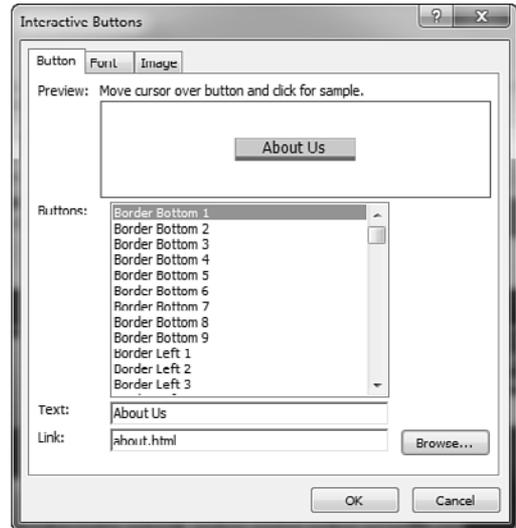
Expression Web offers many features that let you add dynamic content to a site without writing any code.

Interactive Buttons

Interactive buttons (see Figure 1.21) are buttons that use JavaScript for a rollover effect. Creating interactive buttons is easy, and you have numerous predesigned buttons from which to choose.

Figure 1.21

If you need a simple button with rollover animation, an interactive button might be the perfect choice.



➔ For more information on using interactive buttons, see Chapter 20, “Using Interactive Buttons.”

Behaviors

Behaviors are a collection of easily configurable actions that are implemented with a client-side script. Using behaviors, you can easily create dynamic effects that would normally require a considerable amount of code, and you can do it without knowing anything about writing client script.

Behaviors are accessed via the Behaviors panel, as shown in Figure 1.22.

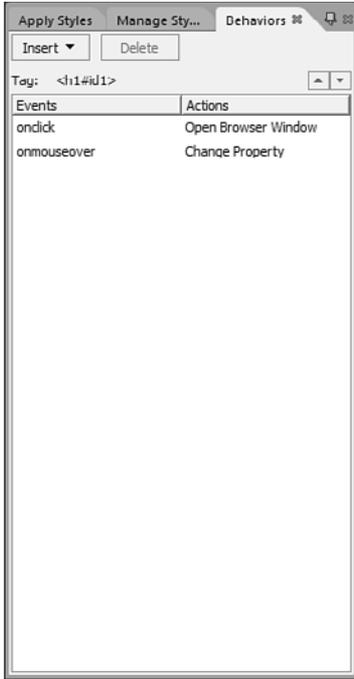
➔ For more information on using behaviors, see Chapter 21, “Using Behaviors.”

Layers

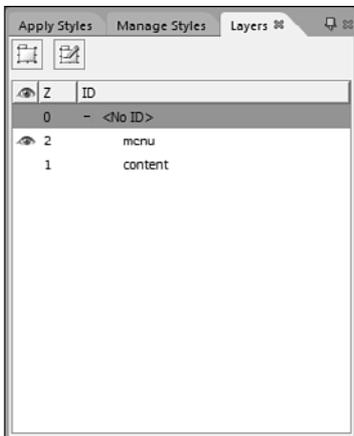
Layers can be used in scripted, dynamic pages. Expression Web provides tools that allow you to create layers and manipulate them easily.

Layers are configured using the Layers panel (see Figure 1.23).

➔ For more information on using layers, see Chapter 23, “Using Layers.”

**Figure 1.22**

Expression Web provides a large collection of behaviors to automatically add dynamic effects to your site.

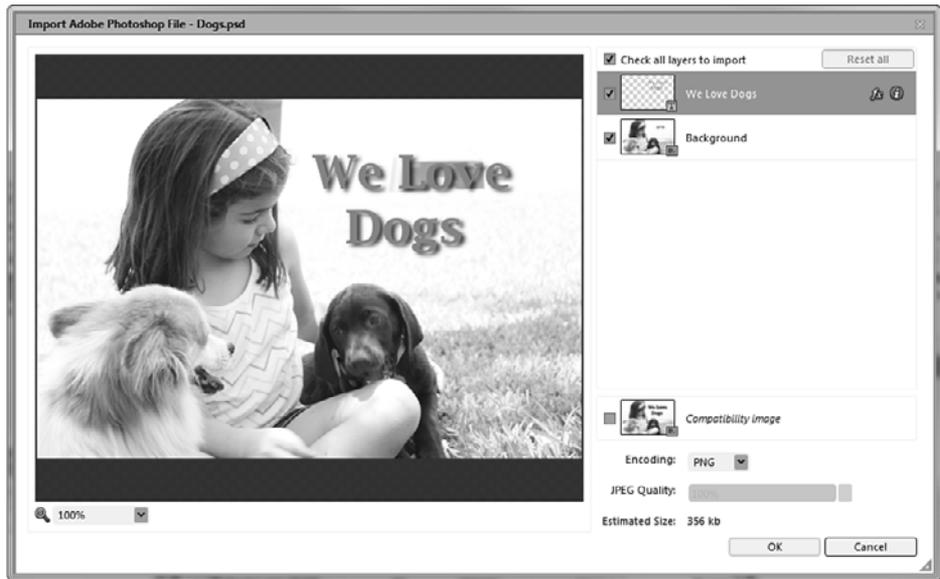
**Figure 1.23**

Layers are portions of a page you can position easily.

Photoshop Content

Expression Web enables you to conveniently include Adobe Photoshop content in your site. You can choose which layers to import as shown in Figure 1.24.

Figure 1.24
Adobe Photoshop users have some powerful tools in Expression Web.



➔ For more information on using Photoshop import features, see Chapter 9, “Using Graphics and Multimedia.”

Deep Zoom Images

Deep Zoom makes it possible to display and zoom into large images on a page without a slowdown in performance. Expression Web lets you insert a Deep Zoom image onto your page.

➔ For more information on inserting Deep Zoom images, see Chapter 9, “Using Graphics and Multimedia.”

Video

In addition to the capability of inserting Silverlight content into your web page, Expression Web also comes with Expression Media Encoder, which makes it easier than ever to add video to your site.

➔ For more information on adding video to your web page, see Chapter 9, “Using Graphics and Multimedia.”

ASP.NET Controls

ASP.NET is Microsoft's dynamic website technology. Expression Web not only supports the use of ASP.NET pages, but also provides quick access to ASP.NET controls using the Toolbox, as shown in Figure 1.25. You can also configure controls within the design surface in Expression Web.

Expression Web also provides support for third-party ASP.NET controls in the `/bin` folder of your site, as well as the ASP.NET Ajax extensions.

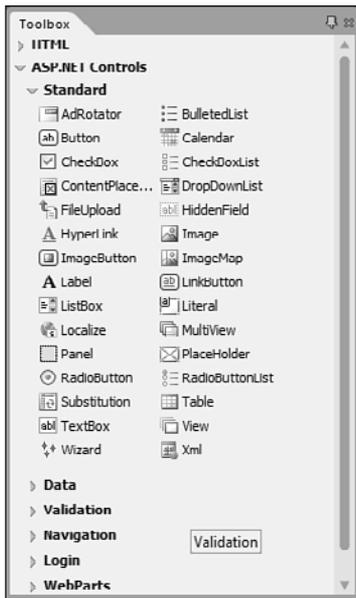


Figure 1.25

Support for ASP.NET controls is included in Expression Web. Controls can be added from the Toolbox.

You don't need to worry when testing your ASP.NET or PHP pages because Expression Web ships with the Microsoft Expression Development Server—a web server that can be used to test ASP.NET and PHP pages.

➔ *For more information on ASP.NET development in Expression Web, see Part VI, “ASP.NET and PHP Development.”*

note

Expression Web does not support adding third-party controls to the Toolbox.

PHP Support

If you're a PHP developer, Expression Web offers color-coding and IntelliSense for PHP code. You also can easily insert commonly used PHP code fragments with the Insert menu in Expression Web and preview PHP pages using the Microsoft Expression Development Server.

➔ *For more information on using PHP, see Chapter 32, “Using PHP.”*

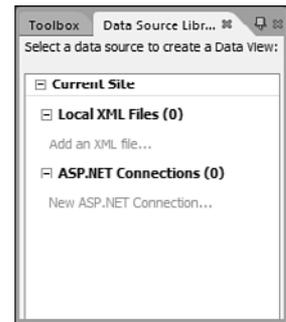
Data Access Features

If your site is data-driven, Expression Web brings plenty of great features to the table. Using the power of ASP.NET, Expression Web allows you to create dynamic, data-enabled sites without writing any code.

Data connections are configured in the Data Source Library panel so they can be manipulated easily within the Expression Web interface (see Figure 1.26).

Figure 1.26

Accessing data couldn't be easier. Data connections are accessed via the Data Source Library panel.



➔ *For more information on accessing data in Expression Web, see Chapter 34, “Displaying and Editing Database Data with ASP.NET.”*

Creating Add-Ins

Expression Web 4 introduces a new extensibility model that allows for the development of add-ins using JavaScript and HTML. You can create dialog add-ins, panel add-ins, or command line add-ins. Distributing add-ins is easy. You simply zip up the files and rename the compressed file to an .add file extension. You can then easily install your add-in by simply double-clicking the .add file.

➔ *For more information on creating add-ins in Expression Web, see the online Chapter 36, “Expression Web 4 Add-In Basics.”*

This page intentionally left blank

CREATING, OPENING, AND IMPORTING SITES

What Is a Site?

At first glance, that might seem like a patronizing question, but it's not. I'm not talking about a site in the generic sense. I'm talking about a site as it relates to Expression Web.

Expression Web is a capable page editor, but it's really designed to work with a collection of related files. That collection of related files is what Expression Web refers to as a *site*.

Expression Web 4 has many features designed with a site in mind. The reporting features of Expression Web, for example, are designed to provide information on an entire site. There are also numerous dialogs that allow you to take a specific action on all files in a site, and the Find and Replace feature in Expression Web is most robust when used with a site and not just a single page.

A site can consist of pages, Cascading Style Sheets (CSS) files, script files, image files, and so on. It also can contain other folders for the purpose of organizing the files within the site.

How Expression Web Maintains a Site

By default, Expression Web tracks files and folders in your site, as well as other important information about the site, using hidden files called *meta-data*. These files are in hidden folders that begin with `_vti` and are located in the same folder as the rest of your site, as shown in Figure 2.1.

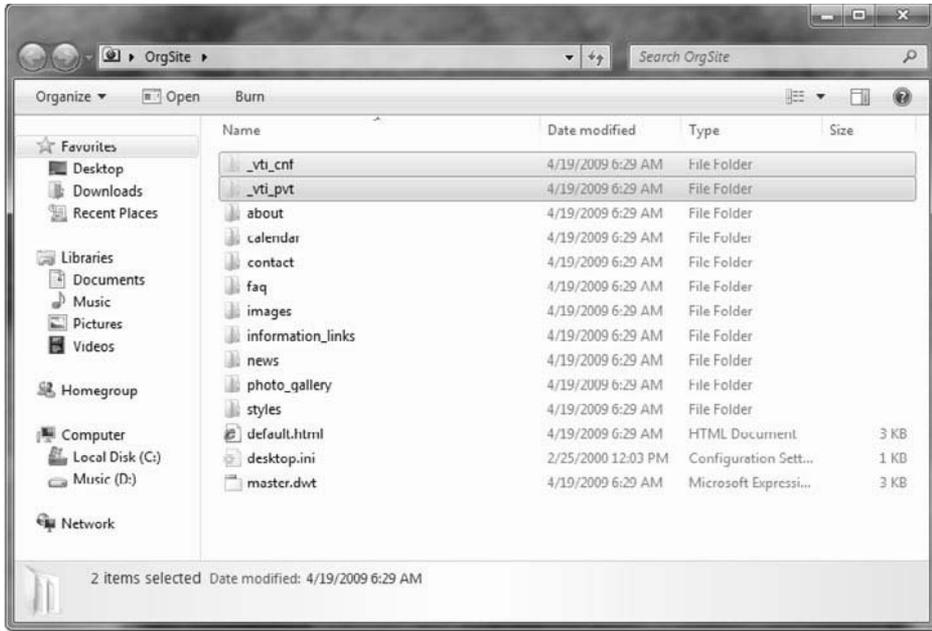


Figure 2.1
The `_vti` folders shown here contain special configuration information that Expression Web uses to keep track of your site.



Don't See `_vti` Folders

If you have Expression Web configured to use metadata files to keep track of your content, but you don't see any `_vti` folders when you open the disk path where the site is located, the `_vti` folders are hidden. The default settings in Windows won't show you hidden folders. To show hidden files and folders, follow these steps for Windows 7, Windows Vista, or Windows Server 2008:

1. Click Start and click Computer.
2. Click the Organize button and select Folder and Search Options.
3. Click the View tab.
4. Select the Show Hidden Files and Folders radio button (Show Hidden Folder, Files, and Drives on Windows 7) and click OK.

In Windows XP or Windows 2003 Server:

1. Click Start and click My Computer.
2. Select Tools, Folder Options.
3. Click the View tab.
4. Select the Show Hidden Files and Folders radio button and click OK.

You should now be able to see the `_vti` folders.

If you choose, you can configure Expression Web so it doesn't use metadata to track your site's files. However, it's best to just allow Expression Web to keep metadata for your site. Doing so allows it to efficiently track changes to your site. However, if you choose not to use them, the functionality of the live site will not be affected.

➔ *“For more information on configuring your site settings, see Chapter 15, “Site Management and Reporting.”*

note

The `_vti` folders are not visible in Expression Web's folder list. Expression Web explicitly hides those folders from view so you don't unintentionally modify them.

Sites and Subsites

Expression Web enables you to create a new site within an existing site (a site known as a subsite), but for Expression Web to recognize the subsite, it needs to add metadata information to the site. If the option to store site information using hidden metadata is disabled, Expression Web will no longer be able to recognize the subsite as a separate site.

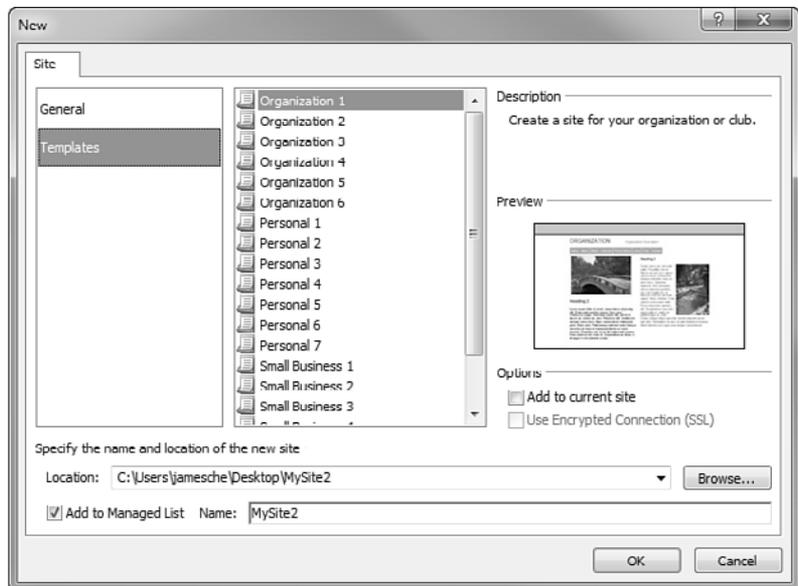
Site Templates

The quickest way to get a site up and running is to use the templates provided in Expression Web. You access the list of templates by selecting Site, New Site, and selecting Templates in the New dialog as shown in Figure 2.2. These aren't cheap-looking, stock templates. They are professionally designed templates that are customizable.

note

An empty site is a convenient way to create a site from a folder of existing files. When you select the Empty Site option, Expression Web will not add any files or folders to the location you specify.

Figure 2.2
Expression Web comes with professionally designed, customizable templates so you can get a fully functional site up and running quickly.



The site in Figure 2.3 was created using one of Expression Web's many site templates. The site uses a Dynamic Web Template so that changes can be made quickly if necessary.

➔ *For more information on Dynamic Web Templates, see Chapter 19, "Using Dynamic Web Templates."*

The templates in Expression Web make heavy use of external style sheets, as shown in Figure 2.4. Because the look and feel of a site is implemented using CSS, it's simple to use the robust CSS features in Expression Web to alter the look and feel of all pages in a site without much effort.

note

You don't have to use a template to create a site. You can use the One Page Site option in the New dialog to create a site with one blank page, or you can create an empty site.

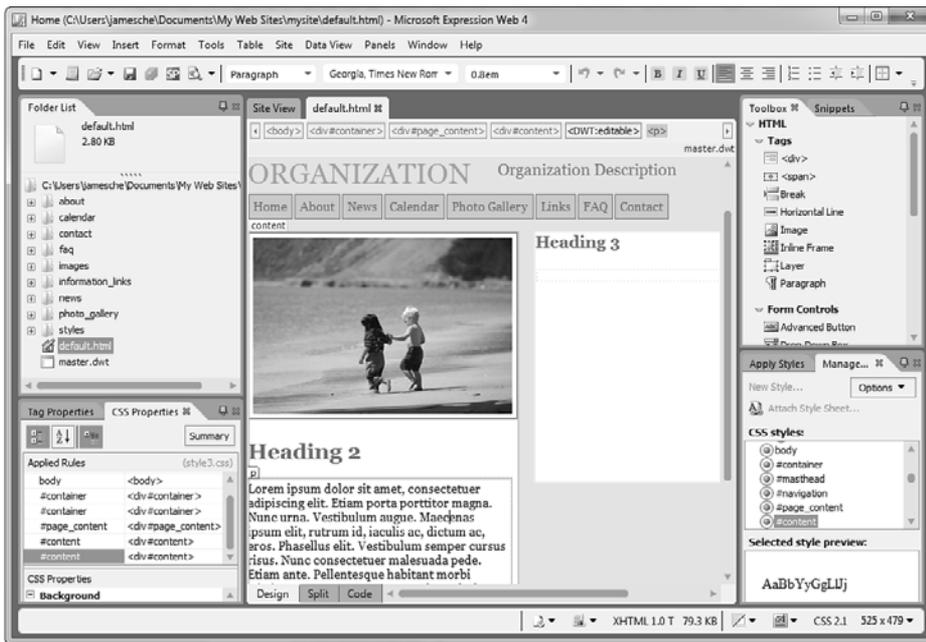
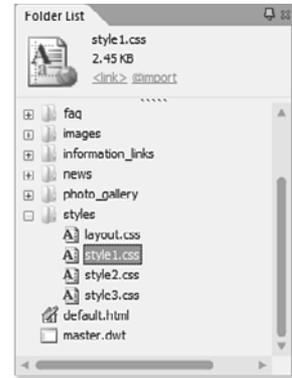


Figure 2.3 Dynamic Web Templates allow you to make changes to a template site just by updating the Dynamic Web Template.

➔ *For more information on working with CSS in Expression Web, see Chapter 18, "Managing CSS Styles."*

Figure 2.4

The look and feel of this site is implemented using several CSS files shown here, making it easy to modify.



Types of Sites

When starting work on a site, you'll need to decide which type of site to create. The three types of sites to choose from are

- Disk-based site
- FTP (File Transfer Protocol) site (or secure FTP using SFTP [FTPS])
- HTTP (Hypertext Transfer Protocol) site (or secure HTTP [HTTPS] with Secure Sockets Layer [SSL])

Disk-Based Sites

Disk-based sites are created at a specific disk location. The advantage of a disk-based site is that it can be created anywhere on your local file system or on a remote drive. You can also create disk-based sites on thumb drives, external hard drives, and so on.

Figure 2.4 shows the folder list of a disk-based site open in Expression Web. Note that the path to the site is a disk location on my C: drive.

A disk-based site is the most flexible kind of site because it allows the site to be located in any folder you want. For example, if you receive files for a site from someone else, you can work with them as a site in Expression Web easily by simply copying the files into a folder on your computer and opening that folder as a site.

To create a disk-based site:

1. Select Site, New Site.
2. Enter the path for the new site, as shown in Figure 2.5, or click the Browse button to navigate to the desired location.



tip

If you tell Expression Web to create a site in a folder that doesn't exist, the folder will be created for you automatically. There is no prompt when this happens.

3. Select a general site or a site template from the dialog.
4. Check the Add to Managed List check box if you want the new site to appear in your Managed Sites List.
5. Click OK to create the site.

You can create a disk-based site on a mapped drive or on a universal naming convention (UNC) share (\\server\share) as well. However, you'll need to have the correct permissions so you can write to the remote location. Check with your system administrator if you're not sure.

If a site is already open in Expression Web and you want to add a new site to the existing site check the Add to Current Site check box and Expression Web adds the new site to the existing site. This is a convenient way to add pages from one of the templates that come with Expression Web to an existing site.



caution

If you're using Dynamic Web Templates in your site, you need to allow Expression Web to store metadata files in your site. You also should leave metadata enabled if you want Expression Web to correct links as pages are moved or renamed.

Don't ignore the warning about removing metadata! Some of the information in the site's metadata will be lost permanently if you choose to remove the metadata.

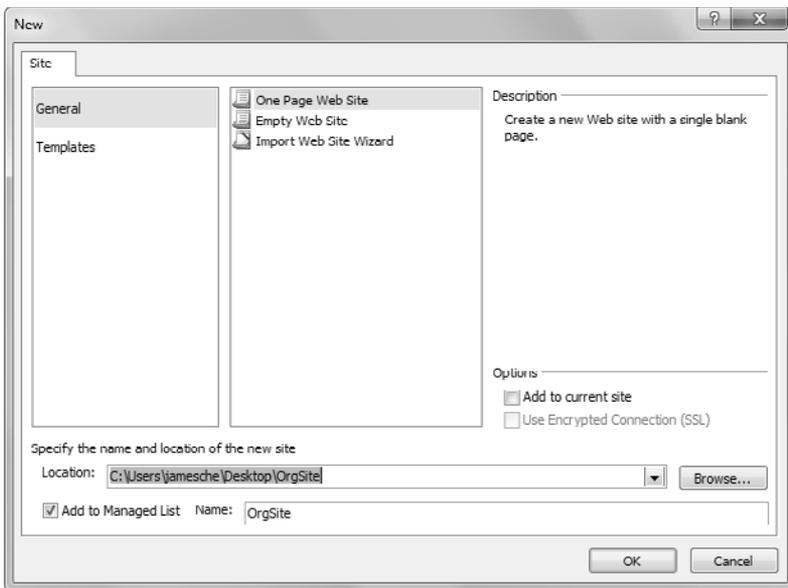


Figure 2.5

To create a disk-based site, enter a disk path in the New dialog.

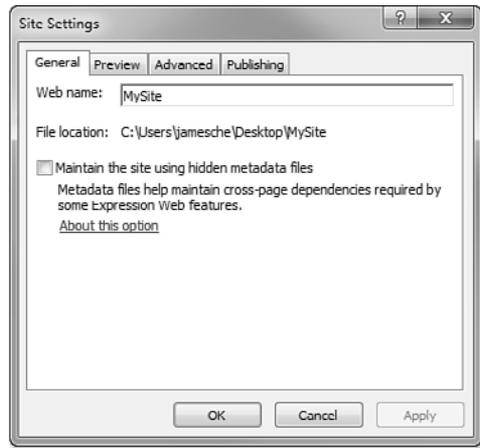
If content already exists at the location you specify, Expression Web adds metadata to the existing folders. If you're not using Dynamic Web Templates, or if you don't want Expression Web to maintain links in your site, you might want to stop Expression Web from saving additional files in your site. If you don't want Expression Web metadata in the folder, select Site, Site Settings and uncheck the Maintain the Site Using Hidden Metadata Files check box in the Site Settings dialog, as shown in Figure 2.6. When you uncheck that box and click OK, you are asked whether you're sure you want to remove the metadata, as shown in Figure 2.7. If you click Yes, all the `_vti` folders and metadata therein are removed.

**tip**

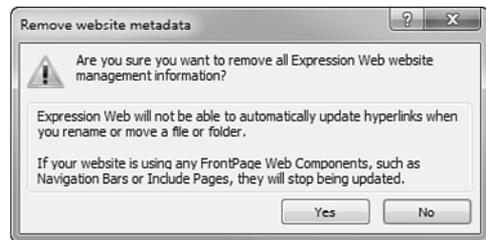
Because a site can be created anywhere on the file system, it's easy to lose track of where a particular site was created. The Recent Sites menu option on the File menu is a convenient way to locate a recently created site. You can also use the Managed Sites List to easily keep track of your sites.

Figure 2.6

To get rid of the `_vti` folders and the metadata they contain, uncheck the option to store site information in metadata files in the Site Settings dialog.

**Figure 2.7**

When you choose to remove metadata files, Expression Web warns you that you'll lose some functionality.



Expression Web can use the Microsoft Expression Development Server to test disk-based sites, allowing you to take full advantage of ASP.NET and PHP: Hypertext Preprocessor (PHP) features that require a web server.

➔ *For more information on the Microsoft Expression Development Server, see Chapter 33, “Using the Microsoft Expression Development Server.”*

FTP Sites

FTP sites are usually located on a remote web server. Most web hosts provide FTP access so you can use FTP with your site. Most applications require you to develop a site locally and then FTP the files to the remote web server. Expression Web takes a different approach in that it offers you the ability to work on an FTP site live on the web server. Because FTP is offered by almost all hosts, it is a convenient way to work with sites using Expression Web. If you choose to use FTP with Expression Web, use SSH File Transfer Protocol (SFTP) or FTP over SSL (FTPS) if possible so that the information (such as usernames and passwords) is encrypted.

To open an FTP site, select Site, Open Site and enter the FTP path. When you click Open, Expression Web asks whether you want to open the site live or copy it to your local machine, as shown in Figure 2.8.



caution

You can delete a site by right-clicking the site name in the Folder List and selecting Delete. However, be careful about deleting sites you've created on your file system. If you create a site in a location with existing files and folders and then delete that site, Expression Web deletes everything in that folder.

Files and folders deleted by Expression Web are not moved to the Recycle Bin. They're gone forever!



caution

If you choose to work on your site live, keep in mind that any changes you make are immediately visible on the live site after you save the page.



tip

To use SFTP, use the format `sftp://site.com`. To use FTPS, use the format `ftps://site.com`. Your host or server administrator will need to configure a secure FTP connection to use one of these methods.



Figure 2.8

Expression Web offers the option of opening an FTP site live on the web server.



caution

If you're a notebook user with wireless capability, you may use open Wi-Fi hotspots. Be aware that open Wi-Fi hotspots are unsecure, and anything you transmit and receive over the wireless network can be intercepted by anyone else using that wireless network.

When you are using an open Wi-Fi hotspot, use a virtual private network (VPN) solution such as Hamachi2 (<https://secure.logmein.com/products/hamachi/>) or GoToMyPC (www.gotomypc.com) so you can securely access the network.

➔ *For a more in-depth discussion of FTP, see Chapter 14, “Publishing a Site.”*

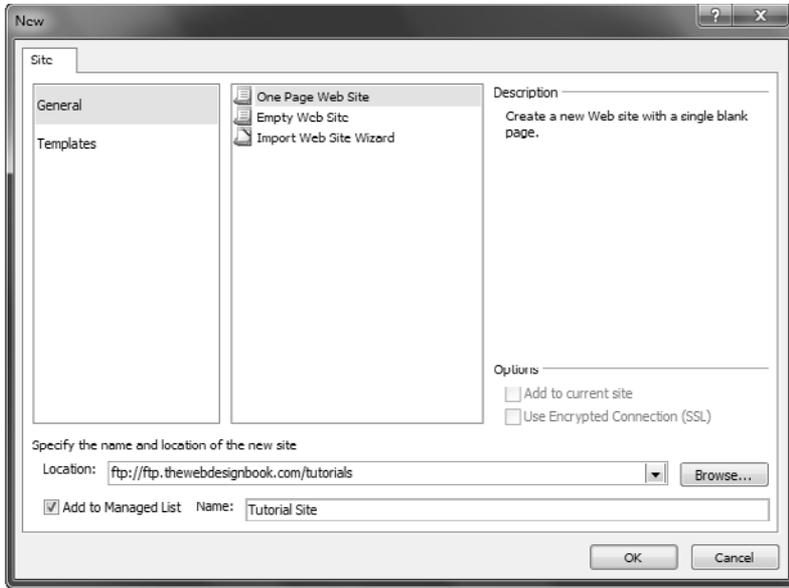
After you choose whether you want to edit the site live or copy it to your local machine, Expression Web prompts you for your username and password. If you want Expression Web to remember your password for the site, check the Remember My Password box, as shown in Figure 2.9.

Figure 2.9

Expression Web remembers your password for a site if you check the Remember My Password check box.



To create an FTP site, enter the FTP path to your web server in the New dialog as shown in Figure 2.10. You will likely be prompted for a username and password as shown previously in Figure 2.9.

**Figure 2.10**

When creating a site using FTP, enter the FTP location. If you don't know the correct location, your web server administrator can help.



Cannot Locate Server Error Creating FTP Site

If Expression Web cannot locate an FTP site you've created, it's likely that you mistyped it. However, there are other reasons this might fail.

Try opening a command prompt and pinging the FTP site. For example, the following command tests for the existence of an FTP server at jimcosoftware.com:

```
ping ftp.jimcosoftware.com
```

If you get no reply from the ping, it means you aren't getting to the server. Check your Internet connection by browsing to a known site. If that works and you still can't access the FTP site, check with the server administrator or your hosting company.



Repeatedly Prompted for Username and Password

Suppose you're trying to create a site on a remote server. When you enter your username and password, the dialog asking for your credentials just keeps popping back up.

This is almost certainly caused by a typographical error in the username and/or password. Assuming you aren't mistakenly entering the wrong case because your Caps Lock key is engaged, carefully retype your username and password. If the server still won't accept them, try contacting your hosting company or server administrator.

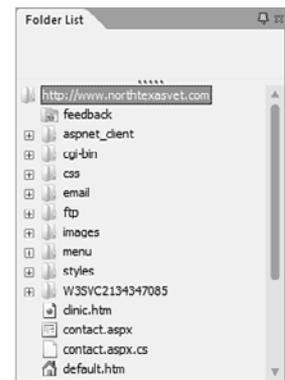
HTTP Sites

HTTP sites employ the same technology used when viewing your site in a web browser. This method offers a greater selection of authentication methods for the web server administrator. In a private network environment, Windows-integrated authentication can be used so that user credentials are securely exchanged. You can also use secure HTTP (HTTPS) so credentials are encrypted.

Figure 2.11 shows the Folder List of an HTTP site open in Expression Web. Notice that the address of the site in Expression Web is the same address that would be used to browse this site in a web browser.

Figure 2.11

If your host has the FrontPage Server Extensions installed, you can open your site live using HTTP (or HTTPS if your server supports SSL) as shown here.



There's one caveat to creating a site using HTTP. The web server on which you are creating the site must be running the FrontPage Server Extensions. Many hosting companies no longer offer support for FrontPage Server Extensions, so if having the capability of using HTTP to open your site is important to you, make sure that the hosting company you choose supports the FrontPage Server Extensions.

It's also important to understand that even though Expression Web allows for authoring with the FrontPage Server Extensions, Microsoft no longer officially supports it. If you have problems when using the FrontPage Server Extensions for authoring and you open a support case with Microsoft, it might require you to reproduce your problem with a disk-based site.

➔ *For more information on the FrontPage Server Extensions, see Chapter 14, "Publishing a Site."*

Creating an HTTP Site

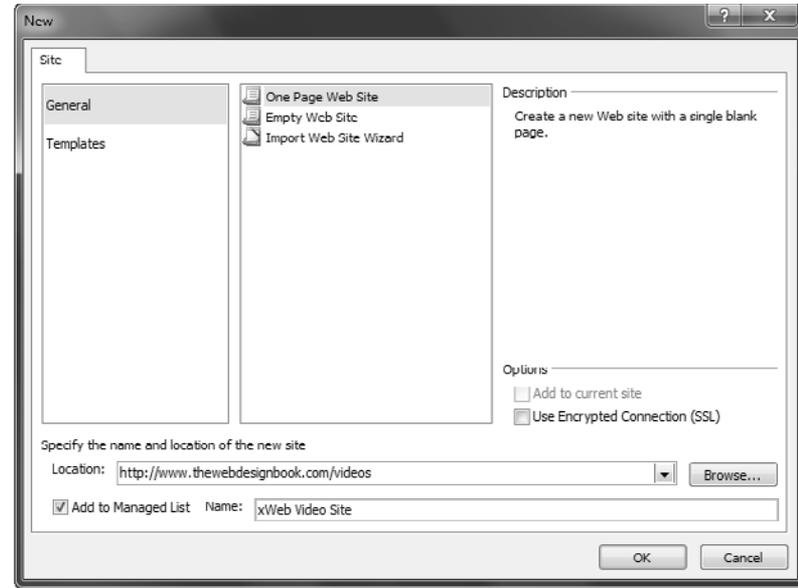
To create an HTTP site, enter the URL for the site in the New dialog as shown in Figure 2.12. If necessary, enter your username and password.

note

Expression Web can also use WebDAV to create HTTP sites, but I've yet to encounter a host that supports the use of WebDAV. If you are using Expression Web in a corporate environment, you may be able to use WebDAV instead of the FrontPage Server Extensions to create HTTP sites.

tip

Many Internet service providers provide custom pages when you attempt to access a site that doesn't exist. If your Internet service provider includes this feature, you might see the dialog shown in Figure 2.15 if you try to create a site on a server that doesn't exist.

**Figure 2.12**

When creating a site using HTTP, use the same uniform resource locator (URL) that you would use to browse the site in your web browser.

➔ *For more information on the FrontPage Server Extensions and WebDAV, see Chapter 14, “Publishing a Site.”*

You cannot create a new site at the root of a web server (for example, at `http://www.mysite.com`). If you try to, Expression Web won't even check to see whether a web server exists at that location. Instead, it will notify you that you need to create a sub-site instead, as shown in Figure 2.13.

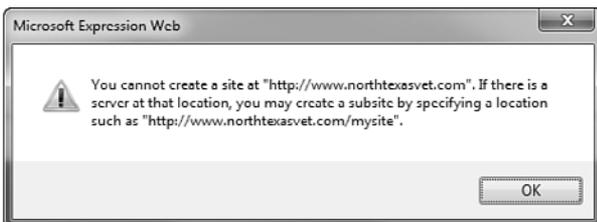
**tip**

If you enter a URL that starts with “https://”, Expression Web automatically checks the Use Encrypted Connection (SSL) check box.

**note**

SSL is a technology developed by Netscape many years ago to allow for the secure exchange of information across the Internet.

To use SSL, your web server administrator needs to configure the web server appropriately.

**Figure 2.13**

If you try to create a new site at the root of a web server, Expression Web immediately notifies you that you must create a subsite instead.

If you enter an address for a new subsite and Expression Web cannot locate a web server at the URL you specify, you see the dialog shown in Figure 2.14. This dialog means that Expression Web attempted to contact the domain you specified (oiuoiu.com in the case of Figure 2.14) and did not find any such domain.

Finally, if you attempt to create a new subsite on an existing web server and either that server doesn't have the FrontPage Server Extensions installed or there is a configuration problem with the FrontPage Server Extensions, you see the dialog shown in Figure 2.15. You also see this dialog in cases where there is a web server at the URL you specify but the web server is not available.



caution

Always remember that sites are copyrighted material. Current laws as of this writing automatically provide for a common-law copyright on any published site.

You can use the Import Site wizard to import any site you want, but it is illegal to use the content or the design of someone else's site without their express written permission.

Figure 2.14
If there is no web server at the location you specify when creating a subsite, Expression Web notifies you.



tip

Keep in mind that when you import a site, you aren't necessarily moving the site from a remote location to your local machine. The act of importing a site is really just the process of copying the site from one location to another.

Figure 2.15
If the FrontPage Server Extensions are not installed, or if there is a problem with the FrontPage Server Extensions on the server you specify, you are notified of the problem.



Creating a Site Using SSL

If you require additional security and your server is configured to allow SSL traffic, you can create a site using SSL by checking the Use Encrypted Connection (SSL) check box, as shown in Figure 2.16.

When you check the Use Encrypted Connection (SSL) check box, Expression Web automatically changes the URL you have specified, if necessary, so it uses HTTPS instead of HTTP. When this option is selected, any data sent over the network (including your username and password) is encrypted.

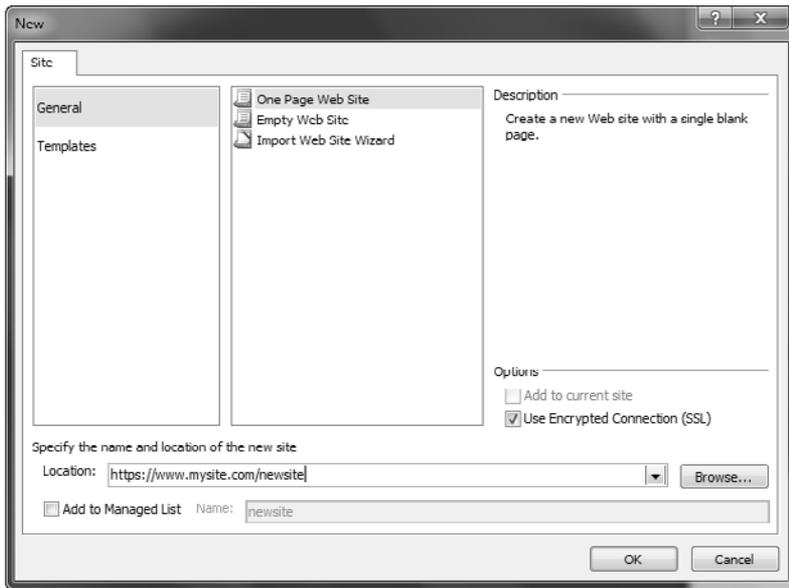


Figure 2.16

You can create a site using SSL by checking the Use Encrypted Connection (SSL) check box. Your server must be configured to accept SSL traffic.

Importing Sites with the Import Site Wizard

Expression Web's Import Site wizard, which consists of three primary steps, is a convenient and powerful way to import site content. Using the Import Site wizard, you can easily control how a site is imported. You can choose whether to use the FrontPage Server Extensions, FTP, and so on, and in some cases, you can also specify how Expression Web decides whether to import certain files.

The Import Site wizard is accessed by selecting Site, Import, Import Site Wizard. Let's have a look at the steps involved in importing a site.

Choosing an Import Method

First, you need to choose how the site's files are imported. You can choose among the following methods of importing a site:

- FTP
- SFTP
- FTPS
- FrontPage Server Extensions
- WebDAV
- File system
- HTTP

All these methods except the FrontPage Server Extensions method use the Publish feature of Expression Web to import the remote site. After you've specified the necessary information for Expression Web to open the remote site, the remote site is published to the destination you've selected.

➔ *For more information on using the Publish feature in Expression Web, see Chapter 14, "Publishing a Site."*

FTP, SFTP, and FTPS

To use the FTP to import a site, choose the FTP option. You need a username and password to import a site using FTP.

To import a site using FTP, enter the FTP address of the remote site in the Web Site Location text box, as shown in Figure 2.17. Depending on the configuration of the remote site, you might also need to enter a directory in the Directory text box, and enable passive FTP by checking the Use Passive FTP check box. You can also increase the number of simultaneous connections from the default of 4 up to a maximum of 10. Doing so improves the speed at which the site is imported. However, there is a limit to the total number of connections your computer can have open at any one time, so increasing the connections in Expression Web can impact other applications.

If you want to import the site using secure FTP, choose either the SFTP or FTPS option when importing.

➔ *For more information on the options available with FTP, see Chapter 14, "Publishing a Site."*

After you've entered the necessary information, click the Next button to continue. You are prompted for a username and password before you can proceed to the next step.



caution

If the remote web server has the FrontPage Server Extensions installed, make sure you use this option when importing the site. If you don't, the files that provide the FrontPage Server Extensions with configuration information won't be imported.



tip

Most hosting companies do not support WebDAV publishing at this time.



note

The file system method is often used in an office network environment where many file servers are available.

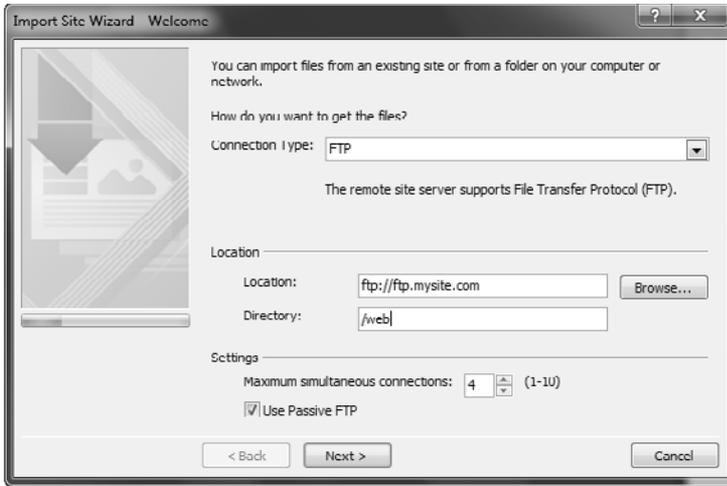


Figure 2.17
Importing using FTP might require a directory and passive FTP. Ask the administrator of the remote site if you're not sure.

FrontPage Server Extensions

If the remote site is running FrontPage Server Extensions, the FrontPage Server Extensions option should be used instead of the HTTP option. If you use the HTTP option, Expression Web will import files by following links in the site just as though you were viewing the site in a browser. However, if you use the FrontPage Server Extensions option, Expression Web opens the remote site and publishes the remote files to your local machine. This ensures that you get all of the files from the remote site.

After selecting this method, enter the URL of the remote site in the Location text box shown in Figure 2.18. Use the same URL that you would use when viewing the site in a web browser.



Figure 2.18
The FrontPage Server Extensions method of importing a site allows you to import subsites as well.

You have the option of including subsites by checking the Include Subsites check box. You can also import a site using the Secure Sockets Layer (SSL).

➔ *For more information on subsites, see “Sites and Subsites,” p. 27, earlier in this chapter.*

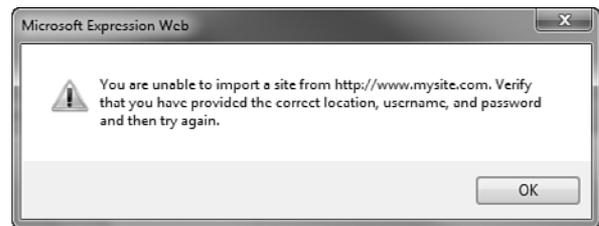
To proceed to the next step, click the Next button. If the FrontPage Server Extensions are not installed on the remote site, you are notified that you cannot import the site, as shown in Figure 2.19.

Assuming the FrontPage Server Extensions are located on the site, you are prompted for a username and password to import the site. The username and password you need to enter are assigned by the administrator of the site or by the hosting company.

**tip**

When you see the dialog shown in Figure 2.19, you should first check to see whether the FrontPage Server Extensions are installed on the remote site. If you find that they are installed and you still get this dialog, check with your host to find out what might be wrong.

Figure 2.19
If the FrontPage Server Extensions aren't installed, Expression Web tells you that the selected site cannot be imported.



WebDAV

If the remote web server supports Web Distributed Authoring and Versioning (WebDAV), you can use this option to import a site. WebDAV is similar to using the FrontPage Server Extensions option, but the FrontPage Server Extensions are not required for WebDAV.

When using WebDAV, you don't have the option of including subsites when you import a site. However, you can import using SSL when using the WebDAV method by checking the Use Encrypted Connection (SSL) check box. If you want to import a subsite's content, you need to specify the path to the subsite instead of the path to the parent site.

Just as with the FrontPage Server Extensions method, you need to enter a username and password when importing a site using WebDAV.

➔ *For more information on WebDAV, see Chapter 14, “Publishing a Site.”*

File System

The file system method is often used when importing a site from a local file system. However, it can also be used to import a site from any location to which you have access via the file system. In

other words, in addition to importing from a local drive, you can also use either a universal naming convention (UNC) path or a mapped drive to import using this method.

A UNC path consists of a server name and a share name and is in the format `\\server\share`. For example, to access a share called `sites` on a server named `server1`, you would use the UNC path `\\server1\sites`.

Because Expression Web provides support for disk-based sites, you can also choose to include subsites when using the file system method by checking the Include Subsites check box.

HTTP

The HTTP method is used to import a site to which you don't have access via any of the previously discussed methods. This is the only method that does not require authorization to complete the import process.

The HTTP import process is different from the other import methods. In fact, you can import any site you can get to on the Internet using this method. Instead of actually publishing the site during the import process, the HTTP import method can be configured to follow hyperlinks from the home page.

When you choose the HTTP method to import a site, the Import Site wizard adds the Set Import Limits step, as shown in Figure 2.20.



tip

Windows automatically sets up a share for the root of every drive on your computer. You can access any share by using the format `\\server\<drive>$`. For example, to access the root of the C drive on a computer named `desktop`, you would use the UNC path `\\desktop\c$`. You need to use an Administrator account to access these shares.



note

The Set Import Limits step actually appears as the third step in the wizard. I am covering it here only because it's unique to the HTTP method of importing.



tip

By default, Expression Web imports all files that are reachable from the home page and all child pages.

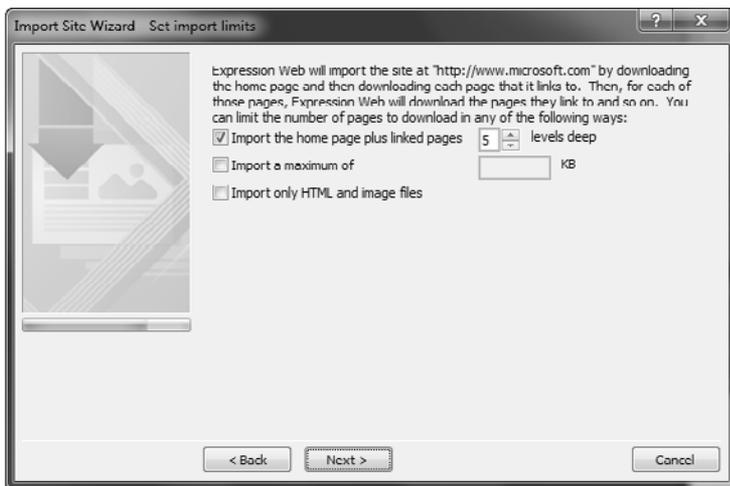


Figure 2.20
The HTTP method of importing a site adds another step to the Import Site wizard.

In this step, you can specify exactly how the Import Site wizard decides what to import using the following check boxes:

- **Import the Home Page Plus Linked Pages # Levels Deep**—When a site is imported, Expression Web follows hyperlinks to the depth that you specify. For example, if you specify 2 for the number of levels, Expression Web imports the home page, all pages linked to directly from the home page, and all pages linked to directly from those pages.
- **Import a Maximum of # KB**—Use this option when you want to put a cap on the amount of content that gets imported. This is particularly useful if you have metered bandwidth or a slower Internet connection.

When this option is selected, Expression Web imports the home page of the site and then follows all links from that page. It continues to follow links and import files until the maximum limit is reached.

- **Import Only HTML and Image Files**—When this box is checked, Expression Web imports only images and HTML files. Other files such as script files, CSS files, Flash files, and so on are not imported.



Import Web Site Wizard Skips Set Import Limits Step

If you're choosing HTTP as your import method, but you don't have the option of choosing the import limits when you run through the wizard, it's likely that the site you are trying to import is configured with the FrontPage Server Extensions. If Expression Web detects that the FrontPage Server Extensions are installed on the remote site, it automatically uses the FrontPage Server Extensions option when importing the site.

You'll have the option of choosing what to import when you are presented with the Site view at the end of the process.



Username and Password Not Accepted

If you're trying to import a site and get rejected even though you've entered the correct information when asked for your username and password, the most likely cause of this is an incorrect password. Take care to type the characters correctly and make sure that your Caps Lock isn't on. If you're positive that you've entered the information correctly, check with the hosting company or administrator of the remote web server.

Specifying a Destination Web Location

The second step of the Import Site wizard (shown in Figure 2.21) is where you specify the destination location for the imported site. This can be a UNC path, a disk location, or an HTTP location. However, when specifying an HTTP location, the site must have the FrontPage Server Extensions installed.

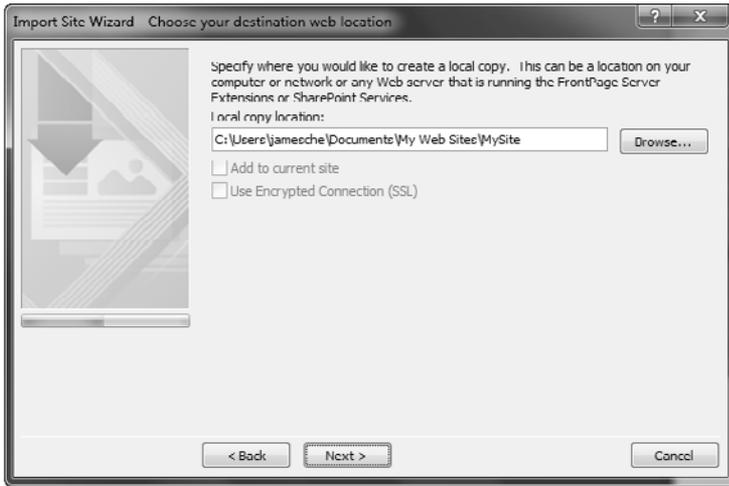


Figure 2.21
You can specify a destination for an imported site in the Import Site wizard.

If you already have a site open in Expression Web, check the Add to Current Web Site check box if you want the imported site to be added to the open site. If you require additional security, you can specify that SSL be used to import the site if you use an HTTP path for the destination.

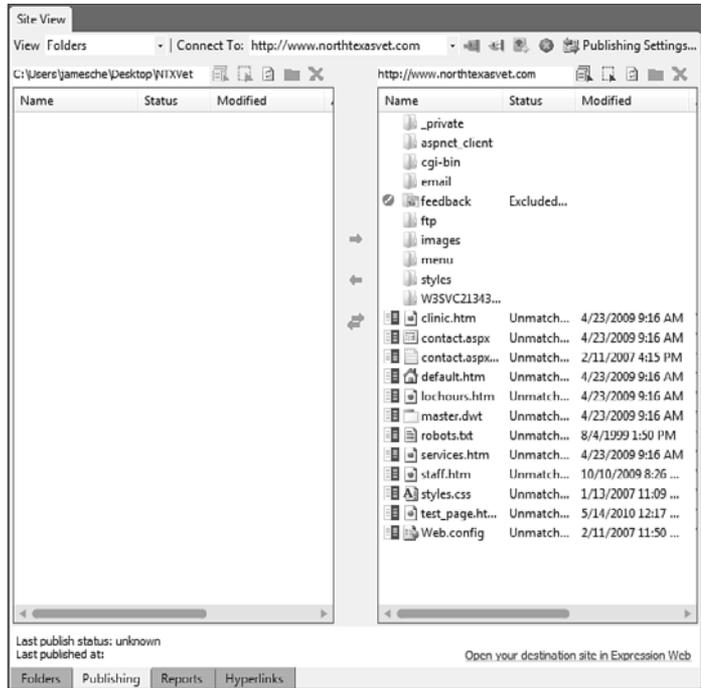
Finishing the Site Import Process

After you've specified all the necessary information, click the Finish button in the last step of the wizard to finish the import process. If you've chosen HTTP as your method of importing, the site begins importing immediately. Otherwise, you are presented with the Site view, as shown in Figure 2.22. You can then complete the import process by publishing your site using the Site view to copy the files and folders.

➔ *For more information on Site view to publish a site, see Chapter 14, "Publishing a Site."*

Figure 2.22

In all import methods except the HTTP method, you use the Site view to complete the import process.



Deciding on a Site Type

As you've seen in this chapter, there are a few choices when it comes to deciding on a site type in Expression Web. Deciding which type is best for your site is often confusing.

Until recently, web designers using the Microsoft platform were limited in their choice of site types when it came to remote sites. If you wanted full functionality, you were restricted to using the FrontPage Server Extensions. Sure, you could use FTP, but then you couldn't work with a remote site live.

Expression Web (and the Visual Studio family of products) has changed all of that, and there's a general tendency by Microsoft these days to shift away from a reliance on the FrontPage Server Extensions. You can now use FTP just as easily as you used HTTP in earlier products.

In my opinion, FTP (preferably FTPS or SFTP) is the best solution for today's sites. FTP is well-suited to fast file transfers, you have the ability to use many tools other than Expression Web to manage your files (including the FTP command-line client that comes with Windows), and unlike the FrontPage Server Extensions option used when creating HTTP sites, FTP will be around for many years to come.

Disk-based sites are also a perfectly valid option, especially if you are creating your site locally and then publishing it to the live server at a later time. One of the greatest advantages of a disk-based

site is that it can easily be moved from one location to another (or stored on a removable drive) without breaking anything. Even if you are using a server technology such as ASP.NET or PHP, a disk-based site is still a viable choice because you can test your server-side code using the Microsoft Expression Development Server.

If given the choice between using a technology that's being phased out and one that has proven to be a long-lived technology, I'll choose the one with greater longevity every time.

CREATING PAGES AND BASIC PAGE EDITING

Creating Pages

Expression Web gives you plenty of choices when creating new pages. Not only can you choose between standard HTML pages, ASP.NET pages and files, PHP files, and ASP files, but you can also create new pages that have an existing CSS layout.

To create a new page in Expression Web, select File, New, Page to display the New dialog shown in Figure 3.1.

The left side of the New dialog contains a list of page types. When a page type is selected, the list at the right is populated with numerous pages that can be created.

Many of the page types listed aren't really pages in the traditional sense. For example, if you select ASP.NET as the page type, you see options such as Web Configuration and Site Map, both of which are Extensible Markup Language (XML) files instead of actual pages.

Creating a page is literally as easy as selecting the type of page in the New dialog and clicking OK. After you create the new page, you need to add content to it before it will serve any useful purpose. Let's review the types of page formats you can choose when using the New dialog to create new pages.

General Page

Table 3.1 shows the page formats available when you select General in the New dialog and where you can find more information on certain formats.

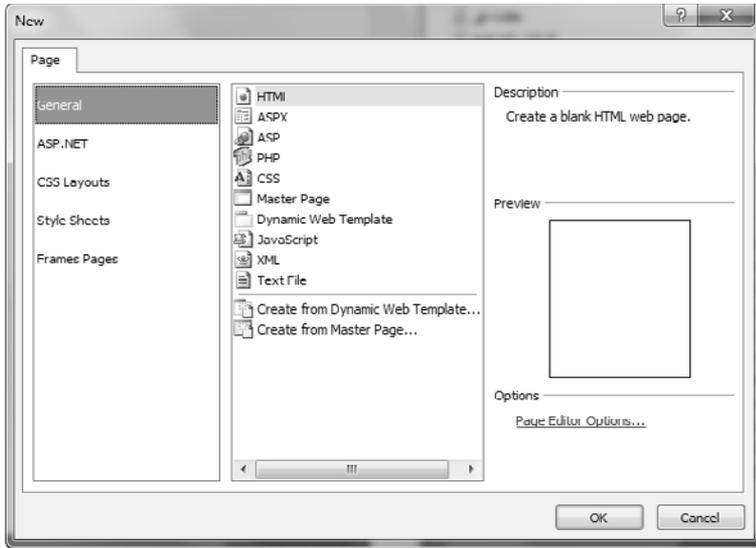


Figure 3.1
The New dialog makes it easy to create any type of page.

Table 3.1 General Page Formats

Type	Description	More Information
HTML	Creates a standard, blank HTML page.	
ASPX	Creates an ASP.NET Web form. When you choose this option, you also have the option of choosing a programming language for the page.	For more information on ASP.NET Web forms, see Chapter 25, “Using Standard ASP.NET Controls.”
ASP	Creates a legacy ASP page.	For more information on legacy ASP, read <i>Sams Teach Yourself Active Server Pages 3.0 in 21 Days</i> from Sams Publishing.
PHP	Creates a PHP file for use with PHP scripts.	For more information on using PHP, see Chapter 32, “Using PHP.”
CSS	Creates an empty CSS file.	For more information on CSS files, see Chapter 17, “Creating Style Sheets.”
Master Page	Creates a new ASP.NET Master Page that can be used as a template for other ASP.NET pages.	For more information on ASP.NET Master Pages, see Chapter 27, “Using ASP.NET Master Pages and User Controls.”

Type	Description	More Information
Dynamic Web Template	Creates a new Dynamic Web Template. Other pages can then be attached to the Dynamic Web Template to create a consistent look and feel for your site.	For more information on Dynamic Web Templates, see Chapter 19, “Using Dynamic Web Templates.”
JavaScript	Creates a new JavaScript file (with a .js file extension). JavaScript entered in that file can then be used on one or more pages by linking the file to the desired page.	For more information on JavaScript files, see Chapter 22, “Client Scripting.”
XML	Creates a new, empty XML file.	For more information on XML files, read <i>Special Edition Using XML</i> from Que Publishing.
Text File	Creates an empty text file.	
Create from Dynamic Web Template	Creates a new page based on a Dynamic Web Template. After selecting this option, you are prompted for the Dynamic Web Template on which the page should be based.	
Create from Master Page	Creates a new ASP.NET Web form based on an ASP.NET Master Page. After selecting this option, you are prompted for the Master Page on which the page should be based.	



note

The Create from Dynamic Web Template and Create from Master Page options only appear if a site is already open in Expression Web.

ASP.NET Pages

The page types shown in Table 3.2 are available when you choose ASP.NET in the New dialog.

Table 3.2 ASP.NET Page Formats

Type	Description	More Information
ASPX	Creates a new ASP.NET Web form. This is the same option available in the General section.	
Master Page	Creates a new ASP.NET Master Page. This is the same option available in the General section.	
Web User	Creates a new ASP.NET user control that can then be inserted into an ASP.NET Web form.	For more information on ASP.NET user Control, see Chapter 30, “Using ASP.NET Web Parts.”

Table 3.2 Continued

Type	Description	More Information
Web Configuration	Creates a new web configuration file (called <code>web.config</code>) that can be used to configure the settings for an ASP.NET 2.0 application.	For more information on using the <code>web.config</code> file, read <i>Special Edition Using ASP.NET</i> from Que Publishing.
Web Configuration 3.5	Creates a new web configuration file (called <code>web.config</code>) that can be used to configure the settings for an ASP.NET 3.5 application.	ASP.NET 3.5 applications allow for the use of ASP.NET Ajax. For more information on using ASP.NET Ajax, see Chapter 31, "Using ASP.NET Ajax."
Web Configuration 4.0	Creates a new web configuration file (called <code>web.config</code>) that can be used to configure settings for an ASP.NET 4 application.	
Site Map ASP.NET	Creates a new ASP.NET site map that can be used in conjunction with ASP.NET navigation controls.	For more information on ASP.NET site maps, see Chapter 26, "Using ASP.NET Navigation Controls."
Create from Master Page	Creates a new page based on an existing ASP.NET Master Page. This is the same option available in the General section.	

CSS Layouts

When you choose the CSS Layouts option in the New dialog, you have the option of selecting from a series of page layouts. When you choose a particular layout, Expression Web creates a new CSS file as well as a new HTML page and then links the new CSS file to the HTML page. Code also is added to both the HTML page and the CSS file that creates the layout you choose.

After selecting the desired layout, you can modify the CSS file to customize the appearance of the page.

➔ *For more information on modifying CSS files, see Chapter 18, "Managing CSS Styles."*

Style Sheets

The Style Sheets section offers a wide assortment of precreated CSS files that you can use in your site. Unfortunately, Expression Web doesn't provide a preview of the styles present in the CSS files, so you will need to apply one to your site to see what kind of formatting it will provide.



tip

The CSS Layouts option allows you to easily create page layouts that are CSS-based instead of using tables or other traditional layout techniques. CSS-based layouts are growing in popularity, largely because today's browsers are capable of rendering them consistently. However, if you choose to use a CSS layout, you should check your site carefully in a variety of browsers.



note

As long as you have Service Pack 2 or later installed, Expression Web 4 does support CSS 3. However, none of the existing CSS layouts use CSS 3 features.

These CSS files are a great way to move to CSS, as long as you are willing to do some modification to them when they don't give you what you want. For example, in my experience, the colors applied by these CSS files are a bit loud and unpleasant in appearance. However, they are easily modified.

Frames Pages

The Frames Pages section provides a series of frame layouts. When you choose one of the frame layouts, Expression Web creates all the pages that make up the layout for you.

➔ *For more information on using frames, see Chapter 6, "Using Frames."*

Importing Files

Adding content typically is more than just entering and configuring text. In many cases, you also are adding graphic files, Flash animations, video files, and other files.

To import files into your site, select File, Import, File to display the Import dialog shown in Figure 3.2.

Click the Add File button to import one or more files. You can also click Add Folder to import a folder and all the files inside that folder. After you've added files or folders, they appear in the list in the Import dialog. If you want to change the name of an imported file, select the file and click the Modify button.

To complete the import process, click OK. The files will not be imported into the site until you click OK.



tip

You might be better off using the CSS files supplied with Expression Web as a way to learn CSS techniques. After you've gained a better understanding of CSS, it's often best to create your own CSS files so you have complete control over them.



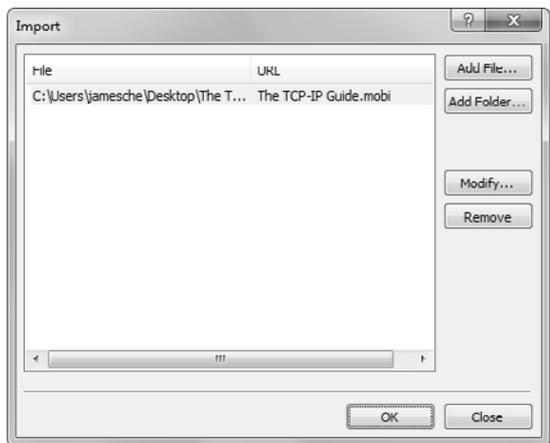
tip

Files and folders added to the Import dialog remain there until you complete the import process, remove them, or close the site. That means you can add files or folders and then click Close and import them later.

If you close a site with files still on the import list, Expression Web asks whether you want to import them before closing the site.

Figure 3.2

You'll likely want to import some graphics and other files into your site. The Import dialog makes it easy.



Formatting Text

Adding text to a page in Expression Web is similar to creating a document in your favorite word processor. Many of the same tools are available.

I'm not going to go into the details of basic text formatting because I'm sure you already understand how to bold and italicize text. Instead, I explain how to work with HTML-specific formatting.

How Expression Web Formats Text

Before you start formatting text in your page, you should become familiar with how Expression Web applies formatting. Because Expression Web is designed to create standards-compliant sites, it often uses CSS to apply formatting. How it goes about doing so is based on the CSS Mode setting.

The current CSS Mode setting is displayed using an icon in the status bar when a page is open. To adjust the CSS Mode setting, click the CSS Mode icon on the status bar and select the desired mode, as shown in Figure 3.3.

By default, the CSS Mode setting is set to Auto. In this mode, Expression Web applies formatting using CSS rules that are configured on the CSS tab of the Page Editor Options dialog.

➔ *For more information on using the CSS tab in the Page Editor Options dialog, see Chapter 11, "Configuring Page Editor Options."*

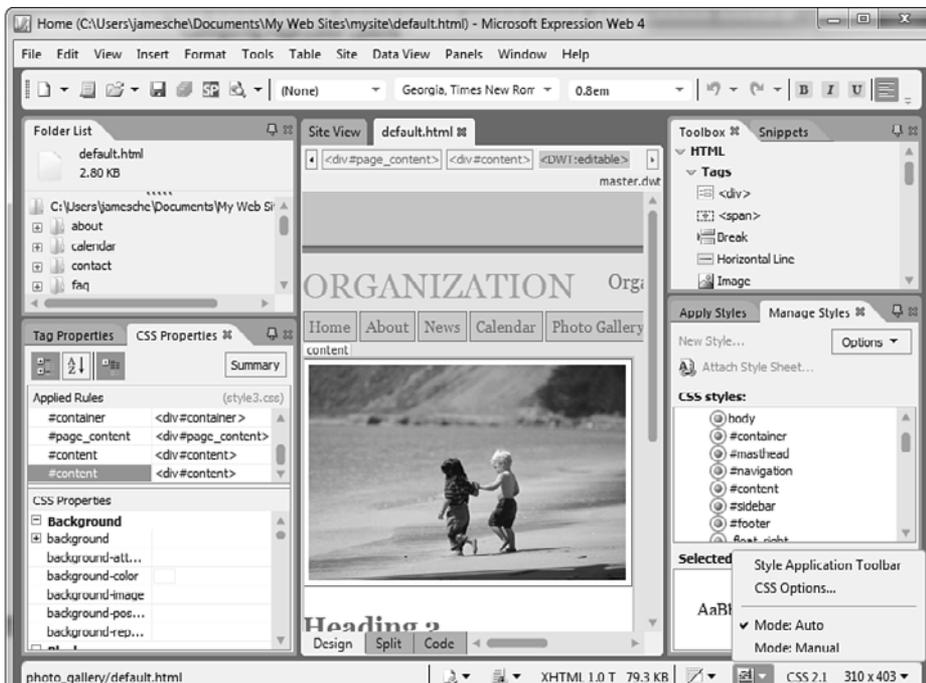


Figure 3.3
The CSS Mode setting controls how styles are applied and can be adjusted via the status bar.

When the CSS Mode setting is Manual, Expression Web automatically displays the Style Application toolbar shown in Figure 3.4. The Style Application toolbar allows you to specify how to apply styles when edits are made.

Figure 3.4
You can control how styles are applied by setting the CSS Mode to Manual and using the Style Application toolbar.



When the CSS Mode setting is set to Auto (the default), you are at the mercy of Expression Web when it comes to where CSS styles get defined. In many cases, that's a suitable situation. However, you might want more granular control over where styles are created. For example, you might have a CSS file attached to the current page, and you may want to ensure that all newly created CSS styles are created inside that CSS file.

By setting the CSS Mode setting to Manual, you can use the Style Application toolbar to specify where new styles are created using the Target Rule drop-down shown in Figure 3.4. Based on what is currently selected, the following options are available in the Target Rule drop-down.

- **An Existing CSS Element**—If the selected page element already has a CSS class or ID applied to it, you can select the CSS ID or class from the Target Rule drop-down. If you then make modifications to the selected element, Expression Web modifies the CSS class or ID applied to it.
- **Inline Style**—If the selected element has an inline style applied to it, selecting Inline Style from the Target Rule drop-down causes the inline style to be modified when formatting is changed.
- **New Inline Style**—Formatting changes are reflected in a new inline style.
- **New Auto Class**—Causes Expression Web to create a new, auto-generated CSS class for any formatting changes that are applied. Expression Web creates the new CSS class in an embedded style sheet inside the current page.
- **Apply New Style**—Allows you to create a new style using the New Style dialog. The location for the new style can be specified in the New Style dialog.

➔ *For more information on creating and editing CSS styles, see Chapter 17, “Creating Style Sheets.”*

**tip**

Font families that you create are used in the Font dialog, the Font drop-down, and in IntelliSense, which is displayed when editing HTML and CSS code.

Font Families

In a word processing application, the font you apply to text will always be the font the viewer of the document sees. Web pages don't work that way. For you to see text formatted in a particular font on a page, that font must be installed on your machine. If it's not, the browser substitutes a font you have.

It's a good idea to always specify a series of fonts to use and specify a generic font (such as sans-serif and monospace) as the last choice. By doing this, if a visitor to your site doesn't have the first font listed, the browser attempts to use the next font in the list and so on. You can easily configure fonts in Expression Web by selecting Format, Font to display the Font dialog shown in Figure 3.5. Alternatively, you can use the Font drop-down on the main toolbar.

note

Some fonts are installed on almost all computers; Arial, Helvetica, and other common sans-serif fonts are considered the safest of all fonts. Times New Roman (a serif font) is another common font. It is the font Expression Web uses by default.

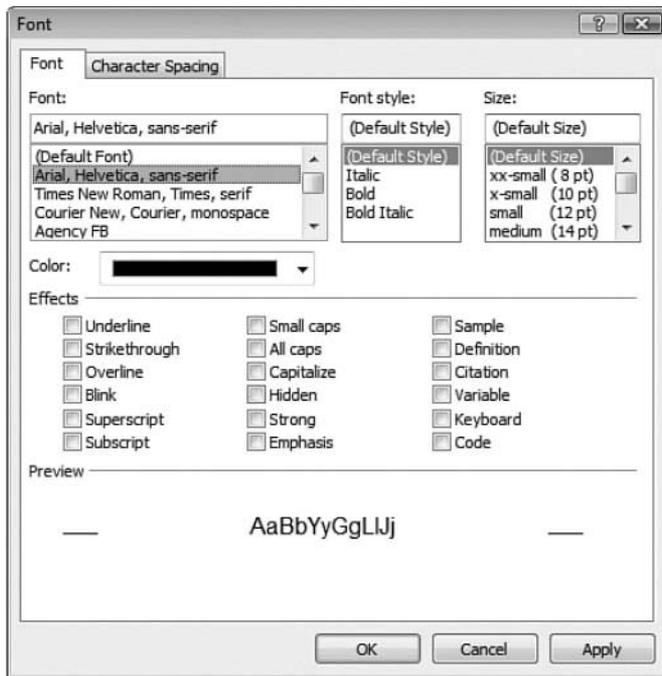


Figure 3.5
The Font dialog lets you easily choose a font family and otherwise modify the appearance of text.

➔ For information on customizing and configuring font families, see Chapter 11, “Configuring Page Editor Options.”

Font Sizes

Font size is one of those things many developers take for granted. Many Web designers say you should always use pixel dimensions (for example, 12 px) for font sizes. In fact, if you want your site to be as user-friendly and accessible as possible, you should use relative font sizes.

➡ *For complete details on making your site compliant with accessibility standards, see Chapter 12, “Maintaining Compatibility and Accessibility.”*

When you use the Font Size drop-down on the toolbar or adjust the font size using the Font dialog, Expression Web automatically uses relative font sizes. The following code shows a CSS style that defines a relative font size that equates to approximately 14-point type:

```
.mainText {  
    font-size: medium;  
}
```

➡ *For more information on CSS styles, see Chapter 17, “Creating Style Sheets.”*

Using relative fonts is a good idea because it gives your site visitor control over the size of the text. When you change the text size in your browser, text that is sized with a relative font size resizes accordingly while text that is sized with an absolute size remains the same size.

Creating Hyperlinks

Expression Web provides tools to create hyperlinks quickly and easily. To create a hyperlink, select the text or graphic you'd like to use for the link and then select Insert, Hyperlink to display the Insert Hyperlink dialog shown in Figure 3.6.

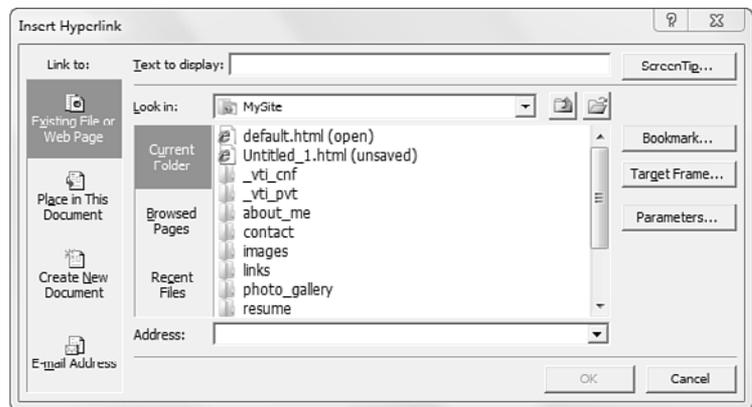


tip

You can also right-click the selection and select Hyperlink from the menu to display the Insert Hyperlink dialog.

Figure 3.6

The Insert Hyperlink dialog contains all the tools needed to create your hyperlinks.



Along the left edge of the Insert Hyperlink dialog are four buttons that define the type of hyperlink to create. By default, Existing File or Web Page is selected. You can also choose to create a link to a place in the current document (an HTML bookmark), a link to a new document, or an email address.

Targeting Hyperlinks

By default, hyperlinks to other pages or other sites load into the current browser window. In many cases, that might be exactly what you want, especially when linking to files within your own site. However, if you are linking to another site or you have some other reason for wanting the current page to still be available after someone clicks your link, you need to override that default behavior.

To change the window in which a hyperlink opens, specify the target for the hyperlink. Click the Target Frame button in the Insert Hyperlink dialog to display the Target Frame dialog shown in Figure 3.7. Expression Web provides a list of common target frames, but you can also specify your own frame name when you are targeting a specific frames page.

➔ *For more information on using frames, see Chapter 6, “Using Frames.”*

➔ *For more information on using the Open Browser Window behavior, see Chapter 21, “Using Behaviors.”*

Hyperlink Parameters

When you’re using ASP.NET, PHP, or another server-side technology, you might need to pass information to the web server in a hyperlink in the form of a query string. A query string is appended to the end of a hyperlink as a series of names and values. Code that runs on the web server can access the values that are passed by referencing the name in the query string.

The following hyperlink passes a value called prod with a value of 6:

```
cool.aspx?prod=6
```

Note that a question mark appears before the query string value.

You can easily create a query string hyperlink in Expression Web by clicking the Parameters button in the Insert Hyperlink dialog. Doing so displays the Hyperlink Parameters dialog. In Figure 3.8, you see the hyperlink parameter that created the hyperlink shown previously.

The Hyperlink Parameters dialog is convenient if you have a lot of parameters and want to work with them often, but I find it just as easy to type parameters manually.



tip

To open a new window that is a specific size or at a specific position, do not use a hyperlink target. Instead, use the Open Browser Window behavior so you can fully control the new window.

New windows opened using this method can be named with any name you choose, and you can then use that name as the target for any hyperlinks you want to load in the window.

Figure 3.7
The Target Frame dialog lets you easily specify a target for a hyperlink.

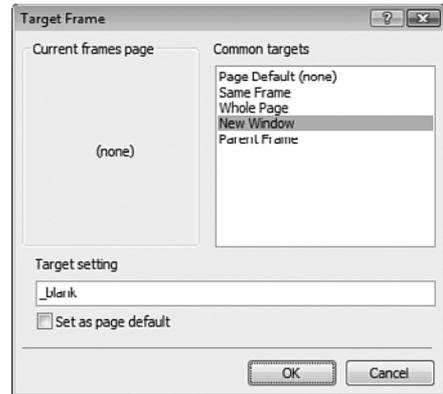
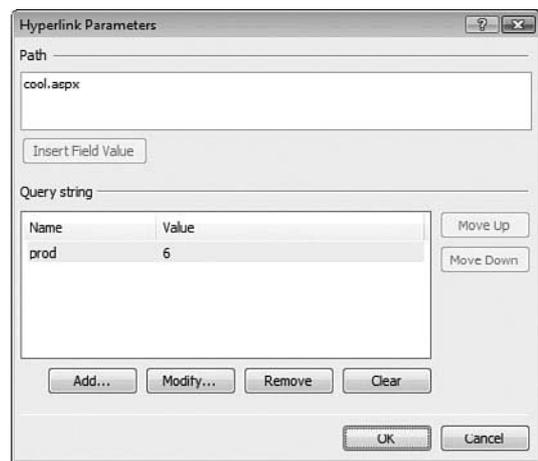


Figure 3.8
The Hyperlink Parameters dialog allows you to create query string parameters in a hyperlink. It's actually just as easy to enter them into the hyperlink text.



HTML Bookmarks

Most hyperlinks take you to the top of the page to which it is linked, but you can also link to a specific point in a page using an HTML bookmark.

An HTML bookmark defines a point within a document. To create a link to an HTML bookmark, you first must create the bookmark in the page. Place the insertion point on the line where you want the bookmark, and then select Insert, Bookmark to display the Bookmark dialog shown in Figure 3.9.



tip

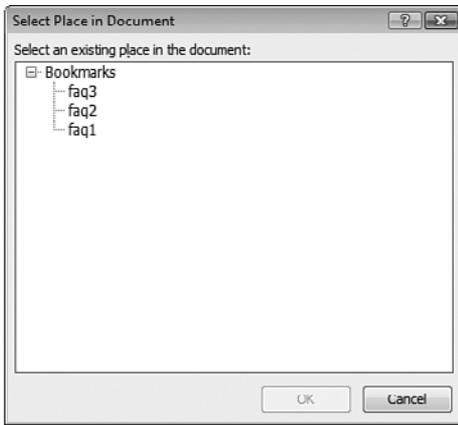
HTML bookmarks are often used on FAQ pages so readers can easily navigate directly to a specific place in the page.

After a bookmark has been created, you can link to the bookmark using the Bookmark button in the Insert Hyperlink dialog to display the Select Place in Document dialog, as shown in Figure 3.10. Select the bookmark for your hyperlink to link to that point in the document.

You can link to bookmarks on pages in external sites as well. When you click the Bookmark button, Expression Web parses the remote page for bookmarks and displays them in the Bookmark dialog.

**Figure 3.9**

HTML bookmarks are managed in the Bookmark dialog and allow you to link to a specific point within a page.

**Figure 3.10**

Expression Web lists all bookmarks on the page you're linking to so you can select the correct point for your hyperlink.

Hyperlink Screentips

To make your site accessible, include a title attribute for all your links. The title attribute is configured using the ScreenTip button in the Insert Hyperlink dialog. The text you enter for the ScreenTip appears as a pop-up when a site visitor hovers over the link. It will also be read by screen readers.

**tip**

Expression Web notes that ScreenTips are supported by Internet Explorer 4.0 or greater. In fact, ScreenTips are supported by all current browsers and are part of the HTML specification.

Spell-Checking

Nothing ruins a professional appearance faster than spelling errors. Even if you're not a bad speller, you're bound to make a mistake or two, so having background spell-checking is convenient.

By default, Expression Web does not check your spelling as you type. If you want to check your spelling, select Tools, Spelling, Spelling. If Design view is active, Expression Web checks the spelling in the active document. If one or more files or folders are selected in the Folder List, Expression Web gives you the choice of checking spelling for the selected page(s) or for your entire site.

Expression Web underlines words it believes to be misspelled with a red line. Right-clicking the underlined text presents you with a menu, allowing you to ignore the problem or take some action to correct it.



Spelling Errors Not Underlined

If you notice a few words that are definitely misspelled in your site but Expression Web did not underline them with a red line, Expression Web is likely configured so spelling errors are not visible.

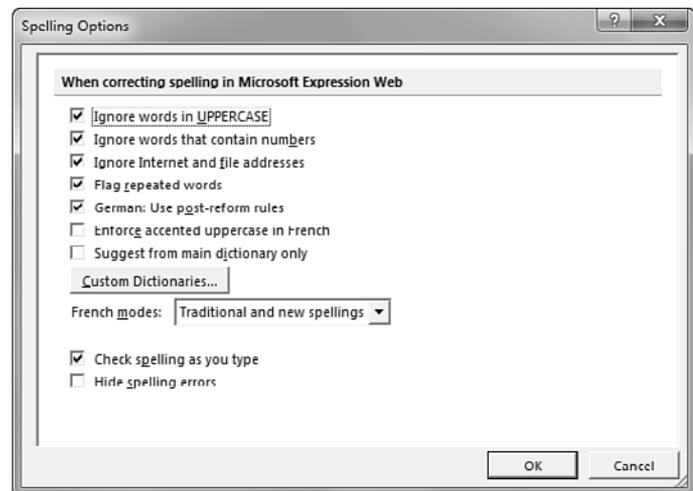
You can also configure Expression Web so that it checks your spelling as you type. To do that, select Tools, Spelling, Spelling Options and make sure Check Spelling As You Type is checked and Hide Spelling Errors is not checked (see Figure 3.11). You can also set other spelling options from this dialog and edit your dictionary.



tip

Unlike previous versions of Expression Web, Expression Web 4 does not share a dictionary with Microsoft Office applications.

Figure 3.11
Spelling options are configured in the Spelling Options dialog. From here, you can also edit your custom dictionary.



Configuring Page Properties

Properties for a specific page are configured using the Page Properties dialog. You can access page properties by opening a page and selecting File, Properties.

The Page Properties dialog has five tabs.

General Tab

The General tab allows you to easily configure the page title, description, keywords, and other general information (see Figure 3.12).



If a Dynamic Web Template or ASP.NET Master Page is applied to the current page, many of the options in the Page Properties dialog will likely be disabled because they will be controlled by the Dynamic Web Template or Master Page.

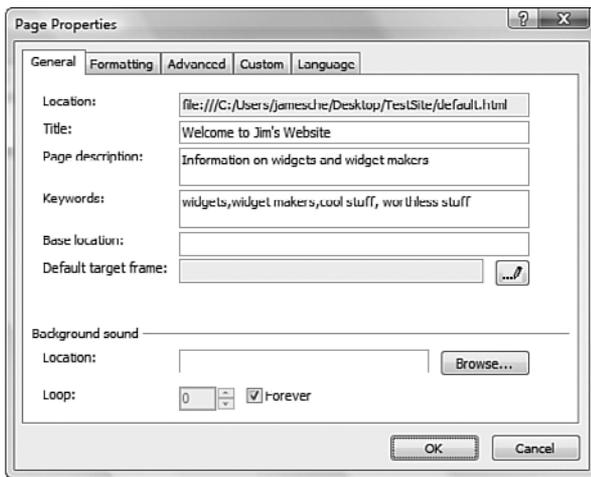


Figure 3.12

The General tab in the Page Properties dialog provides settings for page title and other general page settings.

The title you enter in the General tab displays in the title bar of the web browser when the page is viewed. The description and keywords you enter are added as META tags on the page to aid in search engine indexing.

You can also specify a base location, also known as a base href. The base href is the base URL for the page. For example, if you enter `http://www.mysite.com/products` as the base href, a link that points to `software.htm` will actually link to `http://www.mysite.com/products/software.htm`. This is convenient when your site's directory structure changes often. By using a base href, you can easily update the absolute location of all links by changing one property of the page.

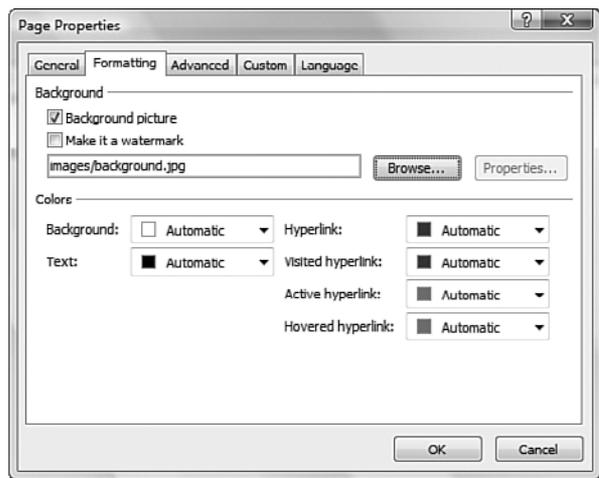
The Default Target Frame property allows you to specify a default frame for links that don't explicitly specify a frame. This is most often used when you want links on a particular page to open in a new window. By setting the default target to New Window, you can force all links to open in a new window by default.

A background sound can also be set on the General tab. The sound is then played when the page is loaded. You can also choose to loop the sound a specific number of times or infinitely. Keep in mind that most people find background sounds annoying and are likely to leave your site if you force them to listen to a background sound. Before you choose to add a background sound, give it careful consideration.

Formatting Tab

The Formatting tab contains properties for a background image, hyperlink colors, and other page colors (see Figure 3.13).

Figure 3.13
Formatting a page's background and colors is accomplished using the Formatting tab.



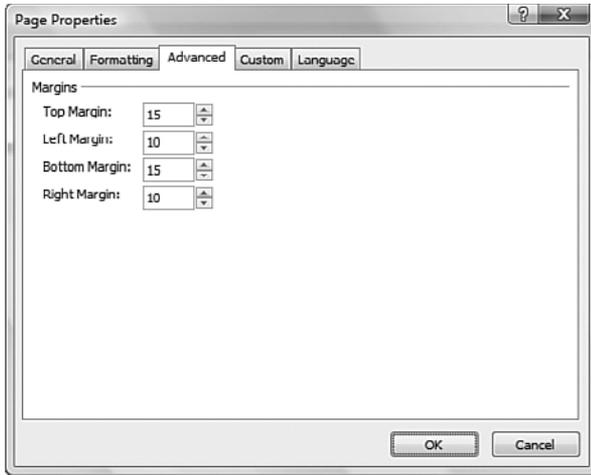
In addition to specifying a background image, you can also specify that an image be added as a watermark. A watermark background remains stationary when the page is scrolled.

Hyperlink color and other page colors are also configured on the Formatting tab. Color changes made here are added to the page as embedded style sheets.

➡ *For more information on style sheets, see Chapter 17, “Creating Style Sheets.”*

Advanced Tab

The Advanced tab, shown in Figure 3.14, contains settings for the page margins. Margin entries made here are applied to the page as an inline style on the <body> tag.

**Figure 3.14**

Page margins are applied as an inline style so they work with all modern browsers.

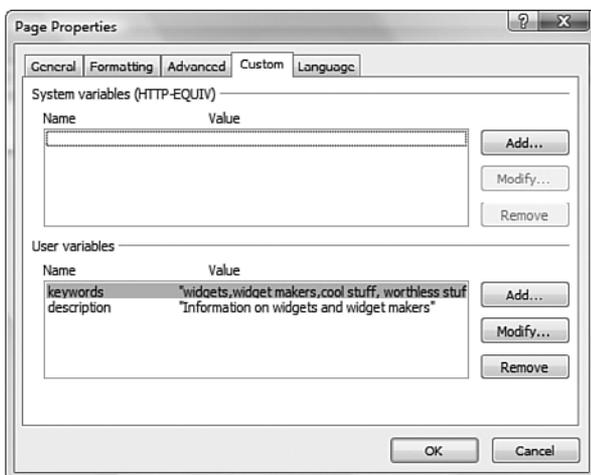
Custom Tab

The Custom tab configures META tags for the current page (see Figure 3.15). The top section is for system variables (HTTP-EQUIV META tags), and the bottom section is for user variables such as keywords and description META tags.

As shown here, any keywords and description information entered on the General tab shown previously in Figure 3.12 carry over to the Custom tab.

note

An in-depth discussion of the META element is outside the scope of this book. If you'd like more information on this powerful HTML element, see <http://www.informit.com/articles/article.aspx?p=28768&seqNum=8>.

**Figure 3.15**

META tags are easily configured on the Custom tab. Note that settings made on the General tab carry over to this tab.

Language Tab

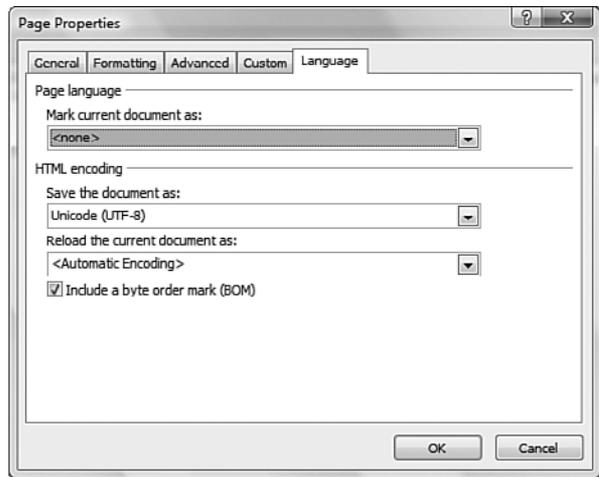
The Language tab configures the language for your page (see Figure 3.16). The language setting determines the page encoding for the page.

Web browsers rely on the page encoding and language settings to determine how to properly display your page. Make sure you have correctly set these for the language and region you are targeting. In almost all cases, the default settings are fine. However, if you are designing a page for a specific region or language, be sure you set the encoding appropriately and test the site for the correct appearance.

The Language tab also enables you to control whether Expression Web includes a byte order mark (BOM) on your page. A *byte order mark* is a hidden sequence of characters at the beginning of a file that can be used to determine encoding characteristics of the file. Some file types (such as PHP files) can experience problems when a BOM is added to the page, so Expression Web enables you to configure which file types will and will not have a BOM added by default. You can override these settings in the Language tab for a particular file as shown in Figure 3.16.

➔ *For more information on configuring BOM settings in Expression Web, see Chapter 11, “Configuring Page Editor Options.”*

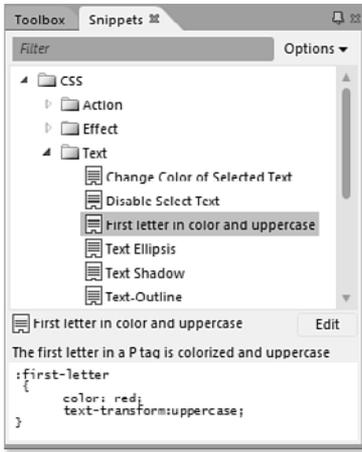
Figure 3.16
Page language and encoding are configured on the Language tab.



Using Code Snippets

Web designers commonly reuse code in several pages. Whether you're reusing a single line of code or a code block, you need to avoid typos at all costs. Code snippets can prevent typos and add a high degree of convenience to code reuse.

To insert a code snippet into your page, you can use the Snippets panel, as shown in Figure 3.17. (If the Snippets panel isn't visible, select Panels, Snippets to display it.) Switch to Code View and double-click on the desired code snippet to add it to your page.

**Figure 3.17**

Code snippets can be added easily by pressing Ctrl+Enter while in Code View.

You can also press Ctrl+Enter while in Code View to display a pop-up of your code snippets. Select a code snippet from the list and press Enter to insert it into your page.



Code Snippets Don't Display

You might press Ctrl+Enter on your page but not get the option of inserting a code snippet; instead, the insertion point keeps moving down a line each time. You can insert a code snippet only when you are in Code View. In Design View, Ctrl+Enter inserts a single line break. In Code View, Ctrl+Enter brings up the Code Snippets IntelliSense pop-up.

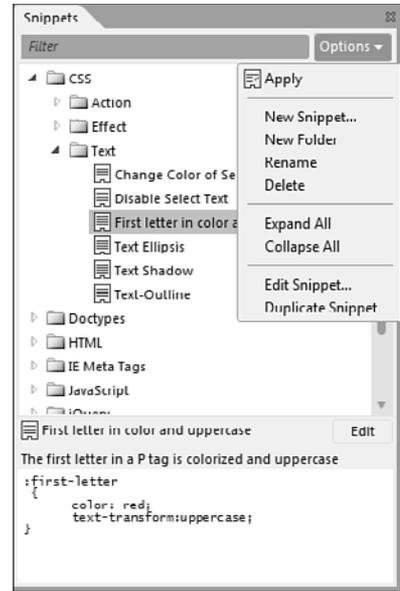
You can easily add, delete, or modify code snippets using the Snippets panel. You can add a new code snippet by clicking Options, New Snippet and entering the required information. You can control where the insertion point appears after inserting the snippet by adding a pipe symbol (|) to the snippet code. If you add two pipe symbols, all code between the two pipe symbols is automatically selected when the snippet is inserted.

When you add your own snippets, they are stored in an XML file located in the AppData\Roaming\Microsoft\Expression\Web 4\Snippets folder in your profile directory. To move your snippets to another machine or back them up (something I strongly recommend), you can simply back up the Snippets.xml file in the Snippets folder.

note

By default, your profile directory is c:\users\

Figure 3.18
Code snippets make reusing code convenient. Expression Web comes with plenty of existing snippets.



Configuring File Editors

By default, when you double-click most files in the Folder List, Expression Web opens that file in its own window. In most cases, that's what you want to happen; in some cases, however, you might want to override that behavior so certain file types are opened in a different application.

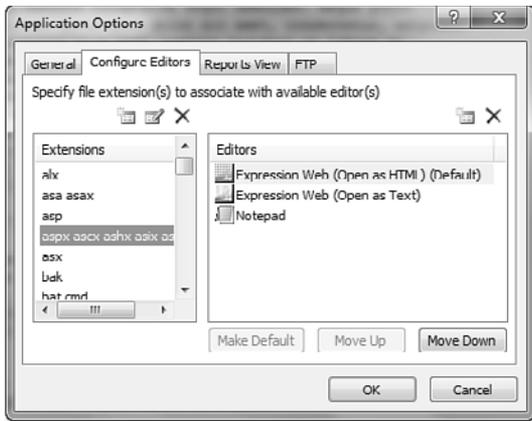
To specify the application to open for a specific file type, select Tools, Application Options and click the Configure Editors tab. The dialog for configuring editors contains a list of file extensions along with the associated editor or editors for each (see Figure 3.19). Available editors can be rearranged in the list or moved to the top of the list so it is used as the default editor. When a file is double-clicked in the Folder List in Expression Web, it will be opened in the default editor for that file type.

note

Some file types (such as images) are not opened in Expression Web by default when you double-click them in the Folder List.

caution

The editor you configure for a specific file extension in Expression Web does not affect Windows file associations. Settings made in Expression Web are specific only to Expression Web.

**Figure 3.19**

The Configure Editors tab allows you to configure specific file types to open with the application of your choice.

The following buttons are available on the Configure Editors tab as shown in Figure 3.19:

- **New Extension**—Allows you to add a new file extension and configure the appropriate editors for it.
- **Modify Extension**—Allows you to change the selected extension. For example, if HTML is selected, you can click Modify Extension to change the extension to HTM. This does not modify the application used to edit that file type.
- **Delete Extension**—Removes the selected extension from the list. If the file type is then double-clicked in Expression Web, you are prompted to select which program should be used to edit that file type.
- **New Editor**—Adds a new editor for the selected file type.
- **Delete Editor**—Deletes the selected editor from the selected file type.

To configure a new file extension, click the New Extension button to display the Open With dialog, as shown in Figure 3.20. Enter the file extension (with no dot in front of it) and select the program to use for editing that file type from the list of available applications. If the program you want to use for the new file type is not listed, click Browse for More and browse to the executable for the application you want to use.

To configure a new editor for an existing file type, select the file type and then click the New Editor button to display the Open With dialog. Select the editor or click Browse for More; then browse to the executable. You can then use the Move Up, Move Down, or Set as Default buttons to adjust the hierarchy of the new editor.

To select from one of the listed editors when editing a file, right-click the file in the Folder List and select Open With as shown in Figure 3.21. Choose the desired editor from the list to open the selected file in that editor.

Figure 3.20

The Open With dialog contains a preset list of applications. Click Browse for More if the application you want to use isn't listed.

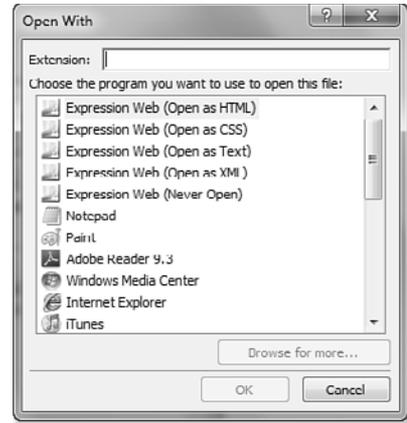
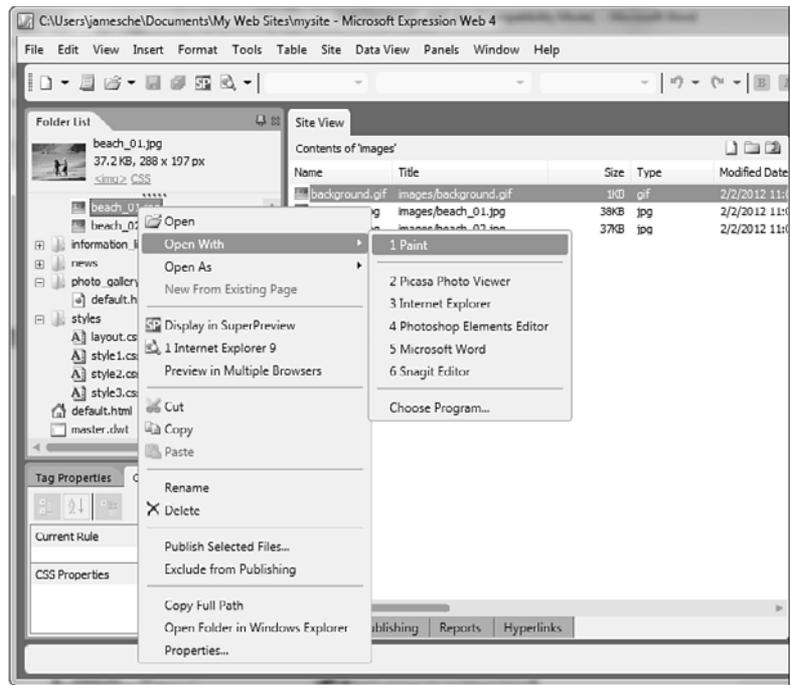


Figure 3.21

Expression Web displays a list of configured editors. Choose the editor you'd like to use to edit the selected file.



Making the Most of Code Snippets

Because Microsoft targeted Expression Web toward designers rather than developers, support for writing code in Expression Web is limited. PHP and JavaScript developers will find some IntelliSense and color-coding (features that are missing for ASP.NET developers), but even those features are limited.

The lack of these coding features is more than just an inconvenience; it can lead to coding errors that are hard to find and can be incredibly frustrating. Code snippets help alleviate such problems.

As your site grows in complexity, you're likely to accumulate a lot of code that you use regularly on your pages. By creating code snippets for sample code you find on the Internet or code you write yourself, you can always have access to a quick and easy repository of code that you can insert without unintentionally creating errors in your site.

USING PAGE VIEWS

Introduction to Page Views

What you see is what you get (WYSIWYG) web design products are supposed to keep you from having to worry about HTML code, but anyone who has spent any amount of time working with a WYSIWYG tool knows that WYSIWYG really stands for “what you see is what you might get.” It’s simply not practical to ignore the code that makes up your page. Fortunately, Expression Web provides three different views of your page so you can work with it the way you want: Design View, Code View, and Split View.

You’ll likely use Design View most often. It shows a view of your page as it will appear in a web browser. Using Design View, you can easily add text, graphics, hyperlinks, and other content and arrange it.

Code View shows the underpinnings of your page—the code. Expression Web offers color-coding, line numbering, IntelliSense, and other tools to make it easy and efficient to work in this view. Chances are this is where you will go to tweak code that Expression Web has already added, but you can add code from scratch here as well.

Split View is a combination of Design View and Code View. Split View divides the Expression Web interface into two panes, one on top of the other. One pane displays the page in Design View, and the other pane displays the page in Code View. Separating the two panes is a movable divider called the split so you can decide how much screen real estate each view should occupy.

Adding content in Design View is primarily covered in Chapter 3, “Creating Pages and Basic Page Editing,” but in this chapter, we go over some of the features on the View menu in Expression Web and cover using Code View and Split View.

Working in Design View

As mentioned previously, Design View is where you will likely spend most of your time when using Expression Web. Design View is an approximate representation of your page as it will appear in a web browser.

Quite a few interface features accessible on the View menu make it easier to work in Design View.

➔ *For more information on using SuperPreview, see Chapter 13, “Using SuperPreview.”*

Visual Aids

Visual Aids are user interface elements that make it easier to work with pages. Expression Web offers nine Visual Aids to make it easier to position items and locate elements on your page.

To access Visual Aids, select View, Visual Aids and select the desired Visual Aid. Visual Aids are displayed by default in Expression Web, but you can turn them off by selecting Show in the Visual Aids menu or by pressing Ctrl+/ on your keyboard. The Show menu item toggles all Visual Aids on and off.

The following Visual Aids are available on the Visual Aids menu:

Block Selection

When Block Selection is enabled (it is enabled by default), Expression Web displays a block selector for HTML block elements. A block selector is a small tag indicator, as shown in Figure 4.1. Block Selection also displays a shaded area indicating the padding around a particular element.

Some block elements are almost impossible to work with when block selectors aren't visible. Expression Web does not hide these block elements even when block selectors are disabled. Expression Web layers are an example. Layers are implemented using HTML <div> elements, and the <div> block selector is always visible regardless of whether the Block Selection Visual Aid is enabled.

➔ *For more information on using layers, see Chapter 23, “Using Layers.”*



note

Expression Web does not use Internet Explorer's rendering engine to display your page in Design View. Instead, Expression Web uses a custom rendering engine designed to give you an approximate view of your page as it appears in any modern browser.

If you want to see how your page appears in a particular browser, use SuperPreview, the Snapshot panel, or the Preview in Browser feature.



tip

The status bar in Expression Web displays the current status of Visual Aids. You can turn Visual Aids on or off by double-clicking the Visual Aids area of the status bar.

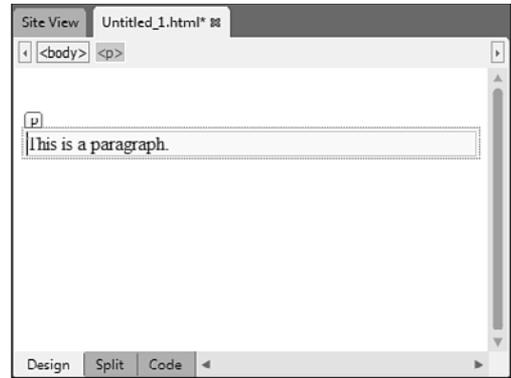


tip

HTML block elements are elements that can appear within the <body> of the document. The W3C has more information at www.w3.org/TR/html4/struct/global.html#h-7.5.3.

Figure 4.1

The Block Selection Visual Aid makes it easier to work with block elements such as paragraphs.



Visible Borders

The Visible Borders Visual Aid displays borders around particular HTML elements, even when they are not selected. For example, when you insert a table in Expression Web, the Visible Borders Visual Aid renders a dotted border around each cell by default, even when the table has no border. When the Visible Borders Visual Aid is disabled, the table will be invisible in Design View unless it is selected.

➡ For more information on using tables, see Chapter 5, “Using Tables.”

Empty Containers

The Empty Containers Visual Aid displays borders around certain elements that act as containers for other elements. For example, an HTML form is visible by default in Design View, even when the form has no controls in it. If the Empty Containers Visual Aid is turned off, HTML forms are invisible in Design View until controls are added to them.

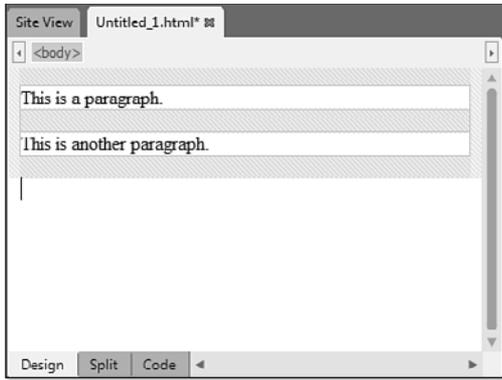
➡ For more information on using HTML forms, see Chapter 24, “Using Form Controls.”

Margins and Padding

The Margins and Padding Visual Aid displays a shaded area around the page margin and around padding areas of all HTML elements, as shown in Figure 4.2.

CSS Display:none and CSS Visibility:hidden

CSS elements with a Display attribute set to none or a Visibility attribute set to hidden are not visible by default. Expression Web provides a Visual Aid for CSS Display:none and CSS Visibility:hidden elements.

**Figure 4.2**

The Margins and Padding Visual Aid makes it easy to see padding around elements and the page's margins.

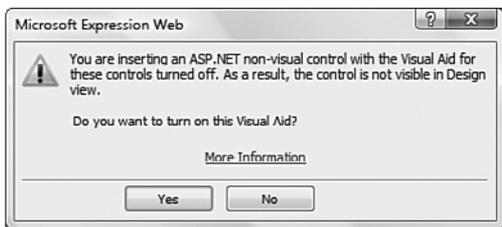
When this Visual Aid is enabled, the elements will be visible in Design View but not in a web browser.

➔ *For more information on using CSS in Expression Web, see Chapter 18, “Managing CSS Styles.”*

ASP.NET Non-visual Controls

Some ASP.NET controls are not visible by default. For example, the `HiddenField` control is an invisible control. The ASP.NET Non-visual Controls Visual Aid allows you to see invisible controls in Design View.

If the ASP.NET Non-visual Controls Visual Aid is not turned on and you insert a non-visual ASP.NET control, Expression Web displays a dialog warning you that the element will not be visible; it also asks whether you want to enable the ASP.NET Non-visual Controls Visual Aid, as shown in Figure 4.3.

**Figure 4.3**

Expression Web notifies you when you insert a non-visual ASP.NET control.

➔ *For more information on using ASP.NET controls in Expression Web, see Chapter 25, “Using Standard ASP.NET Controls.”*

ASP.NET Control Errors

By default, Expression Web displays an error message when ASP.NET controls are improperly configured or if they malfunction. If you disable the ASP.NET Control Errors Visual Aid, no such error message will be displayed.

Template Region Labels

Editable regions are areas of a page that are used in conjunction with Dynamic Web Templates. By default, a label containing the name of each editable region is available only when the editable region is selected. By enabling the Template Region Labels, you can see the names of all editable regions whether they are selected or not.

➔ *For more information on using Dynamic Web Templates and editable regions, see Chapter 19, “Using Dynamic Web Templates.”*

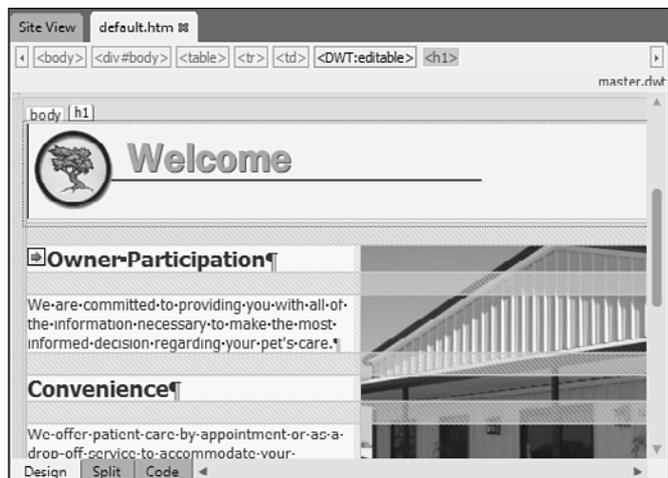
Formatting Marks

Expression Web can display formatting marks for paragraphs, HTML comments, scripts, and other elements that normally are not visible.

To display formatting marks, select View, Formatting Marks, Show or press Ctrl+Alt+/ on your keyboard. After the formatting marks are visible, you can toggle individual types of formatting marks on and off as needed.

Formatting marks are useful when working with complex pages. For example, in Figure 4.4, a formatting mark is displayed just to the left of “Owner Participation” indicating a right-aligned HTML element. By clicking on the formatting mark in Design View, you can easily select the element that is right-aligned.

Figure 4.4
Formatting marks can make page editing easier by allowing easy access to invisible page elements.



Some formatting marks offer powerful capabilities. If you double-click a script formatting mark, you can edit the script it represents, as shown in Figure 4.5.

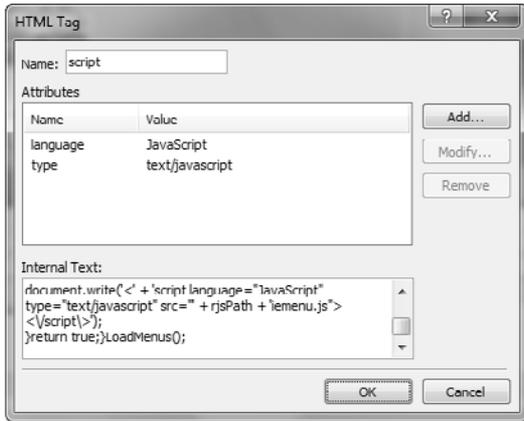


Figure 4.5 Formatting marks make it easier to edit scripts and other non-visual elements.

Ruler and Grid

The Ruler and Grid feature of Expression Web can make it much easier to position items in your page, especially if you are using absolute positioning with CSS.

➔ *For more information on CSS positioning, see Chapter 18, “Managing CSS Styles.”*

The Ruler makes positioning easier by displaying horizontal and vertical indicators that originate from the upper-left corner of the page by default, as shown in Figure 4.6.

As you move your mouse over the design surface, the Ruler shows the precise location of the mouse pointer by displaying position indicator marks on both the vertical and horizontal rulers. This makes positioning and resizing objects easier. For example, when you are dragging a layer to a new position, it's helpful to have a visual representation of the mouse pointer's exact coordinates.

As previously mentioned, the Ruler's origin is the upper-left corner of the design surface. You can, however, move the origin in one of two ways: You can drag the upper-left corner of the ruler to a new location or you can select something on the page, right-click the Ruler, and select Set Origin From Selection. In Figure 4.7, a layer is selected. When Set Origin From Selection is selected on the menu, the origin of the Ruler is reset to the upper-left corner of the layer. To reset the origin to its original location, right-click the Ruler and select Reset Origin.



tip

When you double-click an HTML comment formatting mark, you can edit the HTML comment without switching to Code View.

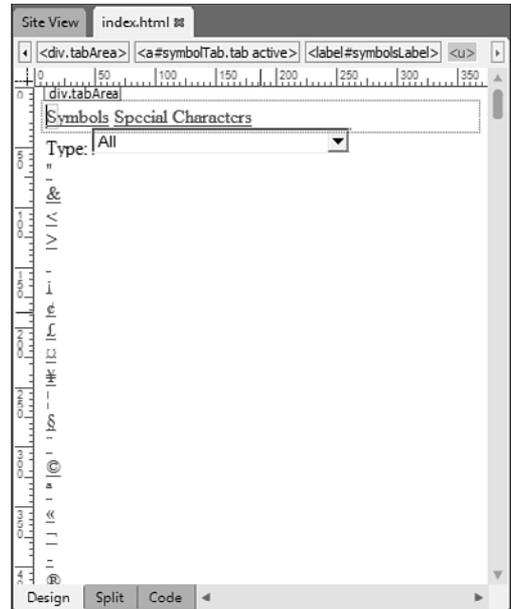


tip

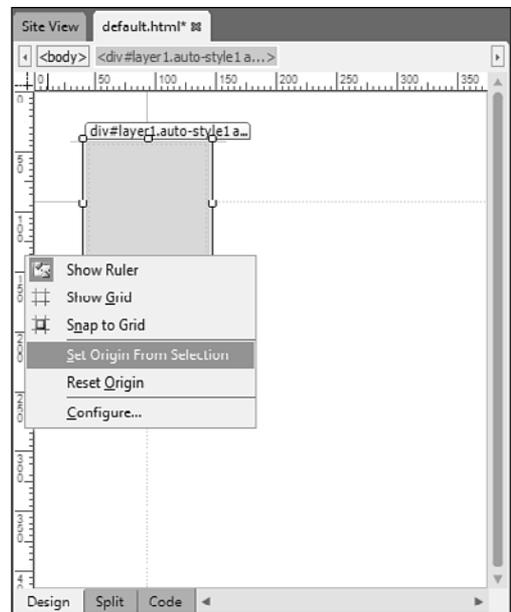
The Ruler uses the same coordinate system as absolute positioning. By default, the upper-left corner of Design View is at position 0, 0. This is called the *origin*.

Figure 4.6

The Ruler adds convenient horizontal and vertical positioning indicators to Design View. The indicators appear at approximately 175 pixels on both rulers in this figure.

**Figure 4.7**

The Ruler's origin can be reset to a selection on the available context menu by right-clicking the Ruler.



Although the Ruler can make it easier to position items, it's not perfect. The layer shown in Figure 4.7 can be repositioned by clicking anywhere on the block selector and dragging. Unfortunately, the block selector itself consumes several pixels on the screen, so the position of the mouse pointer can vary when positioning the layer. Therefore, you cannot rely solely on the Ruler and position indicator marks. To make the Ruler more useful in such scenarios, the Grid can be used to make positioning more precise.

To enable the Grid, select View, Ruler and Grid, Show Grid. When the Grid is enabled, vertical and horizontal lines are drawn on the design surface. By default, these lines are drawn at 50-pixel increments, as shown in Figure 4.8.

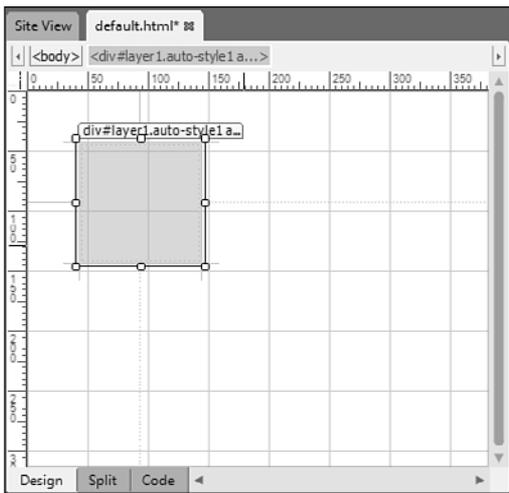


Figure 4.8

Using the Grid in combination with the Ruler makes positioning objects on the page easy.

If you prefer, you can enable the Snap to Grid feature so that objects automatically snap to the grid when positioned. To enable Snap to Grid, select View, Ruler and Grid, Snap to Grid.

Both the Ruler and Grid are customizable by selecting View, Ruler and Grid, Configure. When you select Configure, the Page Editor Options dialog is opened and the Ruler and Grid tab is selected automatically, as shown in Figure 4.9. You can configure the units used for the Ruler as well as the units, spacing, and other settings for the Grid.

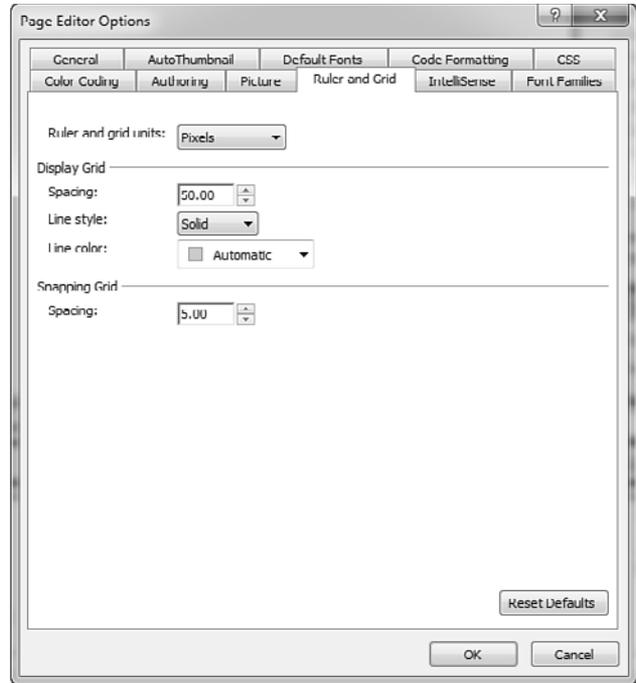
Tracing Images

Almost all sites are designed using a combination of text, graphics, and other dynamic content. In many cases, graphics are used to present interactive user interfaces. Even simple designs use graphics extensively to add user interface elements.

When graphics are used for a site's user interface, a large graphic is often cut up into smaller graphics in a process known as *slicing*. Tables or CSS styles are then used to segment the interface, add links, and so on. This whole process can be tedious and time-consuming. The Tracing Image feature in Expression Web can make it considerably easier by allowing you to superimpose your user interface onto the design surface so you can build your layout around it easily.

Figure 4.9

The Ruler and Grid can both be customized using the Page Editor Options dialog.



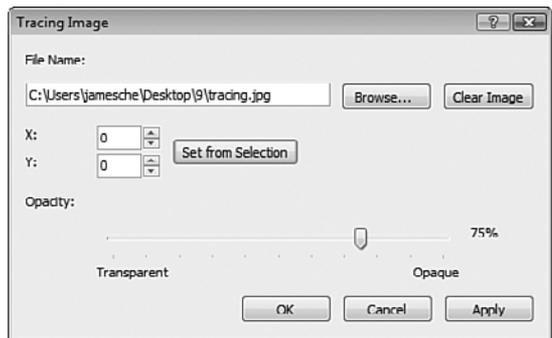
To use the Tracing Image feature, select View, Tracing Image, Configure to display the Tracing Image dialog shown in Figure 4.10. Click Browse and select the figure to use for the tracing image. By default, the tracing image is positioned at the upper-left corner of the page. You can specify a different location or click Set from Selection to position the image relative to the selected element on the page.

note

Although it's true that many web designers use tables to arrange image slices, using CSS is a more widely accepted method.

Figure 4.10

Tracing images can be positioned and adjusted using the Tracing Image dialog.





Tracing Image Disabled

The Tracing Image menu option is disabled on pages that use a Dynamic Web Template or an ASP.NET Master Page. To use a tracing image in this scenario, open the Dynamic Web Template or the Master Page and apply the tracing image.

Because you are building your user interface elements on top of the tracing image, Expression Web allows you to configure transparency for the image. This allows you to configure the image so it's still visible while not allowing it to get in the way of your design elements.

After you've configured the tracing image with the desired settings, click OK to add it to the page. You can then add the user interface elements you will use to lay out your page.



Tracing images are only visible when you are viewing your page in Expression Web.

Adjusting Page Size

When designing your site, you want to make sure it looks the way you intend at all common resolutions. For example, if you design your site on a monitor set to a resolution of 1280×1024 (a common resolution for a 17" or larger LCD display) and then view it on a monitor set to a resolution of 800×600, it's likely that part of the page will overflow the right edge of the screen. User interface elements might also shift around, making the page look unprofessional.

One solution to the screen resolution problem is to design your pages at the lowest resolution in which you expect them to be viewed. (In the example given previously, you would want to design the page at a resolution of 800×600.) There's a problem with this approach, however. You might have spent a bunch of money on a nice monitor with plenty of screen real estate, and now you're being asked to lower the resolution to the point where you're no longer able to take advantage of all of that real estate! Luckily, the Page Size feature in Expression Web was created so you can design for a specific resolution without actually switching to that resolution.

When you're working in Design View, Expression Web displays the current page size in the status bar, as shown in Figure 4.11.

By clicking the page size displayed in the status bar, you can select a new page size, as shown in Figure 4.12. You can also select View, Page Size and select the desired page size from the menu.

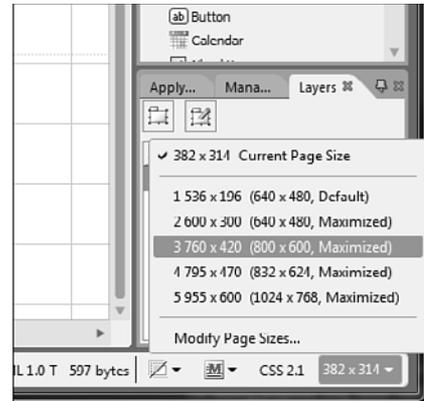


Figure 4.11

The status bar shows you the page size so you can design for an intended resolution.

Figure 4.12

Setting a new page size is fast and easy using the status bar.

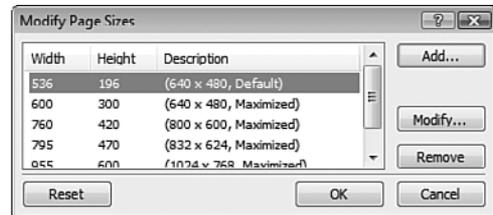


If you choose a page size that is smaller than the current size of the design surface, Expression Web displays shaded borders to indicate the size you have chosen, and you will not be able to add any content outside the selected page size. Using this method, you can take advantage of your entire screen while still ensuring that your page displays appropriately at the lowest desired resolution.

The page sizes that appear on the Page Size menu can be modified by selecting View, Page Size, Modify Page Sizes. The Modify Page Sizes dialog (shown in Figure 4.13) lets you easily add new page sizes or remove or modify existing ones.

Figure 4.13

Page sizes displayed in Expression Web can be modified using the Modify Page Sizes dialog.



Working in Code View

Most users of WYSIWYG design tools aren't too interested in digging into the code that makes up a page. In fact, the code that Expression Web generates is clean and efficient enough to make digging into the code unnecessary in almost all cases. However, there are still times when you might want to add content to code or modify existing code, if for no other reason than to learn more about HTML, XHTML, CSS, and scripting.

Expression Web marks invalid code in Code View by adding a squiggly, red underline to it. It also marks some code errors (such as missing closing tags) by highlighting them in yellow. These visual indicators are a great bonus when trying to locate code problems.

To access Code View, click the Code button at the bottom of the design surface. Expression Web's Code View (shown in Figure 4.14) is a robust environment where you can work with code efficiently. Some of the tools that make working in Design View easy are also available in Code View, and code-specific tools are available that let you work with code even if you're not an expert in HTML, CSS, and scripting.

As shown in Figure 4.14, several features have been designed around making code more readable in Expression Web. Code is indented to make it more readable. Line breaks are also inserted automatically in tables so code is easier to understand. What you can't see in Figure 4.14 (due to the gray-scale image) is the color-coding that makes it much easier to read the code.

Several other features make working in Code View easy and intuitive.

**tip**

If you select something in Design View and then switch to Code View, the code representing the selection is selected as well. The same thing happens when code is selected in Code View and you switch to Design View.

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
2 <!-- saved from url=(0014):about:internet -->
3
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6   <title>Insert Symbol</title>
7   <meta content="en-us" http-equiv="Content-Language" />
8   <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
9   <script type="text/javascript" src="script/jquery-1.4.1.min.js"></script>
10  <script type="text/javascript" src="script/logical.js"></script>
11  <script type="text/javascript" src="script/prepare.js"></script>
12  <script type="text/javascript" src="script/load.js"></script>
13  <script type="text/javascript" src="script/utilities.js"></script>
14  <link href="styleLayout.css" rel="stylesheet" type="text/css" />
15 </head>
16 <body>
17   <div id="page_container">
18     <div id="windowDialogBackground">
19       <div>
20         <div class="tabArea">
21           <a href="#" id="symbolTab" class="tab active"><label id="symbolsLab
22           <a href="#" id="specialCharTab" class="tab"><label id="specialLabel
23         </div>
24       </div>
25       <div id="contentBackground">
26         <div id="tab_content_1">
27           <div id="groupHeaderBackground">
28             Type:
29             <select class="menu" id="characterType">
30               <option value="All">All</option>
31               <option value="HTML">HTML 3.2</option>
32               <option value="Currency">Currency Symbols</option>
33               <option value="Punctuation">Punctuation and Typography</opt
34               <option value="Mathematical">Mathematical Symbols</option>
35               <option value="Arrows">Arrows</option>
36               <option value="Accented">Accented Characters & Accents</opt
```

Figure 4.14

Code View is a robust environment for working in code, but it's also easy for those who know little about coding.

Customizing Code Formatting

The formatting of code can be customized by selecting Tools, Page Editor Options and clicking the Code Formatting tab. As shown in Figure 4.15, there are many options to configure. By selecting a tag from the Tags list, you can customize that particular tag. CSS code can also be customized.

At the top of the Code Formatting tab are settings that apply to all code in Code View. However, these settings do not affect existing code. Only page edits and new pages are affected by changes to these settings.

The following check boxes are available in the upper-left section of the Code Formatting tab:

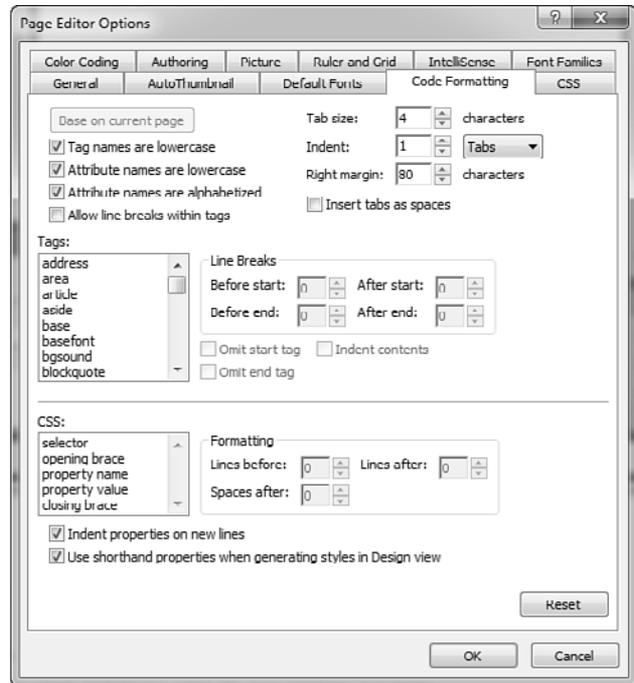
- **Tag Names Are Lowercase**—When this box is checked, HTML tag names are added in lowercase. This is the default setting.
- **Attribute Names Are Lowercase**—When this box is checked, HTML attributes are added in lowercase. This is the default setting.

caution

If you uncheck either of these check boxes, Expression Web creates all new pages with uppercase tag names and attributes. Current Web standards require that HTML tag names and attributes be lowercase. Therefore, pages created with either of these boxes unchecked will not be compatible with existing standards.

Figure 4.15

You don't have to settle for how Expression Web formats your code. You can change the rules any way you choose.



- **Attribute Names Are Alphabetized**—When this box is checked, Expression Web alphabetizes the attributes of HTML elements. This is the default setting.
- **Allow Line Breaks Within Tags**—By default, Expression Web does not add hard line breaks within tags. If this box is unchecked, Expression Web adds a hard line break between attributes within a tag if the length of the tag exceeds the number of characters specified for the right margin.

**caution**

Controlling spacing with non-breaking spaces usually is not recommended. It can make spacing unpredictable and should be avoided.

You can also configure the tab and margin settings in code view by changing the settings in the upper-right of the Code Formatting tab. By default, Expression Web uses tabs as spaces. If you uncheck the Insert Tabs as Spaces check box, Expression Web uses nonbreaking spaces instead.

The Tags section of the Code Formatting tab makes it easy to configure the way specific tags are formatted in Code View. Select a tag from the Tags list and then change the appropriate settings as desired. You can configure whether to omit start tags and end tags and whether contents of the tag are indented as well.

The options available change depending on which tag you have selected. For example, if a tag doesn't have a corresponding end tag (such as the `
` tag), the Omit End Tag check box is checked and disabled.

CSS elements can be formatted using the settings in the bottom section of the Code Formatting tab. You can also control how Expression Web generates more complex CSS by using the Use Shorthand Properties When Generating Styles check box. When this box is checked (the default setting), Expression Web combines CSS properties in one CSS element when possible. For example, if a three-pixel green border is applied to a table cell, Expression Web creates CSS code like the following by default:

```
border: 3px solid #00FF00;
```

If the Use Shorthand Properties When Generating Styles check box is unchecked, Expression Web instead creates the style using the following CSS code:

```
border-top-width: 3px;  
border-right-width: 3px;  
border-bottom-width: 3px;  
border-left-width: 3px;  
border-top-color: #00FF00;  
border-right-color: #00FF00;  
border-bottom-color: #00FF00;  
border-left-color: #00FF00;  
border-top-style: solid;  
border-right-style: solid;  
border-bottom-style: solid;  
border-left-style: solid;
```

**caution**

Unlike the HTML formatting options in the Code Formatting tab, the CSS options affect existing CSS code as well as new CSS code.

If you want Expression Web to use the current page as a guide for formatting rules, click the Base on Current Page button. Expression Web attempts to match the formatting rules as closely as possible to what is on the current page.

Quick Tag Tools

The Quick Tag Tools are context-sensitive buttons that appear at the top of the Code View window. The Quick Tag Tools make it easy to select specific sections of code, edit code, and add new code. For example, when you click text inside a table cell, you see a Quick Tag Tool for the `<table>` tag and also the `<tr>` and `<td>` tags for the selected cell. To select the code that makes up any of these tags, simply click the Quick Tag Tool for the desired tag.



tip

The Quick Tag Tools are available in Code View, but after you make a change to the page while in Code View, the Quick Tag Tools disappear. Microsoft made that decision for performance reasons.

➔ For more information on using the Quick Tag Tools, see Chapter 8, “Using the Quick Tag Tools.”



Quick Tag Tools Not Visible

If the Quick Tag Tools are not visible when you're using Code View, it's likely that the insertion point is not inside code that can be selected or manipulated with Quick Tag Tools. For the Quick Tag Tools to be visible in Code View, the insertion point must be on or inside the `<body>` tag.

IntelliSense

Code View supports IntelliSense for HTML code, CSS code, PHP code, and scripting code. You can use IntelliSense for adding a new tag by placing the insertion point where you want the tag and pressing `<`, as shown in Figure 4.16.

Figure 4.16
IntelliSense aids in adding valid code when working in Code View.



When IntelliSense pops up, scroll to the desired entry and press Enter, or double-click it to insert it. IntelliSense is also available when editing existing tags and working with the Quick Tag Tools.

Context Menu

The context menu in Code View (see Figure 4.17) provides many useful tools for working with code.

One of the most useful features of Expression Web's Code View context menu is the ability to find matching curly braces in client-side script. There are also useful commands on the context menu to aid with moving pages created in other tools to Expression Web. However, other applications might not create XHTML-compliant code. By selecting Apply XML Formatting Rules from the context menu, you can easily make code XHTML-compliant without working through manual code edits.

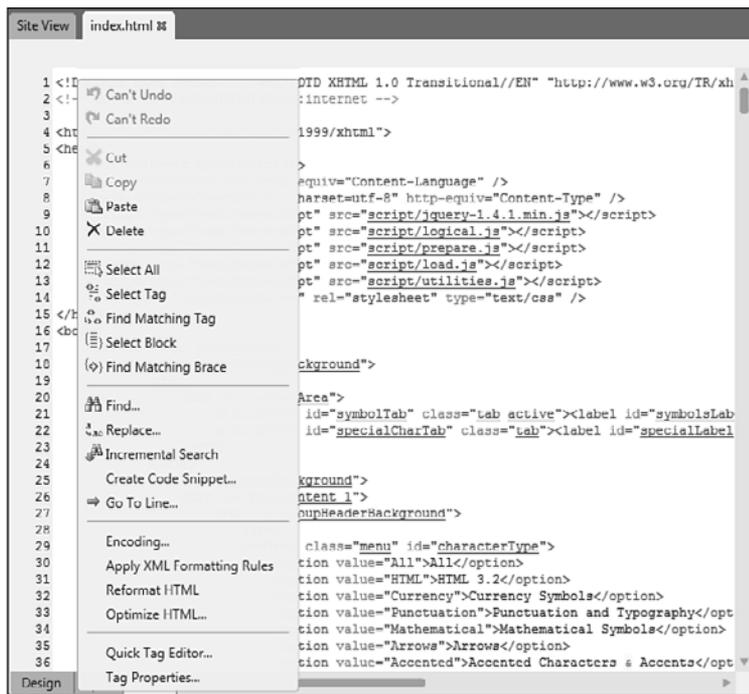


Figure 4.17
Expression Web offers many tools on the context menu in Code View to aid in adding and editing code.

Bookmarks

Another convenient feature available in Code View is the Bookmark. You can create a bookmark on a specific line of code by either selecting Edit, Code View, Toggle Bookmark or by pressing Ctrl+F2. A bookmark shows up as a blue rectangle in the code margin, as shown in Figure 4.18.

To move to the next bookmark in Code View, select Edit, Code View, Next Bookmark or press F2. To move to the previous bookmark, select Edit, Code View, Previous Bookmark or press Shift+F2. To remove all bookmarks, select Edit, Code View, Clear Bookmarks.

note

Bookmarks created in Code View are not the same as hyperlinked bookmarks. The bookmarks in Code View are simply reference points so that specific places in code can be more easily located.

Figure 4.18

Bookmarks can make finding a specific line of code fast and easy.

```

Site View | index.html* 88
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/2001/XMLSchema.dtd" [ ]>
2 <!-- saved from url=(0014)about:internet -->
3
4 <html xmlns="http://www.w3.org/1999/xhtml">
5 <head>
6   <title>Insert Symbol</title>
7   <meta content="en-us" http-equiv="Content-Language" />
8   <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
9   <script type="text/javascript" src="script/jquery-1.4.1.min.js"></script>
10  <script type="text/javascript" src="script/logical.js"></script>
11  <script type="text/javascript" src="script/prepare.js"></script>
12  <script type="text/javascript" src="script/load.js"></script>
13  <script type="text/javascript" src="script/utilities.js"></script>
14  <link href="styleLayout.css" rel="stylesheet" type="text/css" />

```

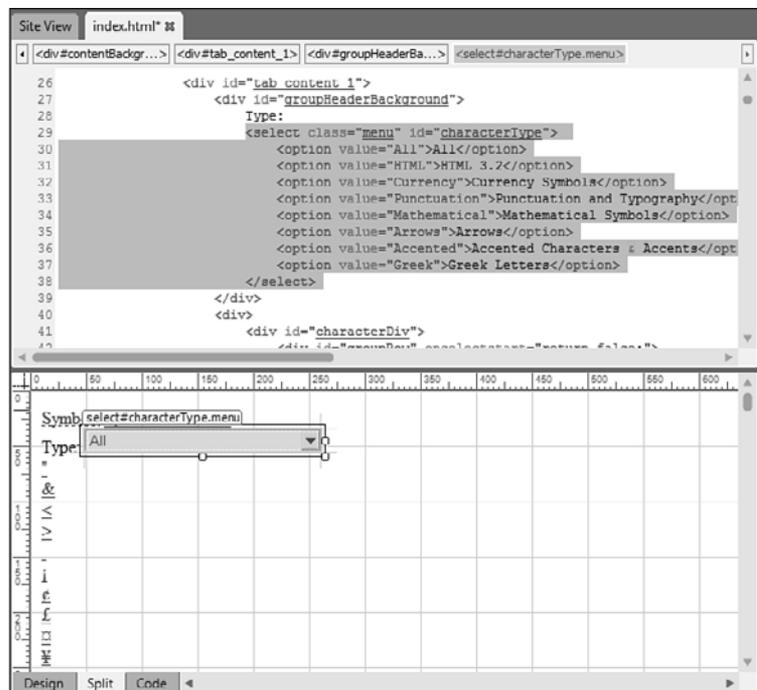
Working in Split View

Some edits are better suited to Design View, and others are better suited to Code View. But what about those times when it might be convenient to use Design View and Code View both at the same time? For those cases, Split View is the solution.

As shown in Figure 4.19, Split View splits the window in Expression Web into two horizontal panes. Code View is displayed on the top, and Design View is displayed on the bottom. You can drag the splitter between the two up and down to adjust the size of each view.

Figure 4.19

If you can't decide which view to use, Split View gives you the option of using both at the same time.



You cannot adjust the split so that the views are arranged vertically, and you cannot switch Code View to the bottom and Design View to the top. Still, Split View is a convenient method of editing when you need to see the code and design surface at the same time.

Taking Advantage of Page Views

Expression Web is a valuable tool for more than just creating sites and pages. For example, Code View is a great tool for reviewing code from someone else that isn't rendering correctly. Simply open the page in Code View and look for any highlighted code errors.

You can also use the Code and Design Views to learn more about how to create a particular layout. Simply select File, Open and enter the URL of the page you want to review. After the page is open, you can dissect it using Expression Web and find out exactly how the particular design features were created.

I'm a frequent visitor to many web design forums, and most of them include plenty of "How did they do that?" posts. By using Expression Web to dismantle pages, you can answer that question easily and add some new skills to your toolset.



When working in Split View, changes made in Code View aren't immediately visible in Design View. To see your changes in Code View reflected in Design View, you'll need to click inside of Design View. Microsoft designed Split View to work this way for performance reasons.

USING TABLES

The Origin of Tables

If you look back at the origins of HTML, you'll see a handful of particle physicists formulating a method for displaying scientific and educational data. Much of that information lent itself to being displayed in a tabular format, and out of necessity, HTML tables were born.

Tables are an excellent example of the true purpose of HTML. HTML was designed to be a layout language, and within a few years of the invention of HTML, tables began to play an important role in documents on the Web. As the Web progressed and HTML drifted into the hands of a more diverse collection of web developers, it began to be stretched to limits far outside its original intent.

Twenty years later, web developers are still using tables prolifically. Web development has drifted away from using tables for tabular data to using tables for complex layouts. You'd be hard-pressed to find a page these days that doesn't contain at least one HTML table, and those that don't are often the work of web developers who are proponents of a building movement to trend away from tables in favor of using Cascading Style Sheets (CSS) for layout.

➔ *For more information on using CSS, see Chapter 17, "Creating Style Sheets."*

I'm not going into detail about replacing tables with CSS. (I'll cover some of that in Chapter 17, "Creating Style Sheets.") I'm going to assume that if



note

I squeezed many years into the preceding history of HTML. HTML was actually created in 1989 by Tim Berners-Lee. The first tables appeared in what was called HTML+ some four years later.

you're reading this chapter, you want to learn how to use tables efficiently, so that's exactly what we're going to do starting right now.

The Makeup of an HTML Table Tag

HTML tables are made up of hierarchical tags that define one or more columns and rows. A table begins with an HTML `<table>` tag that defines the properties of the table. The `<table>` tag has several optional attributes that can be used to control the table's appearance.

It's important to understand the basics of how a table is put together in HTML code. It will help you understand the choices you can make in the Expression Web table dialogs.

The next few pages explain the tags that make up an HTML table. If you're already familiar with the details of how a table is created in HTML, you can skip ahead to the section "Tables in Expression Web" later in this chapter.

note

For details on the many attributes (including notations on which attributes are deprecated), see the W3C's recommendation at www.w3.org/TR/html401/struct/tables.html.

The align Attribute

Barring any other formatting that overrides it, a table is aligned on the left of a page by default. The `align` attribute allows you to align a table on the right or make it centered in the page.

The following `<table>` tag aligns a table in the center of the page:

```
<table align="center">
```

The `align` attribute is a deprecated attribute, meaning it has been replaced with a different method of specifying table alignment. The preferred method of aligning tables is to use CSS. However, when you specify alignment while inserting a table in Expression Web, it uses the deprecated `align` attribute.

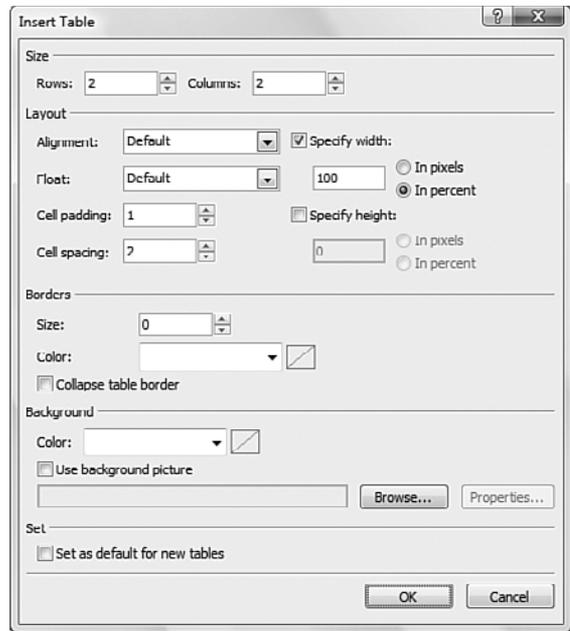
Listing 5.1 is the code that results from selecting Center in the Alignment drop-down of the Insert Table dialog as shown in Figure 5.1.

tip

If the default table settings you see in Figure 5.1 don't suit your needs, you can change them to whatever you want and then check the Set As Default For New Tables check box before you click OK. From then on, the Insert Table dialog defaults to your new settings.

Figure 5.1

The Insert Table dialog in Expression Web provides developers with one dialog for all the options necessary for various table layouts.



Deprecated HTML

Many attributes of the `<table>` element have been replaced with new formatting techniques, most of which involve the use of CSS. Such attributes are regarded as *deprecated* HTML. As you learn how to lay out pages using HTML code, you'll no doubt encounter plenty of techniques, attributes, and tags that are considered to be deprecated. Does that mean you should never use deprecated HTML? This question should really be lumped in with religion, politics, gun ownership, and other subjects that some people feel shouldn't be discussed. Fortunately for you, I'm not one of those people!

To answer this question, you need to think about your audience and which browser they'll be using when they view your site. These days, most deprecated HTML is being phased out in favor of CSS. When I wrote my first book on web design about ten years ago, many popular browsers didn't support CSS very well. Therefore, transitioning away from pure HTML in favor of CSS was often not a good choice.

Today, most people are using modern browsers that offer excellent support for CSS formatting and positioning, so the answer to the question at hand is less complicated. However, it's still not a simple one.

Continued...

I know plenty of people in the “Why upgrade if it still works?” crowd who are still using old Netscape browsers. If you abandon your HTML positioning techniques in favor of CSS, these people are going to experience quite a mess when they visit your site. Many web developers rant over such antiquated software, but the truth is that your site is not likely to greatly impact the masses. In fact, the design techniques of sites do not dictate the browser base. It’s the other way around.

If you decide to abandon a deprecated technique, make sure the replacement technique works well for the lowest-level browser you think your viewers will be using. If it doesn’t, keep the deprecated method and just smile at those who thumb their noses at you for not keeping up.

Listing 5.1 HTML Code for a Table Using the Center Alignment Attribute

```
1 <table "align="center" style="width: 100%">
2 <tr>
3 <td>&nbsp;</td>
4 <td>&nbsp;</td>
5 </tr>
6 <tr>
7 <td>&nbsp;</td>
8 <td>&nbsp;</td>
9 </tr>
10 </table>
```



tip

All table attributes can be adjusted using the Tag Properties and (if applied using CSS) CSS Properties panels.

Notice that Expression Web aligned the table using the `align` attribute. This method of aligning tables is deprecated and the preferred method is to use CSS.

➔ *For more information on formatting page elements with CSS, see Chapter 17, “Creating Style Sheets.”*

➔ *For more information on the Tag Properties and CSS Properties panels, see “Understanding the Tag Properties Panel,” p. 123, and “Using the CSS Properties Panel,” p. 307, respectively.*

Table Borders

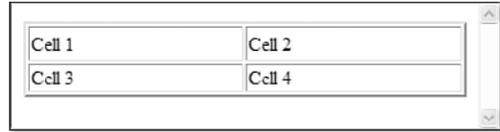
Expression Web uses the CSS Mode setting to determine how table borders are applied. By default, borders are applied using a CSS class, but you can override that using either the Style Application toolbar or by changing the Borders setting on the CSS tab of the Page Editor Options dialog.

➔ *For more information on the Page Editor Options dialog, see Chapter 11, “Configuring Page Editor Options.”*

A table border is affected not only by size, but also by the `cellspacing` attribute. In Figure 5.2, the table shown has a border size of 2 and a `cellspacing` value of 2. Notice that it appears to have a double border. The table in Figure 5.3, on the other hand, has a solid border without the double border appearance. The only difference between the two tables is that the table in Figure 5.3 has a `cellspacing` value of 0 instead of 2.

Figure 5.2

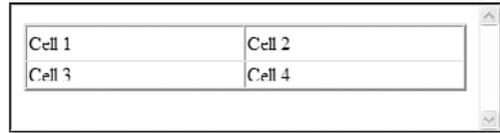
A table with a `cellspacing` value greater than 0 appears to have a double border due to the `cellspacing`. This table has both a `cellspacing` and a `border` setting of 2.



Cell 1	Cell 2
Cell 3	Cell 4

Figure 5.3

In comparison to the table in Figure 5.2, this table has a `cellspacing` of 0. All the other attributes are the same. Notice the double border is now gone.



Cell 1	Cell 2
Cell 3	Cell 4

The `cellpadding` Attribute

The `cellpadding` attribute is a pixel or percentage value that specifies the amount of space between the border of a cell and the contents of the cell.

A common use of `cellpadding` is to provide a buffer for content in separate cells. The table in Figure 5.4 shows a common use of tables in web design. An image is in one table cell, and the accompanying text is in another table cell. In this example, `cellpadding` is set to 0. Notice that the text in the second cell is touching the right edge of the image, making the presentation look unprofessional.

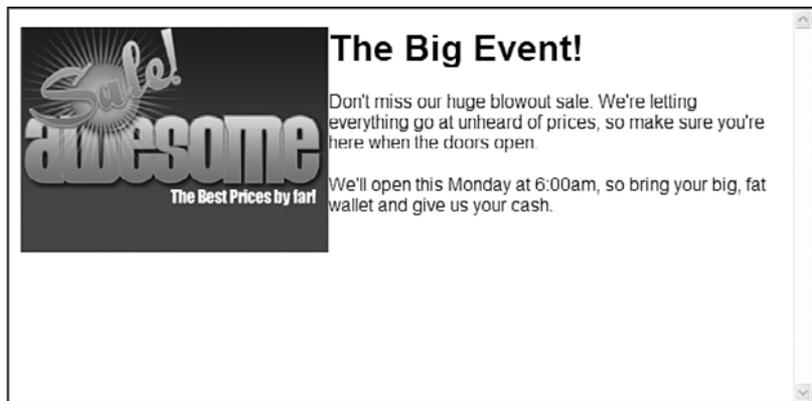


tip

If you'd prefer to use CSS to apply padding to your table cells, you can use the CSS `padding` property. For more information, see www.w3schools.com/css/css_padding.asp.

Figure 5.4

When text runs right up against an image, it looks unprofessional.



	<h2>The Big Event!</h2> <p>Don't miss our huge blowout sale. We're letting everything go at unheard of prices, so make sure you're here when the doors open.</p> <p>We'll open this Monday at 6:00am, so bring your big, fat wallet and give us your cash.</p>
---	--

In Figure 5.5, I've set the cellpadding to 4. All other settings remain the same. Notice that the application of cellpadding has transformed an unprofessional presentation into an attractive and effective one.

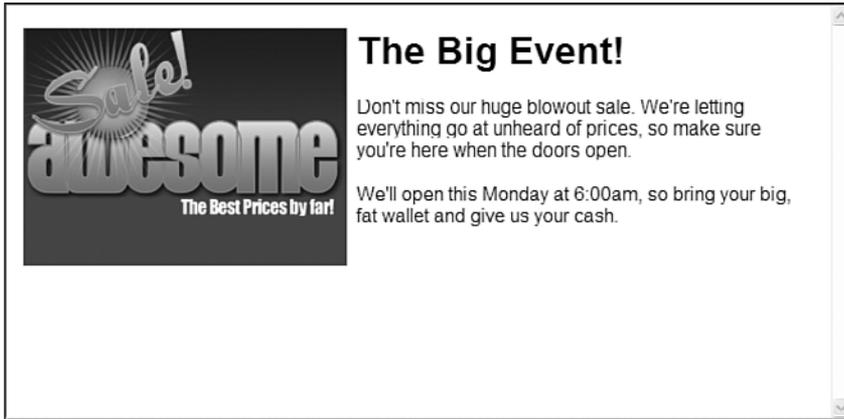


Figure 5.5

The application of cellpadding to the same table has transformed the presentation into one that's professional and effective.

The cellspacing Attribute

The cellspacing attribute controls the space between cells. It can be specified in either pixels or a percentage value.

The difference between cellspacing and cellpadding can sometimes be difficult to see. As shown in Figure 5.6, cellspacing adds space between the border of the table and the border of a table cell, as well as adding space between table cells. cellpadding, on the other hand, adds space between the inside border of a cell and the content occupying the cell.



tip

If you'd prefer to use CSS to apply spacing to your table cells, you can use the CSS border-spacing property. For more information, see www.w3schools.com/css/pr_tab_border-spacing.asp.

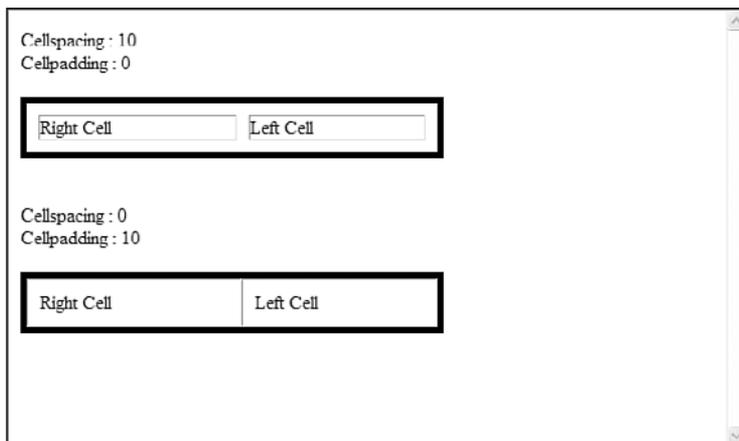


Figure 5.6

When you see cellspacing and cellpadding next to each other, the difference becomes clear.

The frame Attribute

The `frame` attribute provides for more granular control over how the border of a table is rendered. Without a border, the `frame` attribute has no effect.

The `frame` attribute can be configured in the Tag Properties panel and can have any one of the following values:

- `above`—The border across the horizontal top of the table appears in the browser. Other borders are ignored.
- `below`—The border across the horizontal bottom of the table appears in the browser. Other borders are ignored.
- `border`—A border is drawn around the table as a box. This setting produces the same output as the `box` setting.
- `box`—A border is drawn around the table as a box. This setting produces the same output as the `border` setting.
- `hsides`—A border is drawn on the horizontal sides of the table. In other words, a border will be visible across the top and bottom of the table.
- `lhs`—A border is drawn on the left side of the table. (`lhs` stands for left-hand side.)
- `rhs`—A border is drawn on the right side of the table. (`rhs` stands for right-hand side.)
- `void`—A border is not drawn around the table. Even if a border is explicitly set on the table tag, if the `frame` attribute is set to `void`, the border will not appear.
- `vsides`—A border is drawn on the vertical sides of the table. In other words, a border will be visible along the left and right sides of the table.



note

Although the `frame` attribute isn't deprecated, many web designers prefer to use the CSS border properties. See www.w3.org/TR/REC-CSS2/tables.html#borders for specific information.

Rows, Columns, and Cells

Now that you're familiar with the makeup of the `<table>` tag, it's time to dig into the other tags that make up a table.

An HTML table is really just a collection of rows and columns. Rows are defined by `<tr>` tags, and columns are defined by `<td>` tags. `<td>` tags cannot exist outside a `<tr>` tag. The code in Listing 5.2 defines a simple HTML table containing two rows and two columns.

Listing 5.2 Code for Table Containing Two Rows and Two Columns

```
1 <table>
2   <tr>
3     <td>Row 1, Column 1</td>
4     <td>Row 1, Column 2</td>
5   </tr>
6   <tr>
```

```
7         <td>Row 2, Column 1</td>
8         <td>Row 2, Column 2</td>
9     </tr>
10 </table>
```

The colspan and rowspan Attributes

Each `<td>` tag pair that you see in Listing 5.2 represents one cell in the table. In this example, each column and row contain the same number of cells. In some cases, you might want to combine two or more cells into one cell. When you combine cells in a row, you use the `colspan` attribute on the row. When you combine cells in a column, you use the `rowspan` attribute on the column.

Figure 5.7 demonstrates the effect of using the `colspan` and `rowspan` attributes on a table.

Rowspan = 2	

Colspan = 2	

Figure 5.7

Use the `colspan` attribute to combine cells in a row and the `rowspan` attribute to combine cells in a column.

For many people, dealing with the `colspan` and `rowspan` attributes in HTML code is difficult and confusing. Fortunately, Expression Web makes it easy, but it's still helpful to understand what's going on under the covers.

The code that follows defines the table shown in Figure 5.7, which has a `rowspan` of 2. Notice that the second `<td>` pair is missing from the second `<tr>` tag because the first column spans two rows.

```
<table>
  <tr>
    <td rowspan="2">Rowspan = 2</td>
    <td>&nbsp;</td>
  </tr>
  <tr>
    <td>&nbsp;</td>
  </tr>
</table>
```

The `colspan` attribute, on the other hand, merges two or more cells horizontally across two or more columns, as shown previously in Figure 5.7. The code that follows defines the table shown in Figure 5.7, which has a `colspan` of 2:

```
<table>
  <tr>
    <td colspan="2">Colspan = 2</td>
  </tr>
```

```
<tr>
  <td>&nbsp;</td>
  <td>&nbsp;</td>
</tr>
</table>
```

In this table, the second `<td>` pair is missing from the first row because the `colspan` attribute causes both cells in that row to be merged into one column.

Aligning Content in Cells

There are several alignment options for table cells. One way to align content in a table cell is by using the `align` and `valign` attributes. These attributes are actually deprecated in favor of CSS alignment, but many web developers still use them because of their widespread support by most browsers.

The `align` attribute aligns content horizontally in a table cell. Valid values are `left`, `right`, and `center`. The `valign` attribute aligns content vertically. Valid values are detailed in the following sections.

baseline

The `baseline` value aligns content on a common baseline. This option is best used when a cell contains a single line of text.

If you look at a line of lowercase letters, you'll see that the bottoms of most letters are aligned horizontally. (Letters such as `g`, `y`, and `j` are exceptions.) That line upon which the letters are aligned is the baseline. Setting the `valign` attribute to a value of `baseline` aligns the content in your cell on the baseline.

bottom

When the `valign` attribute is set to `bottom`, the cell's content is aligned at the bottom of the cell, as shown in Figure 5.8.

middle

A `valign` value of `middle` centers the cell's content vertically, as shown in Figure 5.9. This is the default setting. If no `valign` attribute is specified, cell content is centered vertically.

Figure 5.8

A `valign` value of `bottom` aligns a cell's content nicely along the bottom of the cell.



**Figure 5.9**

A `valign` value of `middle` is the default setting and centers the cell's content vertically.

top

The most common `valign` value is `top`. The table in Figure 5.10 shows the result of setting the `valign` value to `top`.

**Figure 5.10**

A `valign` value of `top` is the most common configuration for most sites.

This setting is often employed when using tables to align sliced images, as you'll see later in this chapter.

Now that you know the basics of how a table is built in HTML code, let's dig into how tables are created in Expression Web.

Tables in Expression Web

You have several ways you can work with tables in Expression Web, but you can break them all down into four major methods: the Insert Table dialog, the Table menu, the Tables toolbar, and the Tag Properties panel. Each method is suitable to specific types of tasks, but you might gravitate more toward one than the others. Even so, it's best to learn the strengths of each method so you can efficiently create and manage your tables.

➔ For more information on using the Tag Properties panel, see Chapter 7, "Editing Tag Properties."

Inserting Tables

The Insert Table dialog (refer to Figure 5.1) is displayed by selecting Table, Insert Table in Expression Web. If a table has already been inserted on your page, you can access the Table dialog by right-clicking the table and selecting Table Properties from the menu.

The Insert Table dialog is divided into several sections:

- **Size**—The Size section includes settings for the number of rows and columns.
- **Layout**—This section includes alignment settings, width and height settings, and cellpadding and cellspacing settings.
- **Borders**—This section includes settings for border size and border color and a check box that allows you to easily collapse the border.
- **Background**—This section includes a setting for background image and background color.
- **Set**—This section includes a check box to configure the default settings for future tables.

Customizing Tables

Expression Web can automatically format your table using the AutoFormat feature. Select Table, Modify, Table AutoFormat to access the Table AutoFormat dialog, shown in Figure 5.11. However, you'll likely want more control over the formatting, and for that, you'll need to use the options available on the Table menu.

The Table menu is actually made up of two menu systems. The Table menu on the main menu bar allows you to insert a new table. After a table has been added to your page, you can use either the Table menu on the main menu bar or the Table context menu (see Figure 5.12), accessed by right-clicking a table, table row, column, or cell.

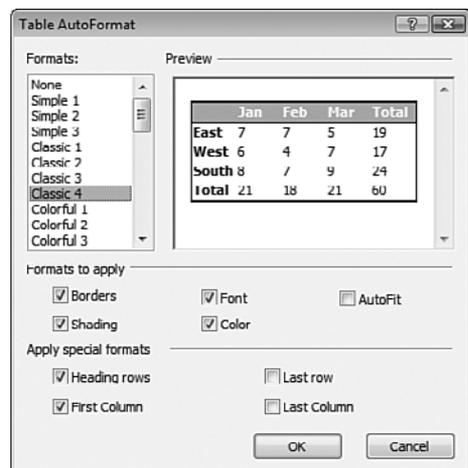


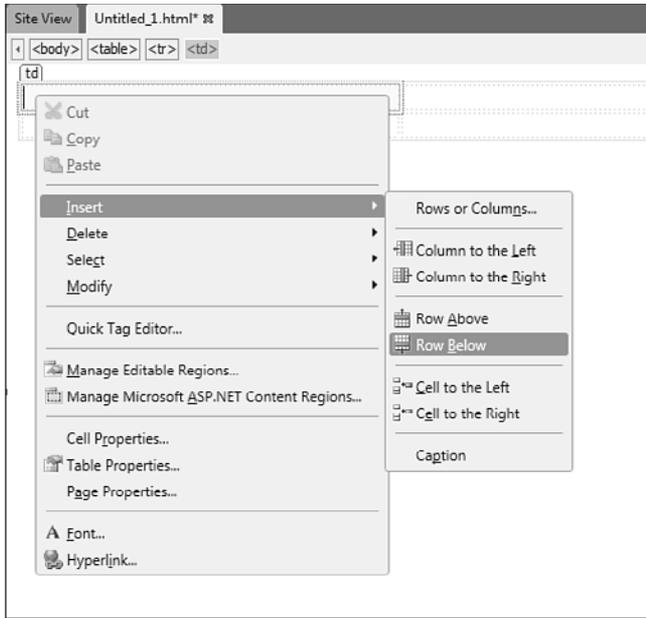
tip

You're not going to want to auto-format your table if you're using it to control layout. The Table AutoFormat feature is well-suited to displaying data in a table, but it's not a good choice for a table that defines layout.

Figure 5.11

The Table AutoFormat feature makes easy work of formatting a table. You have many attractive formats from which to choose.



**Figure 5.12**

The Table context menu is a convenient way to alter a table that has already been inserted into a page.

The Table context menu is essentially the same menu you find on the main toolbar. However, when working on a table in the designer, it's more convenient to be able to simply right-click the table to change it. As we go through the rest of this section, you can use the Table menu on the main menu bar and the Table context menu interchangeably.

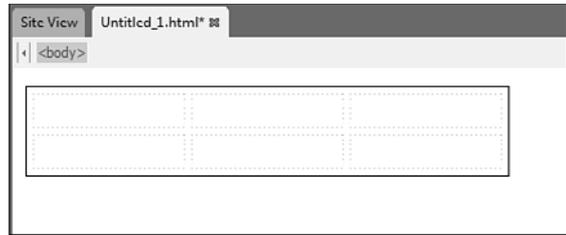
Let's work through some common table tasks using the Table menu. Begin by creating a new page. We'll use this page to experiment with tables throughout the rest of this chapter.

1. Add a new table to your page by selecting Table, Insert Table from the main menu.
2. Make sure that the number of rows is set to 2, the number of columns is set to 3, and the width is set to 400 pixels.
3. Set cellspacing and cellpadding to 5.
4. Set the border color to black, the border size to 1, and click OK to insert the table.

You should now have a table that looks like Figure 5.13.

Figure 5.13

Even a simple HTML table has some pretty complex code behind it, but Expression Web lets you easily create a table.



Selecting and Merging Cells

The table you created has two rows and three columns. Say that you want to add a graphic into the left column of this table and you want that graphic to take up the entire column. There are a couple of ways to accomplish that. You could slice the graphic into two pieces and insert one piece in each cell in the column. However, because the table has `cellspacing` and `cellpadding` values of 5, that won't result in the effect you want. Instead, as shown in Figure 5.14, you'll end up with space around the graphic that makes it painfully obvious that it's two graphic files.

In this case, a better solution is to merge the two cells in the leftmost column into a single cell. You can then insert the Expression Web graphic into that one cell without having to slice it and you'll end up with exactly what you want.

The first step in merging the cells in the leftmost column is to select both cells. There are many ways to do that:

- Use the Quick Tag Tools.
- Move the insertion point to the top of the column until the cursor becomes a downward-pointing arrow and click.
- Select the first cell by clicking it, and then drag the mouse down to select the entire column.
- With the insertion point in one of the cells, hold down the Ctrl key and click in the other cell.
- Select the HTML code in Code View.

➡ *For more information on using the Quick Tag Tools, see Chapter 8, "Using the Quick Tag Tools."*

➡ *For more information on editing HTML code in Code View, see Chapter 4, "Using Page Views."*

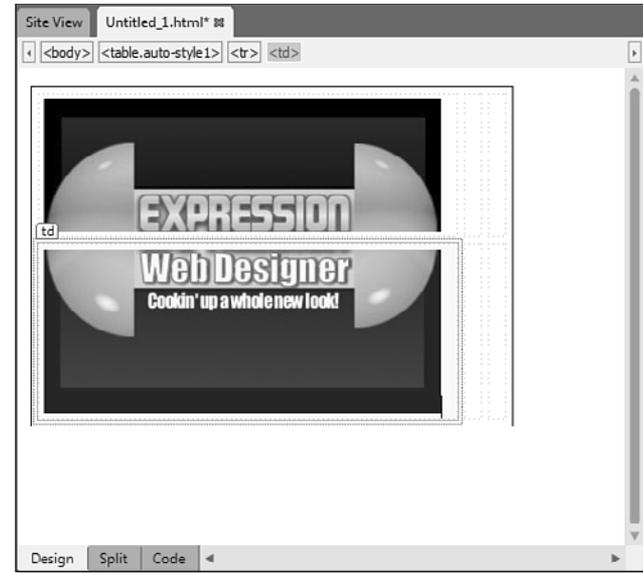
I personally prefer to move the insertion point to the top of a column and then click when I see a downward-pointing arrow. After you select the cells, right-click either of them and select **Modify, Merge Cells**. You should end up with a table similar to what's shown in Figure 5.15.

note

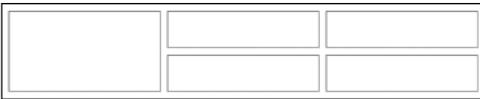
The graphics used in this chapter are located in the `ch5\files\Website\images` folder at informit.com/register.

tip

An alternative here would be to remove the `cellspacing` and `cellpadding`. However, for this example, we want to keep those settings as they are.

**Figure 5.14**

Slicing the Expression Web graphic for this table isn't a good idea. As you can see, you end up with whitespace that makes it look awful.

**Figure 5.15**

The leftmost column of the table has now been merged into a single cell. It's now a perfect spot for our Expression Web graphic.

Click inside the cell you just merged and insert a graphic. You can use the `ewd.jpg` graphic from the `ch5\files\Website\images` folder at informit.com/register. Your table should now look like Figure 5.16.

➔ *For more information on working with images in Expression Web, see Chapter 14, "Publishing a Site."*

Aligning Content in Cells

At this point, the alignment in your table looks fine. However, as new content is added, things will change.

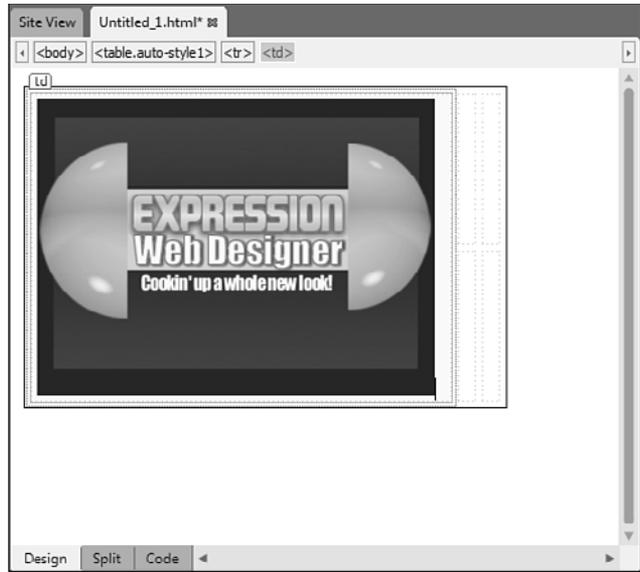
1. Add the `newimage.jpg` image to the top center cell.
2. Add the `killer.jpg` image to the bottom center cell.
3. Add some text to the upper-right and lower-right cells.

note

For the purpose of this example, I created a table that might be too wide for some screen resolutions. We'll talk more about designing for different resolutions in the "Designing for Multiple Resolutions Using Tables" section of this chapter.

Figure 5.16

The image now sits in a column of merged cells and looks perfect.



You should now have a table that looks similar to Figure 5.17. Things have obviously gone terribly wrong, and our table doesn't look good at all at this point.

Fortunately, we can fix this problem with a few tweaks to alignment and a small change in the size of the table.

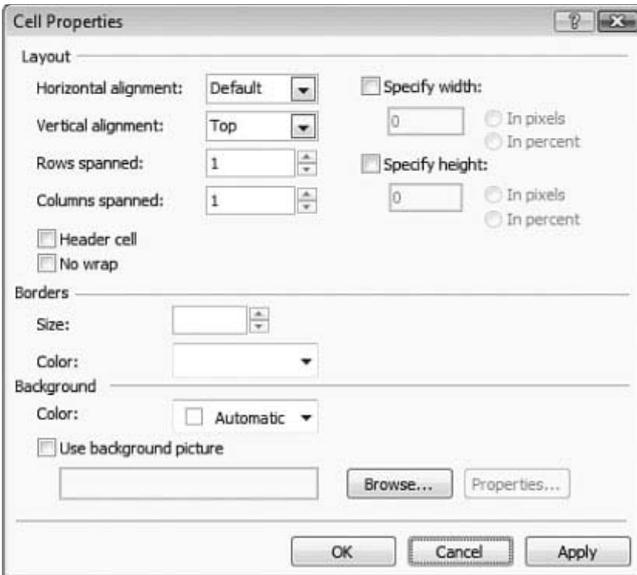
1. Right-click the large graphic in the leftmost cell, and select Cell Properties from the menu. The Cell Properties dialog appears.
2. Click the Vertical Alignment drop-down and select Top, as shown in Figure 5.18. Click OK.
3. Repeat steps 1 and 2 for both cells in the rightmost column.

**tip**

You can split cells into two or more cells by right-clicking in a cell and selecting Modify, Split Cells. You can then split the cell into two rows or two columns.

**Figure 5.17**

Our table has quickly gone from a nice-looking presentation to a terrible one. Fortunately, we can fix this pretty easily.

**Figure 5.18**

The Cell Properties dialog is a quick and easy way to adjust the properties of a single cell.

Your table should now look like Figure 5.19.

Figure 5.19

The table is starting to take shape now. It's still a bit narrow, but the alignment looks good.



Adjusting Table Width

The final step is to change the width of the table. Here are several ways to adjust the width of a table:

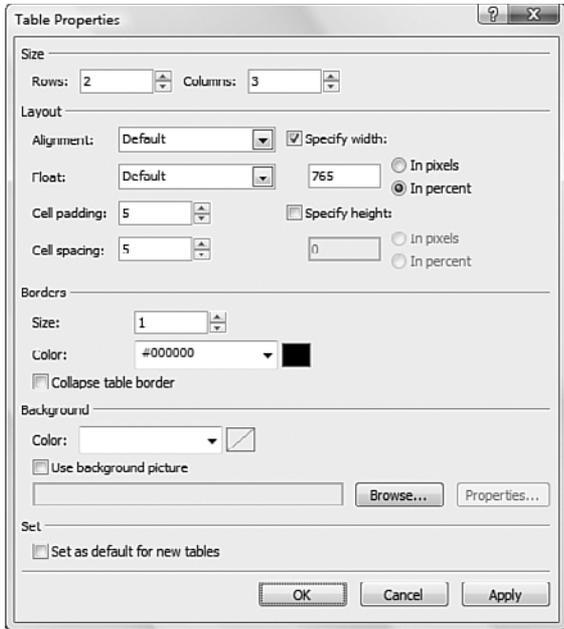
- Select the table and drag the right edge until the table is the desired width.
- Edit the HTML in Code View.
- Use the Quick Tag Tools.
- Select Table, Table Properties, Table on the main menu to access the Table Properties dialog, where you can modify the width.
- Right-click the table and select Table Properties from the menu to access the Table Properties dialog, where you can modify the width.

To alter the width of the table in the Table dialog, check the Specify Width check box and enter a new value for the width, as shown in Figure 5.20. I chose a width of 765 and selected the In Pixels radio button so the table will be 765 pixels wide. Your table should now look like Figure 5.21.



Table Is Too Wide

If you insert a table and the right edge is way off the screen, make sure you aren't setting a width of more than 100%. This is commonly caused by moving from a pixel setting to a percentage setting. For example, if you have a table that is set to 600 pixels wide and you later decide to change it to a percentage, you might accidentally just select the Percent radio button in the Table dialog. Doing so would set the table to 600%, which would cause it to run clear off the right side of the page.

**Figure 5.20**

Increase the table's width to make the text more readable.

**Figure 5.21**

The completed table presents the information in a pleasing and well-formatted way.

How Expression Web Formats Tables

As already discussed, many HTML techniques have been deprecated in favor of updated methods—most commonly using CSS. When you modify table properties using either the Table dialog or Table menu, Expression Web applies your settings using nondeprecated methods, in most cases. It does so in a couple of ways: inline styles and embedded style sheets.

The deprecated method of changing a table's width is to use the `width` attribute on the `<table>` tag, as shown in the following example:

```
<table width="500">
```

The preferred method of modifying a table's width is to use CSS styles, and this is the method Expression Web uses. The following code demonstrates how to specify table width using an inline style, the method Expression Web uses to specify table width:

```
<table style="width: 500px">
```

Expression Web can also use embedded style sheets to format a table, depending on how you have configured style application settings. For example, when you change the alignment of a table to `right`, Expression Web might create an embedded style sheet that defines a CSS class. (I realize that at this point, you might not understand what all this means. We cover CSS in-depth in Chapter 17, “Creating Style Sheets” and Chapter 18, “Managing CSS Styles.”)

The following code is an example of an embedded style sheet that sets the alignment of a table to `right`:

```
<style>
    .auto-style1{
        text-align: right;
    }
</style>
```

When Expression Web uses embedded style sheets to format a table, it wraps your table inside an HTML `<div>` tag and assigns that `<div>` tag the CSS class created previously like so:

```
<div class="style1">
    <table ...>
        ...
    </table>
</div>
```

Those of you who have used Adobe Dreamweaver should be familiar with this technique because it's the same way Dreamweaver works. I appreciate Expression Web using this updated method of applying styles to my table, but I don't really like the naming convention it uses because it's nondescriptive. Instead, I usually create my own styles and assign them as needed. You'll learn how to do that when we cover CSS in subsequent chapters.

Adding and Deleting Rows and Columns

In the example we just worked through, we created a table that had just the right number of rows and columns for our needs. In almost all cases in the real world, you need to adjust the number of rows and columns after a table has already been inserted. Fortunately, Expression Web provides numerous ways for doing this.

It goes without saying that you can open the Table Properties dialog we worked with previously and change the number of rows or columns. Adding a new row using this method always adds the row to the bottom of the table. Adding a new column using this method always adds the new column to the right side of the table. This might not be what you want, so if you want more control, you need to use one of the other methods that Expression Web provides:

1. Right-click the leftmost cell in the table; and select Insert, Rows and Columns.
2. In the Insert Rows or Columns dialog, select Rows, making sure that the number of rows is set to 1 and that Below Selection is selected, as shown in Figure 5.22.
3. Click OK to insert a new row to the bottom of the table.



tip

There are other ways to add a new row or column. The Table menu provides plenty of options for adding and deleting rows and columns. You also can add a new row by placing the insertion point in the rightmost bottom cell and pressing the Tab key on your keyboard.

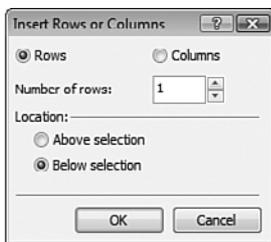


Figure 5.22

The Insert Rows or Columns dialog gives you full control over where rows or columns are added to your table.

Designing for Multiple Resolutions Using Tables

One of the most frustrating endeavors for web developers is ensuring that a site looks great in all screen resolutions. If you design for a resolution of 1024×768 and someone with a resolution of 800×600 browses your page, it won't look the same. There are a few different ways you can alleviate this problem.

The first way is the easiest, and that is to set your main content table to a width of around 95%. Why 95% instead of 100%? It will look better if you allow for a small buffer space around the window's scrollbars, and so on.

The second way is to specify a pixel width for those columns that require it and then have one column that is configured to fill the rest of the available screen space. For example, suppose you have

a table with a menu along the left border and some ads along the right border. You would want the left and right borders to be a fixed pixel size so they match the size of the menu and the ads. The center column can be configured to fill the rest of the screen area by setting it to 100%.

The last method is to create your table so it fits nicely in the lowest resolution you believe people will use when visiting your site. If you choose this route, you'll need to ensure that you account for the fact that you don't really have 100% of the width of the screen to work with. For example, my wife runs her computer at a resolution of 1024×768, but she also always keeps the Favorites panel open in her web browser. Therefore, she has about 800 pixels in width that is visible to her even though her screen resolution is quite a bit higher.

Which method should you choose? The first two are the best choices. If you go with a percentage value, you can always be sure that most people will be able to view all your content without unnecessary scrolling.

This page intentionally left blank

USING FRAMES

Using Frames in Sites

HTML frames have been around for quite a while. A *frames page* is a page that is segmented into two or more sections called *frames*. Each frame contains a separate page and can be configured with scrollbars and a frame border that can be either fixed or resizable.

If you've spent any amount of time discussing web design with other web designers, you've no doubt encountered what I call the "never use it" mentality. Web designers with this mentality maintain a list of design techniques they claim should never be used under any circumstance. The use of frames is a technique that appears on many of those lists, but as with any technique, frames do have some effective uses. The trick is in knowing when to use them.

When to Use Frames

One of the most common uses of frames is in separating navigation elements from other page content, especially when the navigation components require scrolling. The benefit of using frames in such setups is that you can scroll the page containing the navigation elements without scrolling the page containing the main content.

Modern sites will opt for using CSS for these scenarios because the use of frames can cause accessibility issues and problems with search engine indexing. However, not all HTML pages are designed to be used on the Internet. For example, some software discs contain HTML-based documentation that is designed to be viewed locally. The use of frames in this type of situation is fine. It's also not uncommon to see frames used when creating HTML help documentation.



note

If you are designing your site to be HTML5-compliant, avoid frames. Frames are no longer allowed in HTML5. However, the use of inline frames is still valid. I'll cover inline frames later in this chapter.

When Not to Use Frames

It's generally a good idea to avoid the use of frames for sites that will be hosted on the Internet. The following are some of the drawbacks of frames:

- **Search engines often index pages that are part of a frameset**—When that happens, search results can link to pages that don't include your navigation or other important page elements.
- **They might not render the same way on all computers**—Frames include window elements such as scrollbars that aren't always the same size in every browser, thereby causing rendering problems in some cases.
- **When a frames page links to another frames page, you end up with nested frames**—This can be extremely frustrating for users of your site.

You must also take into account that the text size in browsers can be (and often is) adjusted by users with low vision and for other reasons. If your frames are designed for text of a certain size and a site visitor alters the text size in the browser, it can modify your site's layout in unpredictable ways.

Now that you know when it's okay to use frames and when you should probably opt for another technique, let's talk about how to create frames pages in Expression Web.

Creating Frames Pages

Frames pages (as opposed to inline frames, which I discuss later) are actually made up of multiple pages. At the top level is the frameset page. The frameset page doesn't actually contain any visible content. Instead, it contains HTML code that defines two or more frames and their layout.

To create a new frameset in Expression Web, do the following:

1. Select File, New, Page.
2. Select Frames Pages from the list of page types.
3. Select the frame layout of your choice from the list in the New dialog, as shown in Figure 6.1. For this example, I chose the Contents layout.



tip

There is a movement under way (and has been for quite some time) to replace frames with CSS. In fact, doing so is possible, and it works well in modern browsers. You can read about a method of doing this at www.webreference.com/html/tutorial24/.



tip

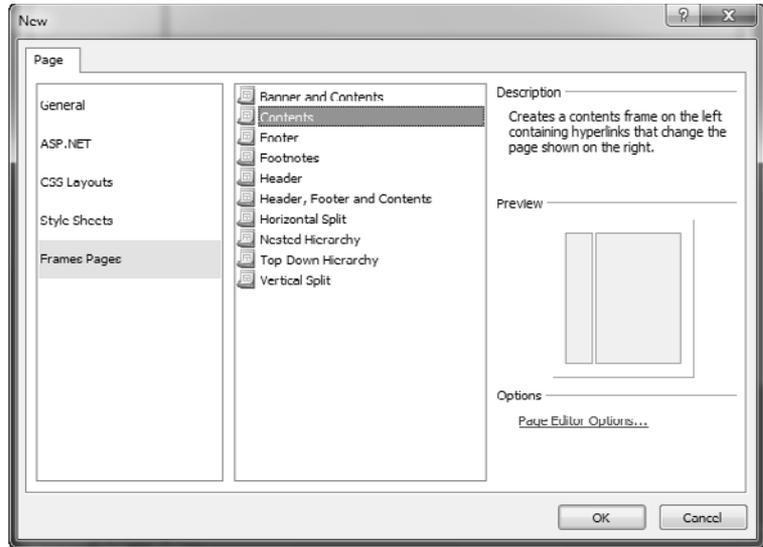
Expression Web comes with a collection of CSS layouts that you can use in lieu of frames. You'll find them in the CSS Layouts section of the New dialog when creating a new page.



note

It's technically possible to have a frameset that defines only one frame. However, there's really no point to this setup.

Figure 6.1
Expression Web gives you many frame layout choices.



After Expression Web creates the frameset, you have a choice of either creating a new page for each frame or choosing an existing page to appear within the frame. To create a new page, click the New Page button shown in Figure 6.2.

Figure 6.2
When you create a frameset, you can choose to create a new page for a frame or use an existing page.



To use an existing page, click the Set Initial Page button to display the Insert Hyperlink dialog, where you can choose a page. You can also enter a URL to a location not in your site, as shown in Figure 6.3. After you click OK, the page or URL you specified is loaded into the frame.

caution

If your frame displays content from a site other than your own, you will be displaying web content not under your own control. Keep this in mind when designing your site.

I've almost never encountered a situation where showing content from another site in a frame was a good idea.

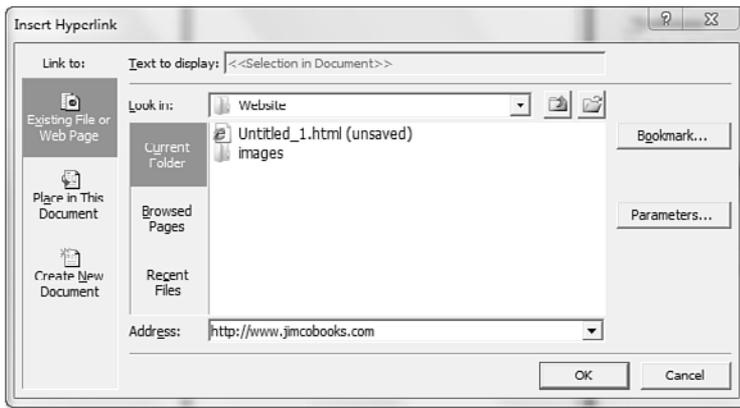


Figure 6.3

In addition to choosing files in your site for frame content, you can enter a URL.

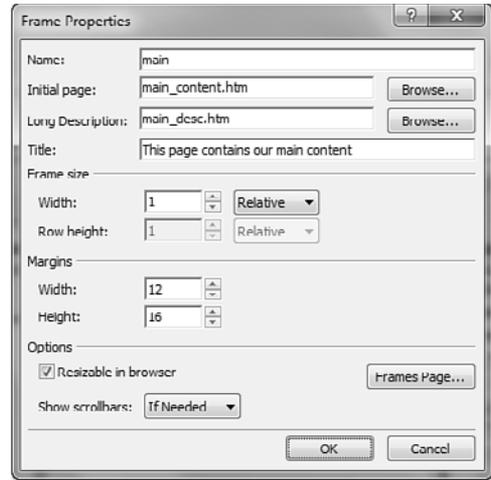
Configuring Frames

To configure frames in a frameset, right-click the frame and select Frame Properties from the menu. Alternatively, you can select the frame and select Format, Frames, Frame Properties. Either method displays the Frame Properties dialog shown in Figure 6.4.

The following properties are configurable in the Frame Properties dialog:

- **Name**—The name of the frame can be used when you need to direct hyperlinks to open in that frame. Links with a `target` attribute of "main" open in the frame being configured (refer to the example shown in Figure 6.4).
- **Initial Page**—The initial page is the URL displayed in the frame when it is first loaded.
- **Long Description**—The long description specifies the URL of a file that contains a long description of what appears in the frame. This is used by screen readers.
- **Title**—The title is a short description of the frame's contents. It is also used by screen readers.

Figure 6.4
All frame properties are easily configured using the Frame Properties dialog.



- **Frame Size Width and Row Height**—Specifies the width and height of the frame. This can be relative to other frames in the frameset, a percentage of available window space, or absolute pixel dimensions.
- **Margin Width and Height**—Specifies the margins for the frame.
- **Resizable in Browser**—When checked, the frame can be resized by dragging the border in the browser.
- **Show Scrollbars**—Configures whether scrollbars are displayed. Can be set to the following: If Needed, Never, or Always.

By clicking the Frames Page button shown previously in Figure 6.4, you can access properties for the frameset page, as shown in Figure 6.5. Adjusting the Frame Spacing value changes the width of the border between frames. To remove the border completely, remove the check from the Show Borders check box.

Splitting Frames

To split a frame into two frames, select the frame and then select Format, Frames, Split Frame. Expression Web displays the Split Frame dialog shown in Figure 6.6, where you can choose to split the frame into columns or rows. When a frame is split into two frames, each of the two frames will be 50 percent of the width or height of the original frame.



caution

If you set Show Scrollbars to Never and the frame's content extends beyond the borders of the frame, any content outside the frame will be inaccessible by users of your site.



caution

The Frame Spacing setting adds a framespacing attribute to the <frameset> tag in your page. The framespacing attribute is only recognized by Internet Explorer and is a nonstandard attribute. It will be ignored by other browsers.

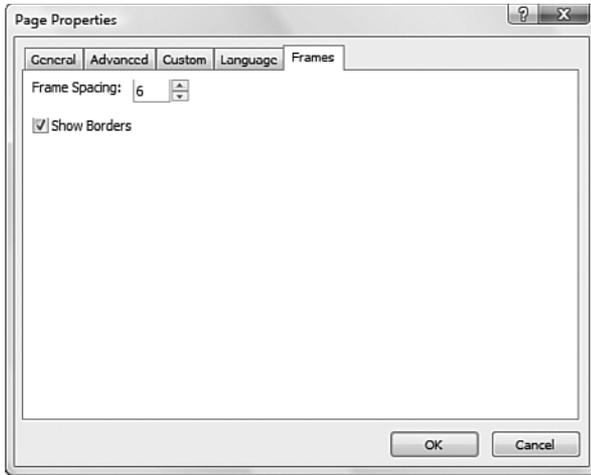


Figure 6.5
Properties for the frameset page are configured in the Page Properties dialog.

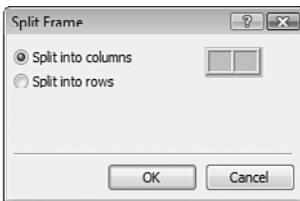


Figure 6.6
A frame can be split into two frames side-by-side or stacked one on top of the other.

➔ For more information on adding tables to a page, see *“Tables in Expression Web,” p. 96.*

Deleting Frames

To delete a frame, select the frame and then select Format, Frames, Delete Frame. Expression Web removes the frame immediately without confirmation. If you remove a frame inadvertently, you can select Edit, Undo Delete Frame or press Ctrl+Z to restore it.

Creating Alternative Content

Frames are a fairly old technology (they were introduced in Netscape 2.0) and are supported in all modern browsers. However, frames are one of the few HTML techniques that have no graceful fallback. When browsers encounter a tag they don't understand, they simply ignore the tag. In most

note

If you are considering splitting a frame, consider whether it might not be more appropriate to simply add a table to the existing page instead. Splitting frames can create a user interface for your site that is awkward to navigate.

cases, that doesn't present much of a problem, but in the case of frames, the page containing the frame tags doesn't contain any content. It only contains tags that define the frames. For that reason, when a browser that doesn't understand frames loads that page, it simply won't display anything at all.

Fortunately, there is a way to specify alternative content for browsers that don't understand frames. By using the No Frames view, as shown in Figure 6.7, you can specify content to be displayed for browsers that don't support frames.

The default message provided by Expression Web (shown in Figure 6.7) isn't very useful, so you'll likely want to add your own content. Therein lies one of the problems with frames. If you're going to create content in the No Frames view that emulates your frames content, you may as well carefully evaluate whether the frameless content is suitable for everyone as a substitute for frames.

note

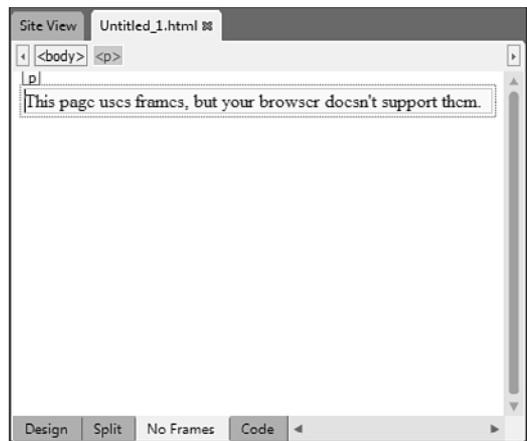
Keep in mind that your site will rarely (if ever) be visited by someone using a browser that doesn't support frames.

tip

I've alluded to this before, but it's worth repeating. As a general rule of thumb, when in doubt, don't use frames.

Figure 6.7

The No Frames view allows you to enter content to be displayed when a browser doesn't support frames.



Targeting Frames

Expression Web allows you to easily configure a hyperlink to target a particular frame. To cause a hyperlinked URL to display in a specific frame, follow these steps:

1. Insert a hyperlink.
2. In the Insert Hyperlink dialog, click Target Frame, as shown in Figure 6.8.
3. In the Target Frame dialog, enter the name of the frame you want to target. You can also simply click the frame you want to target in the Current Frames Page representation, as shown in Figure 6.9.

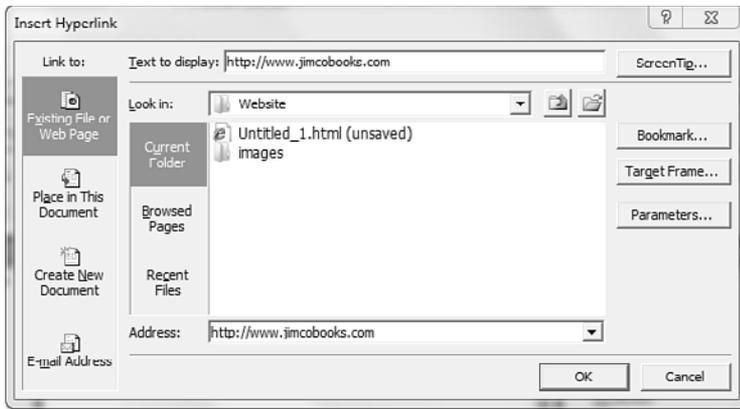


Figure 6.8
The Target Frame button in the Insert Hyperlink dialog makes targeting a specific frame easy.

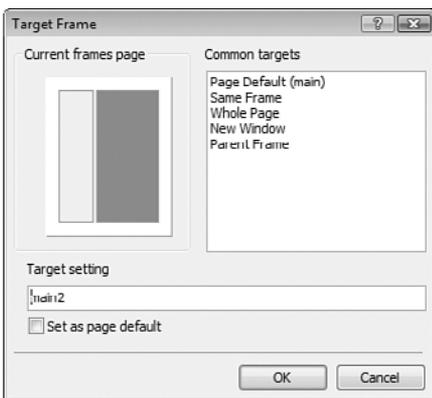


Figure 6.9
Expression Web provides a visual representation of the current frames page. Clicking a frame causes the current hyperlink to target that frame.



Targeted Hyperlink Opens New Window

If you've created a hyperlink that uses the target attribute to target one of your frames, but the page opens in a new window when you click the link, it's likely that you have inadvertently misspelled the frame name in the target attribute. When an unknown frame name is specified for a target, most browsers simply open a new window to display the link.

4. Click OK to dismiss the Target Frame dialog.
5. Enter a URL for the hyperlink or select a page from the Insert Hyperlink dialog and click OK to insert the hyperlink.

Editing Files in Frames Pages

When you right-click a frame and select Open Page in New Window, the page loaded into that frame opens in a new window. This is a convenient way to edit a page because it gives you access to the entire design surface in Expression Web.

When you edit a page within a frame by opening a frameset, it is often awkward to manipulate page elements. By opening the page in a new window, you can more easily edit the page's contents. Just keep in mind that the frame that will hold the page when it's browsed will contain less screen real estate and size your elements accordingly.

Adding and Configuring Inline Frames

So far in this chapter we covered using frames inside a frameset page. Using this method, a single page is broken up into multiple frames. Another type of frame available to you in Expression Web is the inline frame. An *inline* frame is a frame that can be inserted into an existing page and can be any size you want.

➔ For more information on layers, see Chapter 23, "Using Layers."

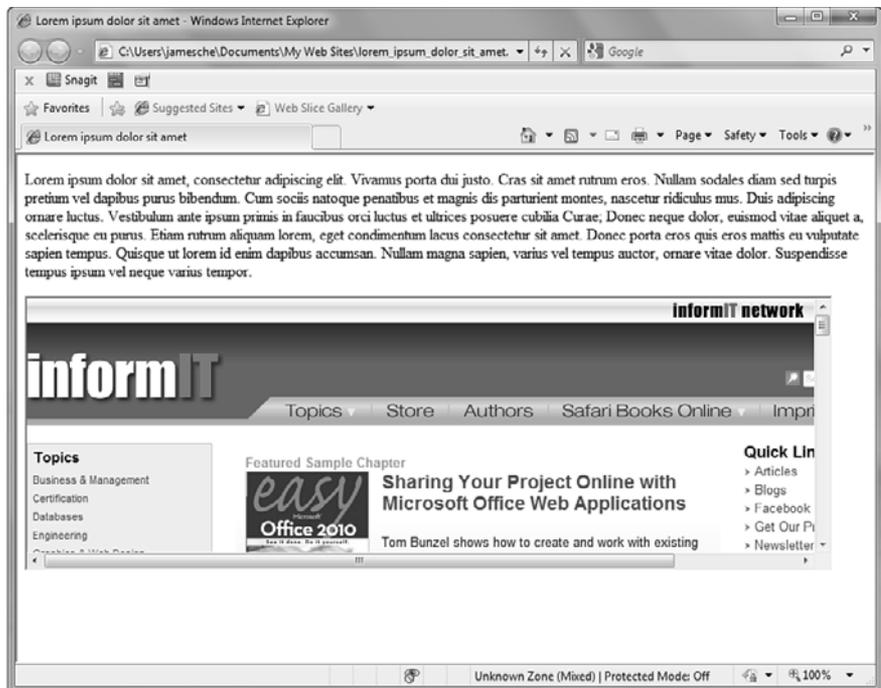
Figure 6.10 shows an inline frame into which I have loaded the InformIT site. Notice the scrollbars that let you scroll around the entire page displayed by the inline frame.



note

Inline frames are the only frames allowed in an HTML5 page.

Figure 6.10
An inline frame allows you to have a page within a page. In this case, the InformIT site is displayed inside an inline frame.



To insert an inline frame, place the insertion point where you want the inline frame and double-click on the Inline Frame control in the Toolbox. When you do, you are given the option of specifying an initial page or creating a new page for the frame's content, as shown in Figure 6.11.

After you've added content in an inline frame, you can control that inline frame's properties by right-clicking the inline frame and selecting Inline Frame Properties from the menu. Doing so displays the Inline Frame Properties dialog shown in Figure 6.12.

 **note**

In most cases, you will want to display your own content inside an inline frame. The inline frame shown in Figure 6.10 displays an external site so you can more clearly see exactly what the inline frame does.

**Figure 6.11**

Selecting a page for display in an inline frame is performed just as it was with a frameset. You can specify an initial page or create a new page.

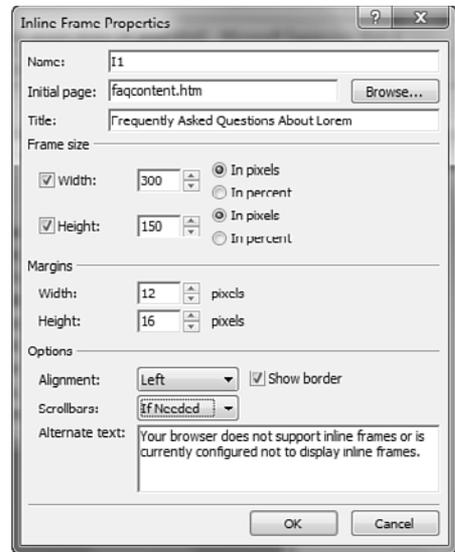
The options available in the Inline Frame Properties dialog are essentially the same as those available in the Frame Properties dialog previously shown in Figure 6.4. One notable exception is that the Inline Frame Properties dialog provides an Alternate Text text box for entering content to be displayed for users who cannot see inline frames.

As with frames pages, you can target an inline frame with hyperlinks by specifying the name of the inline frame in the target attribute of the hyperlink. This is a common way of updating a portion of a page when a link is clicked elsewhere on the page.

 **tip**

You can enter HTML code into the Alternate Text text box in the Inline Frame Properties dialog. By using HTML as alternate text, you can display alternate content that shares a look and feel with the rest of your site.

Figure 6.12
Inline frames are configured using the Inline Frame Properties dialog.



Tips for Frames

As I mentioned at the beginning of this chapter, plenty of designers will tell you that you should never use frames. I'm not one of them. However, you should carefully consider the possibility of using tables or other design elements such as CSS. If you decide that frames are the right choice for you, there are some general tips you should follow so that they don't become a point of frustration for users of your site.

note

If you want some guidelines on when to use frames and when you might want to consider other options, see “Using Frames in Sites” earlier in this chapter.

Frame Borders

As you've seen in this chapter, you can configure a frames page so there are no borders on the frames. Before you choose to go borderless, you should carefully consider your site's design. If you have user interface elements such as graphics that distinguish the frame from the rest of your page, removing the borders might be a suitable option.

Examine the page shown in Figure 6.13. The navigation links at the left side are in one frame, and the main content is in another. However, there's no way to tell that it's a frames page by looking at it. If I scrolled the content page, the navigation would remain in place. That might be what you want, but it also might be confusing to some users. Placing a border on the frame provides a visual indicator that the two parts of the page are separate.

Resizable Frames

When adding frames for navigation elements, it's often tempting to configure the frames so they cannot be resized. Depending on the site's design, that might or might not be a good idea.

If the content in the frame consists of text, it can often be a bad idea not to allow users to resize the frame. Some users of your site might have adjusted the size of the text in their browsers. In some cases, this can cause your text to overflow the frame.

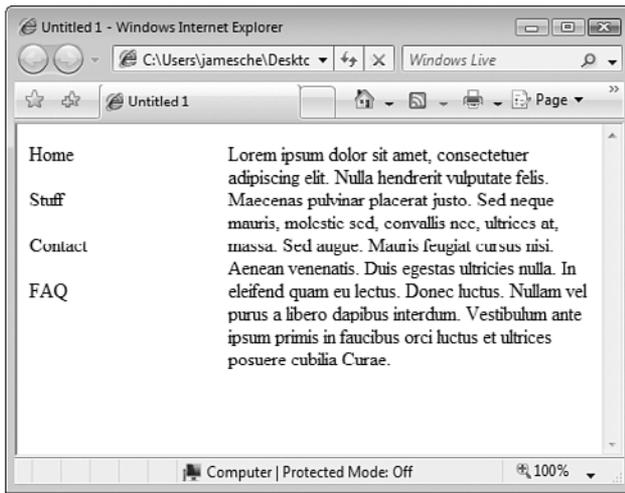


Figure 6.13

Borderless frames can be confusing to users because there is no visual indicator that the frames are separate.

The page shown in Figure 6.14 is an example of bad design with frames. Notice that the Contact link is not fully visible because I have adjusted the text size in my browser.

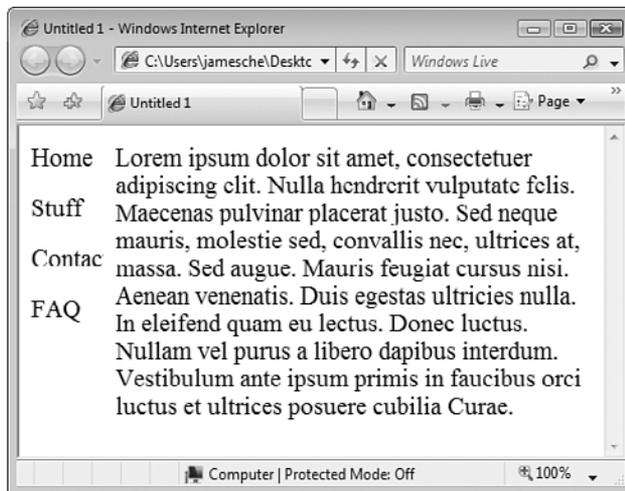


Figure 6.14

Text inside a frame can overflow the frame if a user adjusts the text size in his browser.

When using frames in your site, be sure you test carefully. Adjust the size of the text in your browser and test your site again. The goal is to make your site look the way you intend for all site visitors. With careful testing and design, you can easily reach that goal.

Breaking Out of a Frameset

If you do use frames in your site, you run the risk of your frames page being displayed inside a nested frameset. Figure 6.15 shows an example of a nested frameset caused by a link inside a frameset linking to a frames page.

Figure 6.15
A nested frameset caused by a frames page linking to another frames page. Notice that there are two navigation frames, one belonging to the previous frames page.



The solution to this problem is to include a JavaScript snippet to force the frameset page to remove any nested framesets when it gets loaded. Simply include the following snippet inside your `<head>` section in the frameset page:

```
<script language="javascript" type="text/javascript">  
if (window != top) top.location.href = location.href;  
</script>
```

This page intentionally left blank

EDITING TAG PROPERTIES

An Introduction to Tag Properties

One of the greatest things about Expression Web is that it provides many tools that make it easy to dig into the makeup of a page. The Tag Properties panel is one of those tools. It is a powerful way of examining the makeup of specific HTML tags. Not only that, but you can also easily set tag properties using the Tag Properties panel.

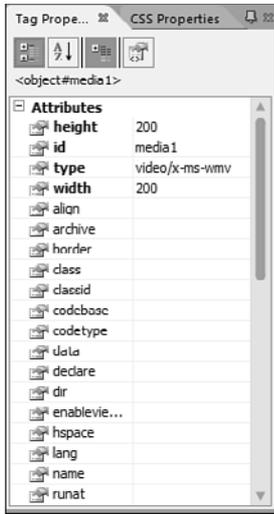
Understanding the Tag Properties Panel

When an HTML element is selected in either Design View or Code View, properties of the selected tag are displayed in the Tag Properties panel, as shown in Figure 7.1.



note

A comprehensive discussion of HTML attributes is beyond the scope of this book. For a complete reference of HTML attributes, read *Special Edition Using HTML and XHTML*, available from Que Publishing.

**Figure 7.1**

The Tag Properties panel provides a convenient way to dig into the properties of any HTML tag.



Tag Properties Panel Not Visible

If the Tag Properties panel isn't visible, select Panels, Tag Properties from the main menu to enable the panel.

By default, the Tag Properties panel shows you properties grouped into three categories: attributes, events, and miscellaneous. (Any attribute that is not standards-compliant will be categorized as miscellaneous.) You can collapse any category by clicking the minus sign next to the category name, as shown in Figure 7.2. To expand a collapsed category, click the plus sign next to the category name.



tip

The Tag Properties panel is context-sensitive. The properties that are visible in it depend on the type of element selected on your page.



tip

The events displayed in the Tag Properties panel are technically attributes as well. However, Expression Web divides each into its own category to make it easier to write code to handle a particular event.

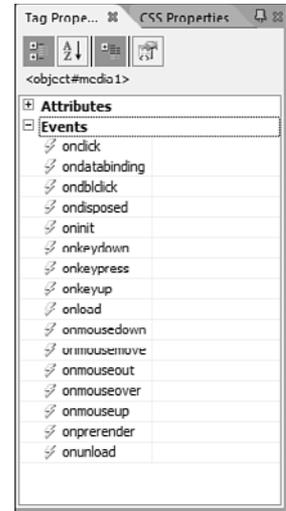


tip

You can also double-click a category name to collapse and expand that category.

Figure 7.2

Categories can be collapsed and expanded using the - and + button, respectively, located next to the category name.



Attributes and Events

Expression Web divides tag properties into attributes and events. Although both of these categories are technically HTML attributes, there is a difference between the two. First, let me briefly explain what an attribute is.

Attributes are parts of an HTML tag that are in the following format:

```
<attribute_name>="<attribute_value>"
```

For example, an HTML hyperlink is defined using the <a> tag. The <a> tag has several attributes as well that define the behavior of the hyperlink. The href attribute specifies the resource to which the hyperlink links. For example, the following tag creates a hyperlink that links to the Microsoft site:

```
<a href="http://www.microsoft.com">Microsoft </a>
```

An HTML tag can have many attributes, some of which are required and some of which are optional. There are many other possible attributes for the <a> tag; only one is shown in the previous code.

In addition to attributes that control the appearance of HTML elements, a special kind of attribute controls what happens during certain events. All pages displayed in a browser trigger certain events. For example, when a page loads, the Load event is triggered. When a user clicks an element, the Click event is triggered. If you don't specify a particular action to take when an event is triggered, the event usually goes unnoticed.

Continued...

If you want code to run when an event occurs, you specify that code using the event handler attribute. For example, to run code when a page loads, use the `onload` attribute. To run code when an element is clicked, use the `onclick` attribute. Specifying code to run when an event occurs is referred to as *hooking* the event. The following code runs a function called `setUpPage` when the page loads:

```
<body onload="setUpPage();">
```

As shown in Figure 7.3, the following buttons are available at the top of the Tag Properties panel:

- **Show Categorized List**—When this button is clicked, the properties displayed in the panel will be categorized as attributes and events.
- **Show Alphabetized List**—When this button is clicked, the properties in the panel will be displayed alphabetically.
- **Show Set Properties on Top**—When this button is clicked, properties for which a value has been specified will always appear at the top of the list.
- **Show Tag Properties**—When this button is clicked, the Properties dialog that corresponds to the selected tag will be displayed.



tip

If you want all the set properties to appear at the top of the list regardless of their category, click both the Show Alphabetized List button and the Show Set Properties on Top button.

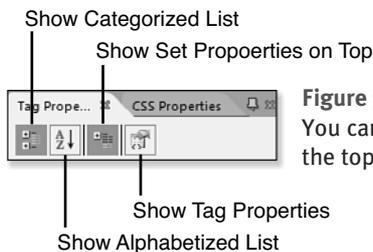


Figure 7.3

You can customize the display of the Tag Properties panel using the buttons at the top of the panel.

Viewing Tag Properties with the Tag Properties Panel

To view the properties of an HTML tag, select an HTML element in Design View or click inside an HTML tag in Code View. The Tag Properties panel displays all the properties that are applicable to the selected tag.

If a property is set for the selected tag, the property name appears as bold text (see Figure 7.4). The name of the selected tag also appears at the top of the Tag Properties panel. If you hover over the tag name, the HTML code for that tag appears inside a ScreenTip.



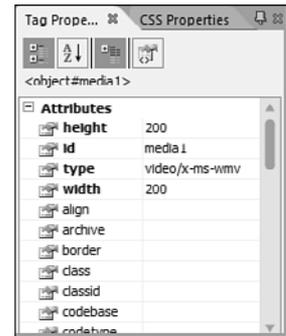
tip

Each property listed in the Tag Properties panel has an icon at the left of the property name. The icon indicates the type of the property. A finger pointing to a page indicates an attribute; a lightning bolt indicates an event.

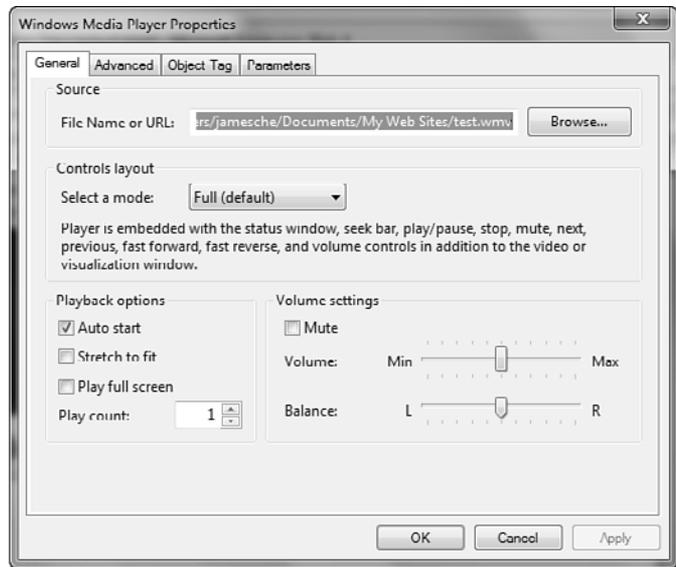
If you want more details on a tag's properties, click the Show Tag Properties button, as shown previously in Figure 7.3. In Figure 7.4, the properties for an `<object>` tag are displayed. Figure 7.5 shows the Windows Media Player Properties dialog that is displayed when the Show Tag Properties button is clicked. Note that the values in the dialog match those in the Tag Properties panel in Figure 7.4.

Figure 7.4

When a property is set, the property name appears in bold text. To see all the HTML code for the selected tag, hover over the tag's name.

**Figure 7.5**

The Windows Media Player Properties dialog is displayed when the `<object>` tag is selected and the Show Tag Properties button is clicked.



Setting Tag Attributes with the Tag Properties Panel

The Tag Properties panel provides a quick and convenient way to set tag attributes. Setting an attribute of a particular tag differs based on the attribute you are setting. For example, some attributes (such as the `alt` attribute of an `` tag) are free text. Other attributes (such as the `align` attribute of an `` tag) have only certain values that are valid. Other attributes (such as the `style` attribute) have a fairly complex syntax. The Tag Properties panel makes setting all these attributes fast and easy.

Creating a Page

Let's create a page so we can more easily work with setting tag attributes:

1. Create a new page.
2. Add a new table with the default settings.
3. Add an image of your choice to one of the table cells.
4. Add a hyperlink to one of the table cells.
 - For more information on adding tables to a page, see Chapter 5, "Using Tables."
 - For more information on adding images to a page, see Chapter 9, "Using Graphics and Multimedia."
 - For more information on adding hyperlinks to a page, see Chapter 3, "Creating Pages and Basic Page Editing."



note

The `alt` attribute is important for images because screen readers read the text in the `alt` attribute aloud so that people with poor vision can better utilize your page.

Setting Tag Properties

Click to select the image you inserted on the page. The Tag Properties panel shows that the `src`, `height`, and `width` attributes have been set for the image. However, the current standards require an `alt` attribute for all `` tags.

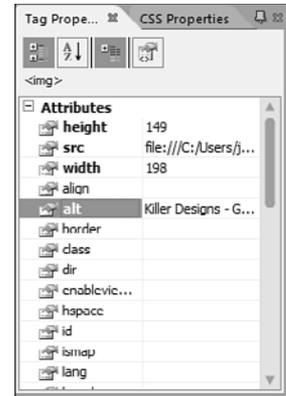
To set the `alt` attribute using the Tag Properties panel:

1. If the image is not selected, select it first.
2. In the Tag Properties panel, locate the `alt` attribute.
3. Enter some text of your choice for the `alt` attribute.

The Tag Properties panel should now look like Figure 7.6.

Figure 7.6

The `alt` attribute for this image has been set using the Tag Properties panel.



Set Properties Don't Appear First

For newly set properties to appear first in the panel, you need to refresh it. The easiest way to refresh the panel is to click the Show Set Properties on Top button once to turn off that setting, and then click it again to turn the setting back on. This refreshes the panel and set properties should appear first in the list as expected.

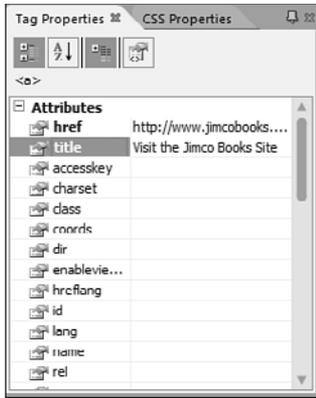
Now let's add a ScreenTip to the hyperlink. ScreenTips are added to hyperlinks using the `title` attribute.

1. Click the hyperlink so the Tag Properties panel shows the properties of the `<a>` tag.
2. Locate the `title` attribute in the Tag Properties panel and add the text you want to appear in the ScreenTip.

In Figure 7.7, the `title` attribute has been specified for the hyperlink.

Sometimes, you might want a hyperlink to open a new window. The HTML `target` attribute allows you to do this easily. You can specify a specific frame name for the `target` attribute, or you can use one of several predefined values. For example, to specify that a link should open in a new window, set the `target` attribute to a value of `_blank`:

1. Click inside the hyperlink to make the `<a>` tag active.
2. Locate and select the `target` attribute in the Tag Properties panel.
3. Click the down arrow for the `target` attribute's value and select `_blank`, as shown in Figure 7.8.

**Figure 7.7**

The `title` attribute has now been specified for this hyperlink. Setting the `title` attribute is important if you want your pages to be standards-compliant.

Now the hyperlink opens in a new window when clicked. You can make the same change to a hyperlink in several ways. In addition to selecting the desired value for the `target` attribute, you can also click the Show Tag Properties button on the Tag Properties panel (refer to Figure 7.3) and make the change in the Edit Hyperlink dialog box.

➔ *For more information on using the Edit Hyperlink dialog, see Chapter 3, “Creating Pages and Basic Page Editing.”*

**Figure 7.8**

There's no need to try to remember all the possible values for the `target` attribute. Simply choose the desired value from the list provided by the Tag Properties task pane.

Using Events with the Tag Properties Panel

In addition to examining and setting tag attributes, the Tag Properties panel is a convenient way to connect event handlers for page elements.

➔ *For an overview of events, see the sidebar “Attributes and Events” on p. 125.*

Let's configure the image on the page so that clicking it links to a different site. We could wrap the image in an `<a>` tag and make it a hyperlink, but for this example, we'll use the `click` event for the image. By using the `click` event, you can perform a specified action when the image is clicked. In this case, we run some JavaScript code when the image is clicked.

Switch to Code View and add the following code before the closing `</head>` tag:

```
<script language="javascript" type="text/javascript">

    function openLink(url)
    {
        window.navigate(url);
    }

</script>
```

This code opens the URL passed to it.

➔ *For more information on JavaScript, see Chapter 22, "Client Scripting."*

We now need to hook up the `click` event to the image. When the image is clicked, we want to run the `openLink` JavaScript function we just added. Here's how you do that:

1. Select the image.
2. In the Tag Properties panel, locate the `onclick` event.
3. Set the following value for the `onclick` event:

```
javascript:openLink("http://www.jimcosoftware.com");
```

4. Save the page and preview it in your browser.
5. Click the image to navigate to the Jimco Software site.

**tip**

If you are previewing the page in Internet Explorer via the file system, Internet Explorer warns you about running active content and asks whether you want to allow it. You need to allow active content to test the `click` event of the image.

Alternatively, you can add the Mark of the Web (MOTW) to the page and Internet Explorer will run the script without a warning. To add the MOTW, select Insert, HTML, Mark of the Web.

Tag Properties and Web Standards

To ensure that your pages comply with current web standards, be careful when setting properties using the Tag Properties panel. It's easy to accidentally fall out of compliance when setting attributes using this tool.

For example, if you select the image you added to the sample page for this chapter and set the border attribute to 4 in the Tag Properties panel, Expression Web adds a border attribute with a value of 4. However, the border attribute is considered a deprecated attribute, and it will be flagged as such by Expression Web's Code View, as shown in Figure 7.9.

```
29 </td>
32 <
33 tr>
34 <
```

Figure 7.9
Be careful when specifying attribute values. You can easily fall out of standards-compliance.

Is this a bug in Expression Web? Not at all. Expression Web is simply doing what you asked it to do. In this case, you explicitly told Expression Web to add code that is not compliant with current standards. Expression Web warns you when you do this, but it won't stop you.

When setting attributes using the Tag Properties panel, check Code View often for HTML incompatibilities.

➡ *For more information on using Code View to detect HTML compatibilities, see Chapter 3, "Creating Pages and Basic Page Editing."*

USING THE QUICK TAG TOOLS

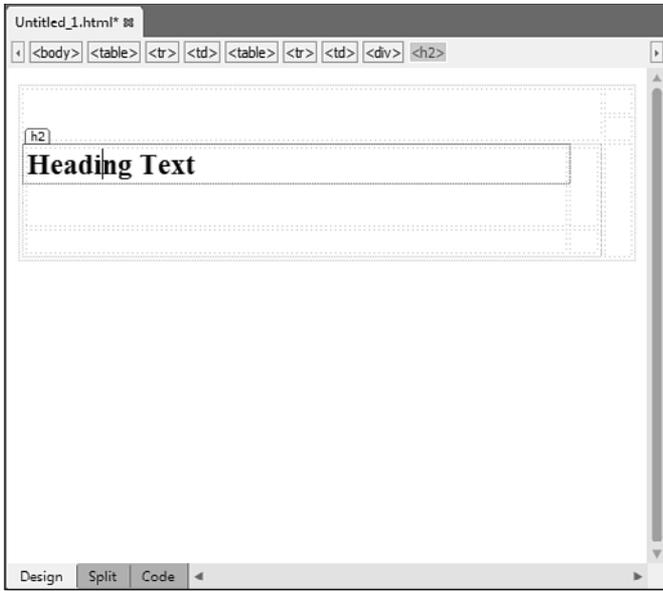
Introduction to the Quick Tag Tools

Years ago, I obtained a copy of HomeSite. HomeSite was a web development tool created by Nick Bradbury, and it was a great tool as long as you knew HTML. However, if you didn't know HTML well, you could get lost quickly. HomeSite didn't have a nice what-you-see-is-what-you-get (WYSIWYG) interface like those we enjoy in today's web development tools.

Pages in those days were simple compared to today's pages, and yet even in those simpler times, it was often difficult to keep track of nested HTML elements such as table cells. As pages have increased in complexity, and as new HTML elements have been added to our repertoire, it has become even more difficult to comprehend the relationship between a page element and the underlying HTML code.

Tables are an excellent example of this increasing level of complexity. In the days of pure HTML editors, web developers (no one called them "designers" at that time) kept tables pretty simple. Creating a large nested table structure was just asking for trouble because it would quickly become unmanageable. The advent of WYSIWYG editors removed that barrier, and heavily nested tables became so commonplace that locating the HTML code for a particular table cell was like finding the proverbial needle in a haystack.

Microsoft provided a nice solution to that problem with the introduction of Quick Tag Tools in FrontPage 2003. Fortunately for all of us, Quick Tag Tools are still available in Expression Web. Quick Tag Tools, shown in Figure 8.1, provide a context-sensitive way of locating and working with specific elements in a page.

**Figure 8.1**

Expression Web's Quick Tag Tools make it easy to locate and work with elements in your pages.

The Quick Tag Tools consist of two sets of tools: the Quick Tag Selector and the Quick Tag Editor. Let's look at each of these Quick Tag Tools separately and at how you can use them to effectively edit your page content.



Quick Tag Tools Unavailable

If you're clicking in your page but don't see any of the Quick Tag Tools, chances are you have the Quick Tag Selector option turned off. Select View, Quick Tag Selector to turn it back on.



tip

In Design View, only visible page elements have a corresponding tag selector. You will not see tag selectors for CSS code, script, or other such elements.

To access tag selectors for code elements such as script tags, switch to Code View or Split View.

Locating and Selecting Elements Using the Quick Tag Selector

If you've ever tried to work with the HTML code in a page with nested elements such as the nested table scenario I mentioned earlier, you no doubt understand how difficult it can be to locate HTML

code for a specific element. The Quick Tag Selector in Expression Web makes editing complex pages ridiculously easy by providing a series of context-sensitive buttons for each page element.

As you select different elements in Design View or select code in Code View or Split View, the Quick Tag Selector displays one or more tag selectors in a toolbar above the view, as shown previously in Figure 8.1.

➔ *For more information on Design View, Code View, and Split View, see Chapter 4, “Using Page Views.”*

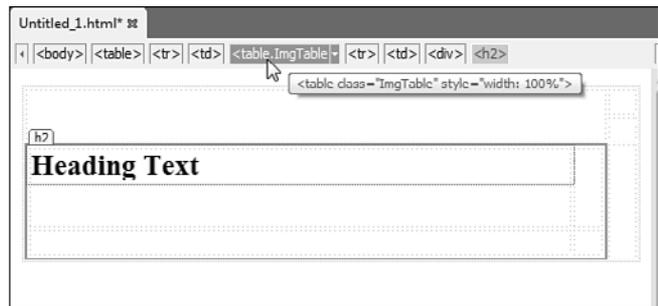
The Quick Tag Selector works a little differently in Design View than in the other views. As you hover over individual tag selectors, Expression Web draws a highlighted border around the corresponding page element in the designer, as shown in Figure 8.2.

**tip**

You can see the entire HTML code for any tag simply by hovering over the tag selector with your mouse. A ScreenTip appears showing the HTML code for that tag.

Figure 8.2

As you hover over tag selectors, Expression Web displays a highlighted border around the corresponding page element.



Not only is this convenient when you're trying to identify page elements, but it's also an extremely valuable tool when you are working with CSS elements because it displays CSS classes and IDs as part of the tag selector. Notice in Figure 8.2 that the tag selector for the table indicates that the `ImgTable` CSS class was applied to the table.

➔ *For more information on CSS classes and IDs, see Chapter 17, “Creating Style Sheets.”*

To select a page element or block of code using the Quick Tag Selector, simply click the tag selector button. If you are in Design View, the selected page element will be highlighted, as shown in Figure 8.3. If you are in Code View or Split View, the code for the selected element will be highlighted, as shown in Figure 8.4.

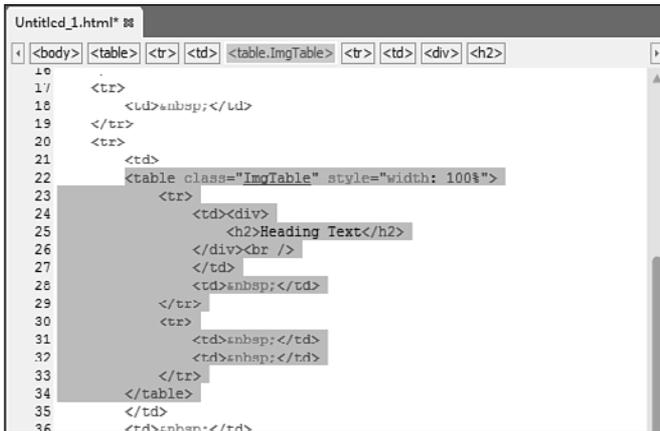
If you click the small arrow at the right edge of a selected tag selector or right-click a tag selector, a menu appears. You can then choose **Select Tag** from the menu to select the page element or code block. Alternatively, if you choose **Select Tag Contents** from the menu, only the contents of the tag are selected and not the tag itself.

**tip**

If you choose **Select Tag Contents** for a `<table>` tag, both the `<td>` tags and the content within them will be selected. The `<tr>` tags will remain unselected.

**Figure 8.3**

In Design View, selecting a quick tag selector highlights the corresponding page element.

**Figure 8.4**

In Code View or Split View, selecting a quick tag selector highlights the corresponding code.

Editing Page Content Using the Quick Tag Editor

Now that you know how to locate and select content using the Quick Tag Selector, let's look at how to edit your page's content using the Quick Tag Editor.

The Quick Tag Editor is accessible by either clicking the small arrow at the right edge of a selected tag selector or by right-clicking a tag selector. The following options are available:

- **Edit Tag**—Displays a Quick Tag Editor dialog populated with the HTML for the tag for easy editing.
- **Remove Tag**—Removes the selected tag. If the tag wraps any other content, the content that it originally wrapped remains on the page.
- **Insert HTML**—Displays a Quick Tag Editor dialog where you can enter HTML to be inserted onto the page.

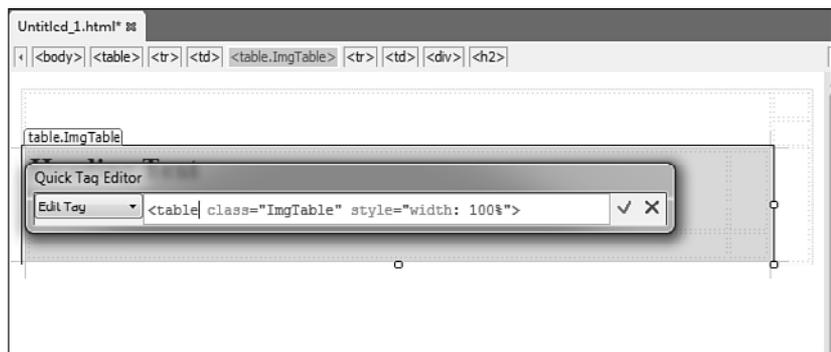
- **Wrap Tag**—Displays a Quick Tag Editor dialog where you can enter HTML. The HTML entered wraps the selected tag.
- **Positioning**—Provides easy access to the Positioning menu for the positioning of page elements.
- **Tag Properties**—Allows you to easily access the Properties dialog for the selected tag.

Let's look at each of these options and learn how to use them to edit your content.

Editing a Tag

Selecting the Edit Tag option in the Quick Tag Editor provides you with a Quick Tag Editor dialog containing the HTML tag, as shown in Figure 8.5. Expression Web provides color-coding and IntelliSense support in this dialog as well.

Figure 8.5
The Quick Tag Editor dialog allows for the editing of a specific HTML tag and is complete with IntelliSense and color-coding.



- ➔ For more information on color-coding, see Chapter 11, “Configuring Page Editor Options.”
- ➔ For more information on IntelliSense, see Chapter 3, “Creating Pages and Basic Page Editing.”

The easiest way to fully comprehend the Quick Tag Editor is to create a simple page and experiment with the tools. Follow these steps:

1. Create a new page.
2. Insert a new table using the default settings.
3. Place the insertion point inside one of the table cells.
4. Insert a new table inside the first table's cell using the default settings.



tip

The Quick Tag Tools disappear from Code View after you make a change to your code. This is by design for performance reasons.



tip

Instead of clicking the green check mark button to commit your change, you can simply press Enter on your keyboard.

If you now click within the tables you've inserted, you'll see that the tag selectors available will change. Click inside the inner table and you should see a tag selector for both tables. Let's add an ID attribute to the inner table using the Quick Tag Editor:

1. Right-click the tag selector for the inner table and select **Edit Tag** from the menu to open the Quick Tag Editor, as shown in Figure 8.6.
2. The insertion point should be just to the right of the `<table>` tag. Press the spacebar to open the IntelliSense window.
3. Type `i` to jump to the `id` attribute, as shown in Figure 8.7.
4. Double-click the `id` attribute to add it.
5. Enter `innerTable` as the value of the `id` attribute.
6. Click the check mark button at the right end of the Quick Tag Editor dialog to commit the new code to the page.

The Quick Tag Editor does a cursory check on the HTML you enter, and if it's blatantly invalid, Expression Web displays an error, as shown in Figure 8.8.



caution

You can enter invalid attributes or other errors that will not cause an error dialog. For example, the Quick Tag Editor happily allows me to add an attribute to a `<p>` tag called `jim` even though there is no such attribute.

Be careful when editing your HTML to avoid accidental errors.



tip

Even if the Quick Tag Editor approves an edit, Expression Web's Code View might warn you that your code is invalid. Check Code View for any possible problems.

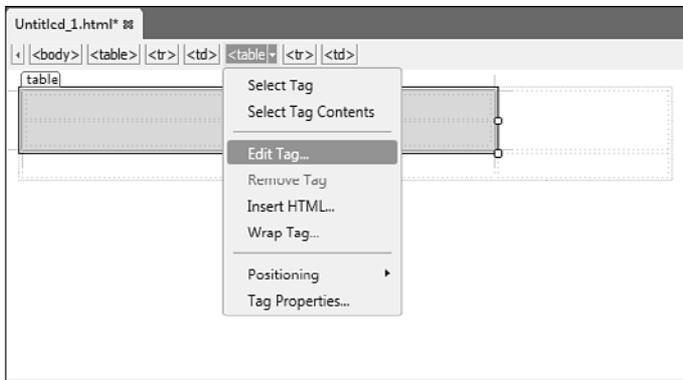


Figure 8.6

The **Edit Tag** option displays the Quick Tag Editor where you can edit the HTML code for the selected element.

Removing a Tag

The **Remove Tag** option is a convenient way of removing the selected HTML tag. When you choose this option, only the selected HTML tag is removed. Any content contained within the tag remains.

Figure 8.7
IntelliSense support in the Quick Tag Editor allows you to easily and accurately add an ID attribute to the table.

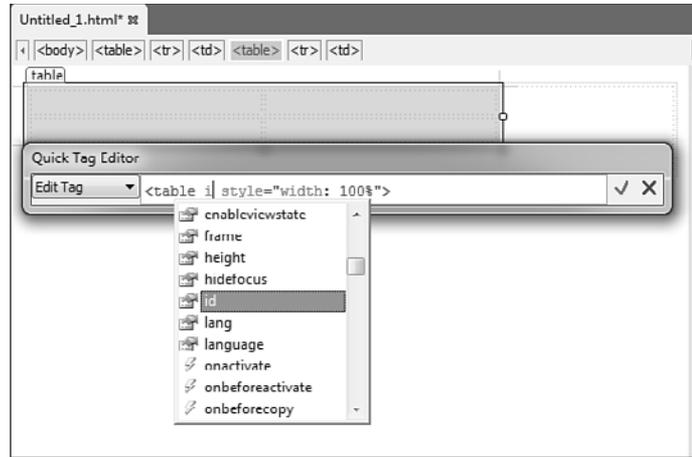
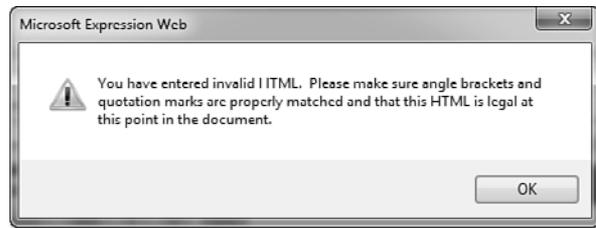


Figure 8.8
The Quick Tag Editor displays an error if you attempt to enter obviously incorrect HTML.



To use the Remove Tag option, do the following:

1. Place the insertion point inside a table cell and enter the text **Microsoft**.
2. Double-click the text you entered to select it.
3. Right-click the selected text and select Hyperlink from the menu.
4. Enter **http://www.microsoft.com** in the Address box and click OK.

After completing these steps, Expression Web inserts an <a> tag into the page that links to the Microsoft site. To remove that tag follow these steps:

1. Click the hyperlink you just created in Design View.
2. Right-click the tag selector for the <a> tag and select Remove Tag from the menu.

When the <a> tag is removed, the text for the hyperlink remains. Only the tag that wrapped the text is removed.



Remove Tag Not Enabled

The Remove Tag option is available only for tags that contain other content. It also can be disabled for tags that cannot be removed without causing errors in the page. For example, even though a `<td>` tag contains the contents of a table cell, the tag cannot be removed without creating invalid code. Therefore, the Remove Tag option is not enabled for `<td>` tags.

Inserting HTML

The Insert HTML option provides you with a Quick Tag Editor dialog for inserting HTML code into a page. As simple as that may sound, it is sometimes difficult to predict where code gets inserted.

If nothing is selected, code that you insert using the Insert HTML option is inserted at the insertion point. However, if you have an element selected on the page, the HTML is inserted immediately before the selected element. Because of this, you can attempt to insert HTML that alone is perfectly valid, but in the context of the selection, the code is invalid. For example, if you select a `<td>` tag and attempt to insert a `<div>` before it, you see the dialog displayed previously in Figure 8.8 and Expression Web prevents you from inserting the code.

To insert HTML using the Insert HTML option, follow these steps:

1. Create a new page.
2. Right-click the `<body>` tag selector and select Insert HTML.
3. In the Quick Tag Editor, replace the empty opening and closing brackets with the following HTML:

```
<font>Inserted with the Quick Tag Editor</font>
```

4. Press Enter or click the green check mark button to add the HTML to the page.

The HTML code you just entered is not standards-compliant because the `` element is deprecated. The Quick Tag Editor does not prevent you from inserting the deprecated code. It does, however, highlight the code as falling outside the standards in Code View, as shown in Figure 8.9.

➔ *For more information on developing standards-based sites, see Chapter 12, “Maintaining Compatibility and Accessibility.”*



tip

Expression Web performs only a cursory examination of the HTML code you are entering. It allows you to enter invalid HTML as well. For example, if you omit a closing tag on an element that requires one, your invalid code is inserted without warning.



caution

Expression Web was designed to produce standards-compliant code. Therefore, be careful when inserting your own HTML code so you don't add code that falls outside the standards.

Figure 8.9
The Quick Tag Editor happily inserts HTML that falls outside current standards, but Code View warns you about it.

```

Untitled_1.html*
<body>
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
2 <html xmlns="http://www.w3.org/1999/xhtml">
3
4 <head>
5 <meta content="en-us" http-equiv="Content-Language" />
6 <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
7 <title>Untitled 1</title>
8 </head>
9
10 <body>
11
12 <font>Inserted with the Quick Tag Editor</font>
13 The World Wide Web Consortium now regards the tag <font> as outdated. Newer constructs are recommended.
14 </body>
15
16 </html>
17

```

Wrapping a Tag

The Wrap Tag option wraps the selected HTML element with the tag that you enter in the Quick Tag Editor dialog. Consider the HTML that we just entered using the Insert HTML option. We inserted some text wrapped in a `` tag. As I mentioned, the `` tag is deprecated and we should have used the `<p>` tag instead. To correct that problem using the Wrap Tag option, do the following:

1. Click inside the text you entered using the Insert HTML option.
2. Right-click the `` tag selector and select Wrap Tag.
3. In the Quick Tag Editor dialog, enter `p` between the opening HTML tag characters, as shown in Figure 8.10.
4. Press Enter or click the green check mark button to commit the change.
5. Switch to Code View or Split View to examine the code that was entered. Notice that the `` tag is now wrapped in a `<p>` tag.
6. Click inside the text again.
7. Right-click the `` tag and select Remove Tag.

Figure 8.10
The Wrap Tag option wraps an HTML tag with the tag that you enter in the Quick Tag Editor.



After completing the steps, the `` tag is removed and replaced with a `<p>` tag, and the code is now compliant with current standards.

Controlling Positioning

The Positioning option provides easy access to CSS positioning settings for the selected element. The following positioning options are available:

- **(None)**—This is the default setting.
- **position: absolute**—When this option is chosen, the selected element can be dragged to any position within the page. You can also set the z-order of the element.
- **position: fixed**—This option is similar to the `position: absolute` option except that fixed elements do not move when the page is scrolled in the browser.
- **position: relative**—This option allows you to position an element relative to where it was originally inserted onto a page.
- **position: static**—This option is functionally the same as setting positioning to (None).

➡ *For more information on CSS positioning and z-order, see Chapter 23, “Using Layers.”*

Editing Tag Properties

The Tag Properties option provides a means of accessing Properties dialogs of HTML elements using the Quick Tag Editor. To use the Tag Properties option:

1. Create a new page and insert a new table using the default settings.
2. Place the insertion point inside the table.
3. Right-click the `<table>` tag selector and select Tag Properties.

The Table Properties dialog is displayed so you can alter the properties of the table. Alternatively, if you select the Tag Properties option for a `<td>` tag, the Cell Properties dialog is displayed.

Some HTML tags do not have a corresponding Properties dialog in Expression Web. If you click Tag Properties for a tag that does not have a corresponding Properties dialog, Expression Web will not display a dialog.

When to Use the Quick Tag Editor

The Quick Tag Editor is a powerful tool for making changes to your HTML code. However, as Spider-Man would say, “With great power comes great responsibility.” Microsoft put a lot of work into ensuring that the code Expression Web creates is standards-compliant. They did that because professional web designers asked for a tool they could trust to create pages that would

validate against current standards. Using the Quick Tag Editor indiscriminately can easily sabotage Expression Web's code. For example, although Expression Web goes to great lengths to keep from generating inline CSS styles that might make it difficult to maintain a page, you can add inline styles yourself and override any styles that Expression Web has already created.

Speaking of CSS, one of the greatest uses for the Quick Tag Editor is making changes to CSS code. For example, applying a CSS class or ID to a specific tag is simple and fast using the Quick Tag Editor. However, you can certainly use the Quick Tag Editor for editing any code. When you do, check Code View often to ensure you haven't introduced nonstandard code into your page. Remember that Expression Web does not prevent you from shooting yourself in the foot. Keeping your code standards-compliant is up to you.

This page intentionally left blank

USING GRAPHICS AND MULTIMEDIA

Web Image Formats

It's often said that content is king in a site. If that's true (and I agree that it is), graphics definitely hold the title of queen. Quality graphics often make the difference between an amateur site and a professional one. Nothing can add an aura of professionalism like quality graphics.

Not all graphics are created equal, however. Part of projecting a professional impression is the proper choice of image formats. The most common image formats on the Web are GIF and JPEG. PNG is another image format that is growing in popularity for many reasons, some of which we discuss in this section.

The GIF Format

The GIF format is best suited to nonphotographic images that contain mostly straight lines. (Text, for example, is best suited to the GIF format.) GIF images can contain a maximum of 256 colors, but you can save them with fewer colors to achieve a smaller file size.

The GIF format has the advantage of being a *lossless* format. That means, in most cases, you can edit a GIF image as many times as you want without reducing the quality of the image. GIF images are also color-indexed, which means each image has a table of colors saved with it. By reducing the number of colors in the color table, you can reduce the file size of an image. If you remove a color that actually exists in the image, the GIF format can combine other colors in the color table to try to reproduce the color that was removed. This process is known as *dithering*.

GIF images also support transparency. Colors in the color table can be marked as transparent. A color marked as transparent is not visible in the final image; instead, the content underneath the image shows through. Transparent images are useful when you want background colors to show through, and they are useful as a formatting tool. Many designers place a transparent GIF inside a table cell to act as an invisible spacer.

The JPEG Format

JPEG images are best suited to photos and other graphics that consist of many colors. The JPEG format supports full-color images and can be compressed with excellent results. However, unlike GIF images, once you compress a JPEG image, it loses quality that can never be recovered. If a JPEG is edited multiple times, the quality can be dramatically degraded.

JPEG images do not support transparency, so if you want an irregularly shaped JPEG to appear on a particular background color, make the background of the actual image the same color as the background color on which it will appear.

The PNG Format

PNG images combine the best features of GIF and JPEG in a single format. PNG images support transparency and full-color reproduction.

Inserting Images

To insert an image, select Insert, Picture, From File. After you insert the desired image, Expression Web displays the Accessibility Properties dialog, as shown in Figure 9.1.



Figure 9.1

Using the Accessibility Properties dialog is the easiest way to ensure that most visitors can access your site.

You can also insert a picture directly from a scanner or camera by selecting Insert, Picture, From Scanner or Camera. The interface you see with this method is dependent on the camera or scanner.



note

The PNG format supports alpha transparency, which means a color can be set to varying degrees of transparency.



caution

Not all browsers fully support the PNG format. You can check your browser's compatibility at www.w3.org/Graphics/PNG/.



caution

It is strongly recommended that you include alternative text and long descriptions for all images to comply with current accessibility guidelines.



Images Sometimes Not Visible

If you added some images to your site and they show up fine when you're on your own computer, but you don't see the images when you go to another computer, it's likely that your image links are pointing to files on your local computer. Open the page and look at the images in Code View. If you see a file path as the source of an image, that's the problem.

The following code shows an image source path pointing to a local file path:

```

```

If you browse this page on the computer containing that image, the graphic will show up fine. However, if you go to another computer, the graphic won't display.

To resolve this issue, import the image into the current site so the link to the image is relative to the existing site. The following code shows an image that is correctly linked to the current site:

```

```

Inserted images can be located anywhere. If an image is not already located in the current site, Expression Web displays the Save Embedded Files dialog and prompts you to save the image into the site when the page is saved, as shown in Figure 9.2.

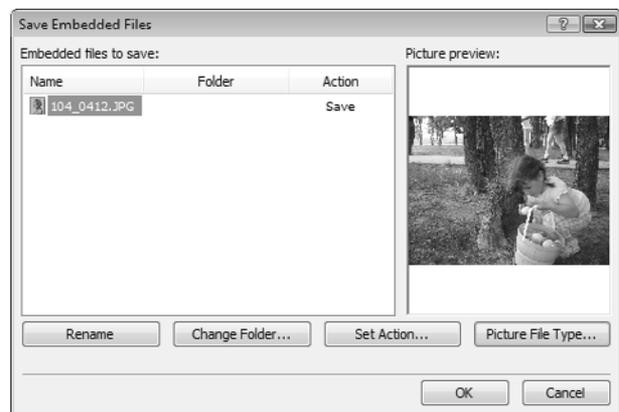


note

Expression Web also displays the Save Embedded Files dialog when you edit an existing image on a page.

Figure 9.2

Expression Web prompts you to save embedded images when a picture is not already located in the current site.



On some occasions you might want to link directly to an image on another site. When you do this, Expression Web displays the Save Embedded Files dialog and prompts you to save the image into the site. If you'd prefer not to save the image into your site, click the Set Action button and select the Don't Save This File option, as shown in Figure 9.3.



caution

Before you link directly to an image on someone's site, make sure it's okay with the site owner. There are plenty of legitimate uses of this technique (affiliate images, for example), but it's something you don't want to do unless the owner of the site hosting the image approves.

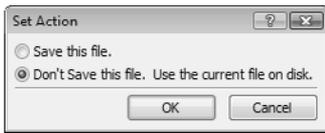


Figure 9.3

You can choose to leave an image where it is instead of saving it to your site.

Formatting Images

Many formatting options are available after you've inserted an image.

Resizing Images

To resize an image, select the image and drag one of the sizing handles that appear around the edges of the image. By default, Expression Web allows you to change an image's proportions. In other words, when you change an image's width, the height will not change proportionately by default.

In many cases, you just need to size an image without changing the proportions. To do that, press and hold the Shift key while dragging the sizing handle at the lower-right corner of the image. As you size the image either horizontally or vertically, Expression Web sizes each dimension proportionately.

After an image is resized, Expression Web displays the Picture Actions button. Clicking the button displays the options shown in Figure 9.4.

Using the Picture Actions options, you can choose between simply modifying the height and width attributes in the HTML code or resampling the file.

Resampling is often the best choice, especially if you have sized the image to a smaller dimension. When you resample an image that is smaller than its original dimensions, Expression Web applies a complex algorithm to the image. Doing so not only makes it look much better, it also substantially reduces the file size.



tip

Expression Web also includes the ability to import Photoshop images directly into your site. For more information, see "Importing Adobe Photoshop Files" later in this chapter.

Figure 9.4

The Picture Actions options can make an image file smaller for faster downloads.



Changing Picture Properties

To edit the properties of an image, select the image, and then select Format, Properties to display the Picture Properties dialog shown in Figure 9.5.

On the General tab of the Picture Properties dialog, you can change many general options for an image, including configuring a hyperlink for the image. You also can edit the image in an external editor by clicking the Edit button. When you click the Edit button, the image file opens in the image editing application associated with the image type.



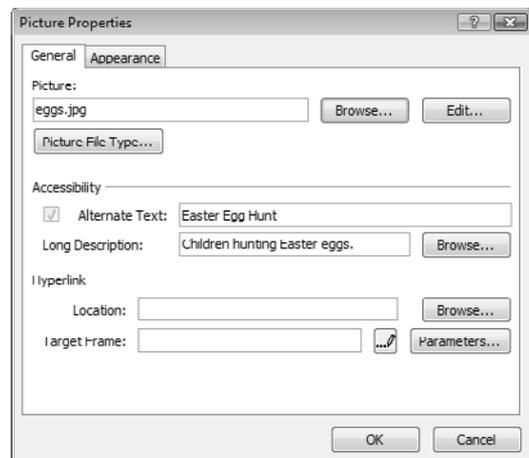
tip

You can also access the Picture Properties dialog by double-clicking an image or by right-clicking an image and selecting Picture Properties from the menu.

➔ For information on configuring editors for images and other files, see “Configuring File Editors,” in Chapter 3, p. 65.

Figure 9.5

Many properties of an image can be modified using the Picture Properties dialog.



The Appearance tab of the Picture Properties dialog provides access to many useful image formatting tools. As shown in Figure 9.6, you can modify an image's wrapping style, layout, and size.

One of the most useful properties of an image is its wrapping style. Figure 9.7 shows an image with the default wrapping style of None. The same page is shown in Figure 9.8, but the wrapping style of the image has been changed to Left. Notice that the image appears on the left edge of the text in Figure 9.8.

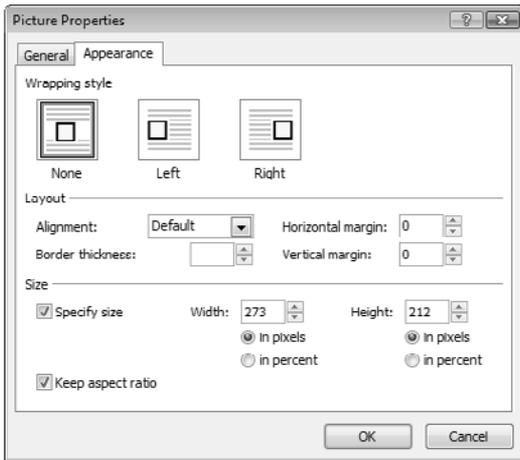


Figure 9.6
Many useful properties can be modified on the Appearance tab of the Picture Properties dialog.

You might have noticed that the text in Figure 9.8 appears right up against the image. By adjusting the Horizontal Margin in the Picture Properties dialog, you can add a small buffer between the image and the text, as shown in Figure 9.9.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer egestas. Nulla facilisi. Phasellus nec ante. Nulla sodales odio nec pede. Nulla ligula felis, gravida ac, dictum eget, faucibus eget, sem. Fusce sem. Proin blandit, neque et convallis consectetur, sem nulla accumsan est, sit amet facilisis purus elit vel diam. In hac habitasse platea dictumst. Nam posuere, nisi non convallis egestas, arcu dolor ultrices tortor, sit amet tristique mauns purus non nisl. Morbi nisi dolor, mattis eu, venenatis vel, scelerisque a, quam. Aenean facilisis pretium mi. Nam nec metus ac sem condimentum malesuada. Mauris luctus eleifend orci. Proin mauris. Morbi convallis porta odio. Morbi volutpat felis sit amet nibh. Phasellus nec augue. Pellentesque sed urna nec nisl vehicula volutpat.

Figure 9.7
The image pictured here has the wrapping style set to None, the default setting. Notice that the text appears below the image.

Figure 9.8

Here is the same page as in Figure 9.7, but the image now has a wrapping style of Left. Note that the image appears on the left edge of the text.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer egestas. Nulla facilisi. Phasellus nec ante. Nulla sodales odio nec pede. Nulla ligula felis, gravida ac, dictum eget, faucibus eget, sem. Fusce sem. Proin blandit, neque et convallis consectetur, sem nulla accumsan est, sit amet facilisis purus elit vel diam. In hac habitasse platea dictumst. Nam posuere, nisl non convallis egestas, arcu dolor ultrices tortor, sit amet tristique maurs purus non nisl. Morbi nisi dolor, mattis eu, venenatis vel, scelensque a, quam. Aenean facilisis pretium mi. Nam nec metus ac sem condimentum malesuada. Mauris luctus eleifend orci. Proin mauris. Morbi convallis porta odio. Morbi volutpat felis sit amet nibh. Phasellus nec augue. Pellentesque sed urna nec nisl vehicula volutpat.

Donec arcu magna, tempor nec, consequat condimentum, malesuada et, lorem. Aenean tempor interdum mi. Integer eleifend, turpis vel suscipit malesuada, dui turpis accumsan quam, ut ultricies magna massa gravida lorem. Sed at elit eget odio dignissim cursus. Sed consectetur. Etiam sit amet diam. In suscipit mollis urna. Curabitur tempus dui nec arcu. Integer ac odio. Mauris sed ligula. Fusce fringilla elit sit amet enim. Morbi quis ligula. Nullam vehicula sapien in odio. Fusce condimentum odio ultricies arcu ultrices adipiscing. Aenean non nunc eget sapien tristique consequat. Suspendisse eu magna vel dui malesuada adipiscing. Fusce metus sem, sodales nec, ornare vitae, tempor nec, nisl. Proin nec sapien.

Figure 9.9

With the addition of a 10 px horizontal margin, the page now looks much cleaner.



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer egestas. Nulla facilisi. Phasellus nec ante. Nulla sodales odio nec pede. Nulla ligula felis, gravida ac, dictum eget, faucibus eget, sem. Fusce sem. Proin blandit, neque et convallis consectetur, sem nulla accumsan est, sit amet facilisis purus elit vel diam. In hac habitasse platea dictumst. Nam posuere, nisl non convallis egestas, arcu dolor ultrices tortor, sit amet tristique mauris purus non nisl. Morbi nisi dolor, mattis eu, venenatis vel, scelerisque a, quam. Aenean facilisis pretium mi. Nam nec metus ac sem condimentum malesuada. Mauris luctus eleifend orci. Proin mauris. Morbi convallis porta odio. Morbi volutpat felis sit amet nibh. Phasellus nec augue. Pellentesque sed urna nec nisl vehicula volutpat.

Donec arcu magna, tempor nec, consequat condimentum, malesuada et, lorem. Aenean tempor interdum mi. Integer eleifend, turpis vel suscipit malesuada, dui turpis accumsan quam, ut ultricies magna massa gravida lorem. Sed at elit eget odio dignissim cursus. Sed consectetur. Etiam sit amet diam. In suscipit mollis urna. Curabitur tempus dui nec arcu. Integer ac odio. Mauris sed ligula. Fusce fringilla elit sit amet enim. Morbi quis ligula. Nullam vehicula sapien in odio. Fusce condimentum odio ultricies arcu ultrices adipiscing. Aenean non nunc eget sapien tristique consequat. Suspendisse eu magna vel dui malesuada adipiscing. Fusce metus sem, sodales nec, ornare vitae, tempor nec, nisl. Proin nec sapien.

Converting Images

You can easily convert an image from one format to another. To convert an image, right-click the image and select Change Picture File Type to display the Picture File Type dialog, as shown in Figure 9.10.

You might want to convert an image to another format to make the image file size smaller. For example, if you have been given a photograph in GIF format for use in a site, you'll likely be able to substantially reduce the file size of the image by converting it to a JPEG image. On the other hand, if you are given a logo in JPEG format that contains only a few colors, you can likely make the file much smaller by converting the logo to a GIF image.

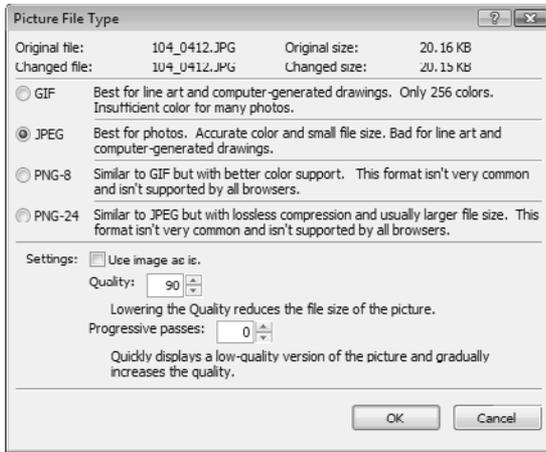


Figure 9.10
Changing a picture file type can reduce an image's file size.

The options available in the Picture File Type dialog differ based on the image format you select. Figure 9.10, shown previously, shows the options for the JPEG format. If you select GIF as the image format, you are given the option to make the image transparent.

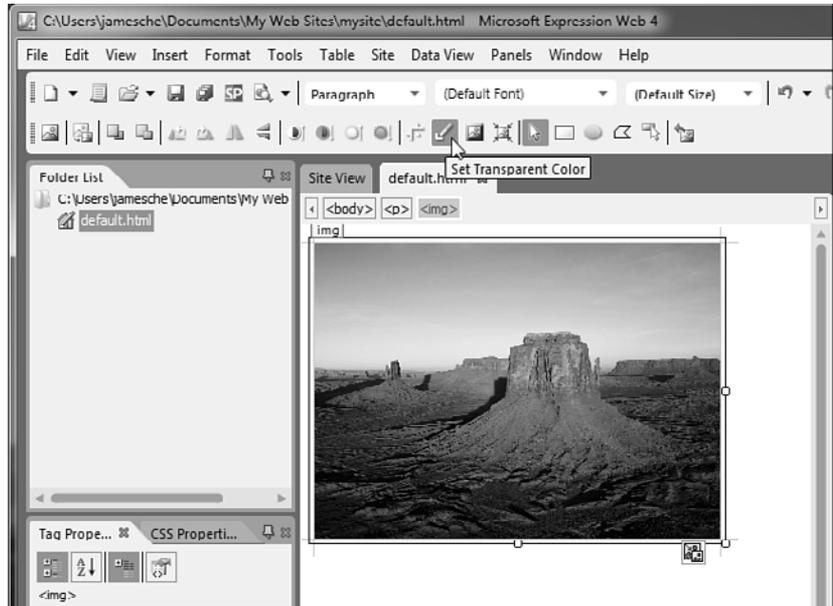
You can set the transparency of a GIF image only if a transparent color has already been selected for the image using the Pictures toolbar or a graphics editing program. Follow these steps to set the transparent color using the Pictures toolbar:

1. Select View, Toolbars, Pictures to display the Pictures toolbar.
2. Select the image.
3. Click the Set Transparent Color button on the toolbar, as shown in Figure 9.11. If the image is not already a GIF image, Expression Web informs you that it will convert the image.
4. Click the color you want to make transparent. The color should immediately disappear, revealing the color underneath it.

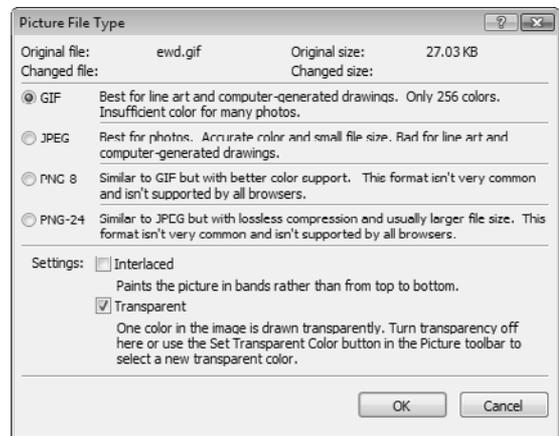
After you've added transparency to an image, you can use the Picture File Type dialog to disable transparency again by unchecking the Transparent check box shown in Figure 9.12.

Figure 9.11

The Pictures toolbar lets you easily set a transparent color on a GIF image.

**Figure 9.12**

The Transparent check box is checked when transparency has been applied to an image. Unchecking it removes the transparency and restores the image's original color.



Creating Image Thumbnails

A common technique for displaying larger images is to show a small version of an image (called an image thumbnail) that can be clicked to show the larger version. This technique (often used in e-commerce sites) prevents site visitors from having to download large versions of images they aren't interested in viewing.

Expression Web can automatically create image thumbnails for your images and also automatically creates links to the larger images. To create an image thumbnail, follow these steps:

1. Insert an image.
2. If the Pictures toolbar is not visible, select View, Toolbars, Pictures to display it.
3. Select the image and click the Auto Thumbnail button on the toolbar, as shown in Figure 9.13.

**tip**

You can also select an image and press Ctrl+T to create an auto thumbnail.

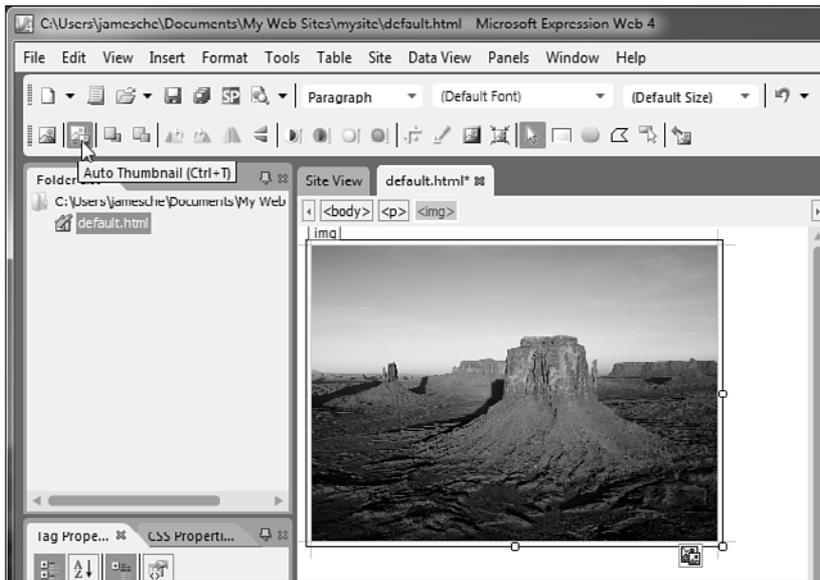


Figure 9.13
Creating a thumbnail image is as easy as clicking a button.

When you create an auto thumbnail, Expression Web resizes the image to be much smaller. It also adds a hyperlink that displays the larger version when you click it.

**tip**

To remove the blue border around image thumbnails, set the border to 0 pixels in the Page Editor Options dialog. Note that changes here affect only new thumbnails and not existing ones.



Larger Image Appears Too Small

If you've created an auto thumbnail, but when you click the link to see the larger image it shows up big for a split second and then shrinks to a smaller size automatically, what you are experiencing is likely caused by the Image Toolbar feature introduced in Internet Explorer 6. This feature automatically resizes images so they fit in the current window.

You can disable this feature for an entire page by adding the following META tag to the page:

```
<meta http-equiv="imagetoolbar" content="no" />
```

To disable the feature for a single image, use the `galleryimg` attribute. The following image will not use the Image Toolbar feature in Internet Explorer:

```

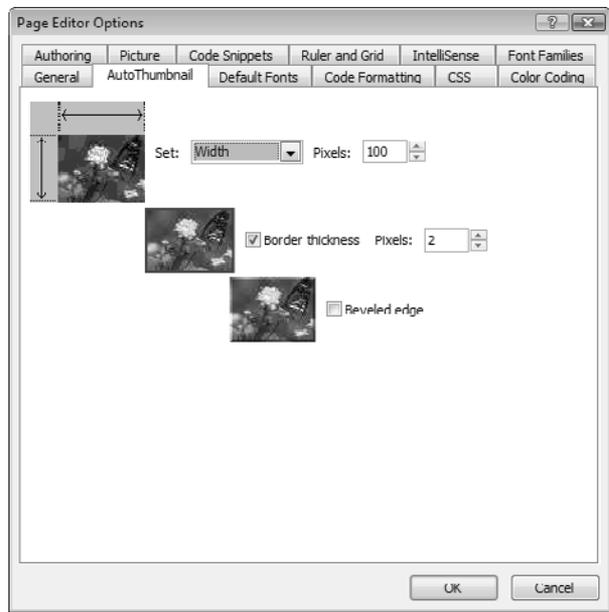
```

Configuring Auto Thumbnails

By default, Expression Web creates a thumbnail image that is 100 pixels wide. It also displays a blue border around the hyperlinked thumbnail. You can change these default settings using the AutoThumbnail tab on the Page Editor Options dialog, as shown in Figure 9.14.

Figure 9.14

You can configure how Expression Web creates auto thumbnails by using the Page Editor Options dialog.



Creating Image Maps

An image map is an image on which one or more regions are configured as hyperlinks. You can configure any number of hotspots on an image and configure a new hyperlink for each one.

To create an image map, follow these steps:

1. Insert an image and enable the Pictures toolbar.
2. Select the image, and then click one of the hotspot buttons on the Pictures toolbar. You can select a rectangular hotspot, a circular hotspot, or a polygonal hotspot, as shown in Figure 9.15.
3. Click and drag the image to create the hotspot. When creating a polygonal hotspot, click and drag to create each side until the polygon is closed on all sides.

When you release the mouse (or when a polygonal hotspot is closed on all sides), Expression Web displays the Insert Hyperlink dialog so you can configure the hyperlink for the hotspot. After the hyperlink has been created, you can edit it by right-clicking the hotspot and selecting **Hyperlink** from the menu or by double-clicking the hotspot.

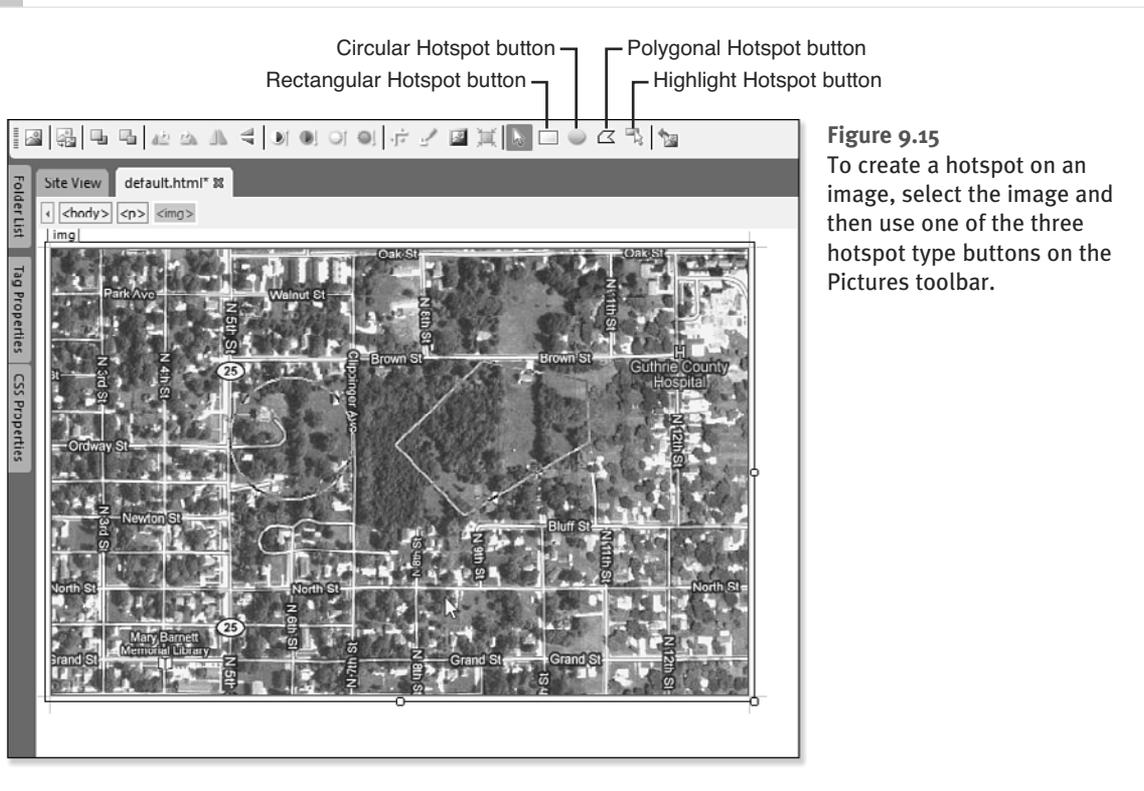


Figure 9.15

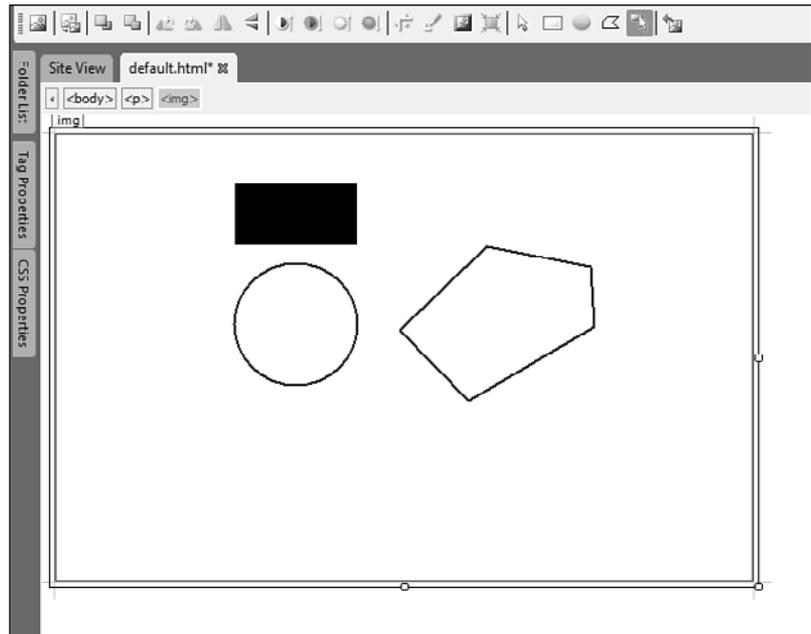
To create a hotspot on an image, select the image and then use one of the three hotspot type buttons on the Pictures toolbar.

➔ For information on using the Insert Hyperlink dialog, see Chapter 3, “Creating Pages and Basic Page Editing.”

After you add a hotspot, you can resize it by selecting it and dragging the sizing handles. You also can move it to a different position by clicking and dragging it to the new location. You can highlight all hotspots on an image by clicking the Highlight Hotspots button shown previously in Figure 9.15. Highlighting hotspots makes it easy to locate all hotspots on an image. When hotspots are highlighted, selected hotspots appear as filled shapes and unselected hotspots appear as outlined shapes, as shown in Figure 9.16.

Figure 9.16

Hotspots can be highlighted for easy identification. Selected hotspots are filled shapes, while unselected ones appear as outlines.



Inserting Multimedia

Web designers have always sought out ways to add more interactivity to sites. One way is to add multimedia elements to a site. In the early days, web developers were limited to animated graphics, a trend that quickly became overused and annoying. However, as technology has improved over the years, multimedia options have increased to the point where adding effective multimedia to your site is now easy.

Expression Web offers the capability to insert Flash movies, Microsoft Silverlight applications and video, Microsoft Deep Zoom images, and Windows Media content. If you're using HTML5, you can also use the `<video>` tag or the `<audio>` tag to add multimedia.

Inserting Flash Movies

To insert a Flash movie, select Insert, Media, Flash Movie. You'll be asked to browse for the *.swf file, as shown in Figure 9.17.

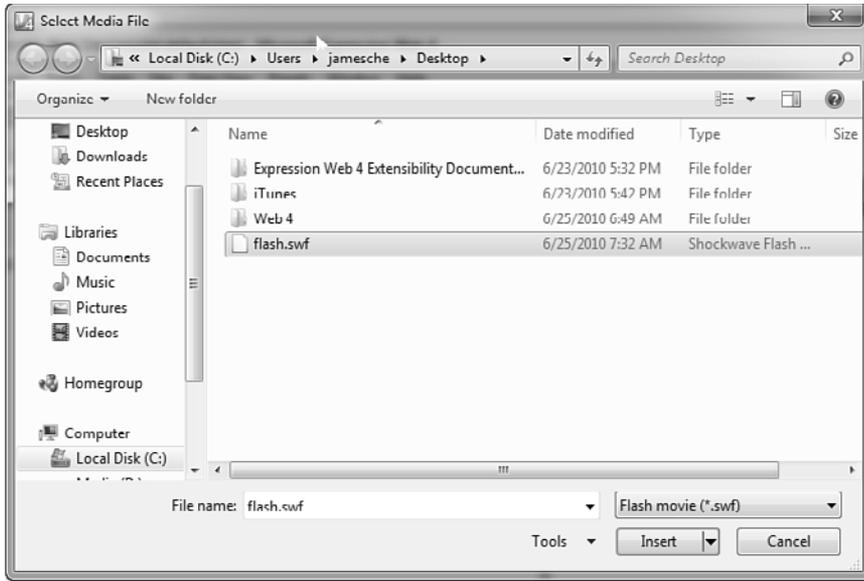


Figure 9.17
When inserting a Flash movie, you'll need to browse to the *.swf file that you want to insert.

After the Flash movie has been inserted, you can configure it by either right-clicking it and selecting Flash SWF Properties or by double-clicking it in Design View to access the Flash SWF Properties dialog. The Flash SWF Properties dialog contains two tabs, the Appearance tab (shown in Figure 9.18) and the General tab (shown in Figure 9.19).

You also might be familiar with the Flash Video (*.flv) format. Flash Video is high-quality compressed video that plays back in the Flash player inside your web browser.

Although Flash Video files can be played as standalone files, Expression Web does not allow for the insertion of Flash Video files without using Code View. You'll need to package your Flash Video inside a Flash SWF file to use it with Expression Web's Flash Movie feature. Many Flash authoring tools are available that allow you to do this, including Adobe's own Flash application, SWiSHzone.com's SWiSH Max, and many other applications.



tip

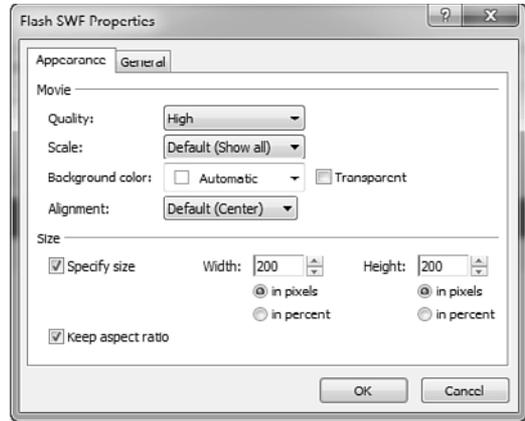
When you insert a Flash movie in Expression Web, the animation will not play in Design View by default. To play the animation in Design View, right-click the Flash movie and select Play Movie in Flash Format.

note

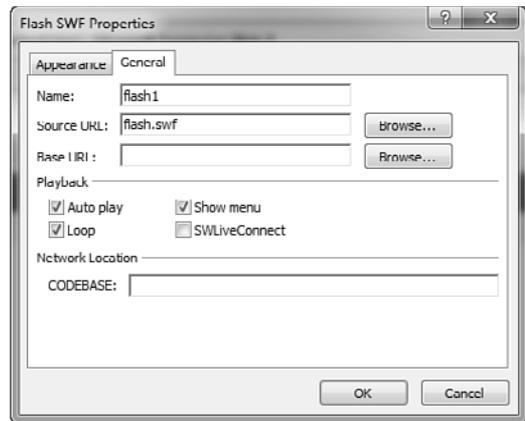
Changes in the Flash SWF Properties dialog alter the HTML code used to display the Flash movie. The properties and settings available are documented by Adobe and can be accessed at kb.adobe.com/selfservice/viewContent.do?externalId=tn_12701.

Figure 9.18

The Appearance tab in the Flash SWF Properties dialog allows for easy manipulation of properties that affect the playback of your Flash movie in the browser.

**Figure 9.19**

The General tab in the Flash SWF Properties dialog contains additional settings for the configuration of Flash movies.



Inserting Silverlight Applications

Microsoft Silverlight is a technology for building animated and interactive applications and for presenting high-quality video on the Web. Because Silverlight applications can be hosted on a page, they share some characteristics with Adobe Flash. However, Silverlight applications can be programmed using the Microsoft .NET Framework, so the capabilities of Silverlight are virtually limitless.

Silverlight applications are developed using a language called XAML (pronounced *zamu*), an acronym that stands for Extensible Application Markup Language. XAML is similar in syntax to XML and—like XML—can be written

note

A full discussion of Silverlight development is outside the scope of this book. If you're interested in reading more about Silverlight development, read *Essential Silverlight 3* from Addison-Wesley Professional.

using any text editor. However, the best tool for writing XAML and developing Silverlight applications is Microsoft's Expression Blend. Expression Blend is a design tool that allows you to create interactive Silverlight interfaces by dragging and dropping user interface elements in a graphical environment.

Expression Blend provides tools to add basic interactivity to your Silverlight application, but you can also use Visual Studio to add complex interactivity using C#, VB, or any other language that targets the .NET Framework.

To insert a Silverlight application, select Insert, Media, Silverlight. You are prompted to select a folder in which your Silverlight .xap file exists, as shown in Figure 9.20.

caution

Your Silverlight application relies on the folders and files in the folder that Expression Web imports into your site. Don't move or rename those files or folders; doing so will break your Silverlight application.

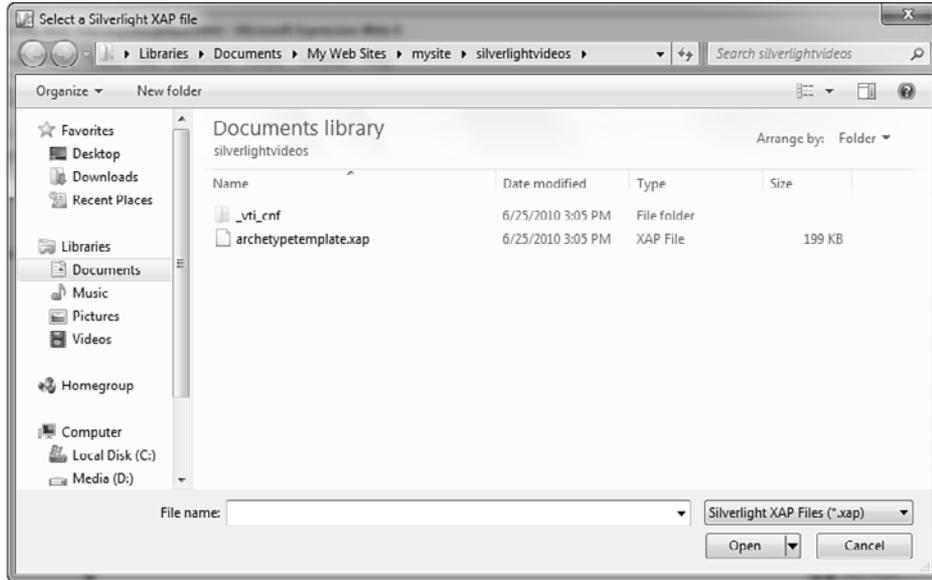


Figure 9.20 You'll need to point Expression Web to the folder where your Silverlight application resides.

Inserting Silverlight Video

Expression Web lets you easily add video to your site that plays using the Microsoft Silverlight 4 plug-in. When you use Expression Web to insert Silverlight video, the video you select is encoded automatically and adds the code necessary to play your video using Silverlight.

To insert Silverlight video, select Insert, Media, Silverlight Video and browse to the video file you want to include. After selecting the video file, Expression Web displays the Insert Silverlight Video dialog, as shown in Figure 9.21.

note

Visitors to your site need to have the Silverlight plug-in installed to view your videos. If Silverlight is not installed, a graphic is displayed that notifies the visitor to install it along with a link to the download from Microsoft's site.

 **note**

Inserting Silverlight video requires Expression Encoder. Expression Encoder ships with Expression Studio, but if you did not choose to install it when you installed Expression Studio, you need to add it to your installation before inserting Silverlight video using Expression Web.

The top part of the Insert Silverlight Video dialog shows the input and encoding options. You can select the encoding type for your video using the Encoding drop-down. Choose the appropriate encoding option for the bandwidth you expect your site visitors to have available.

The bottom portion of the dialog shows output options. You can select the output folder for the encoded video, the template for video controls, and various video options.

**tip**

For information on each of the options in the Insert Silverlight Video dialog, hover your mouse over each option and Expression Web displays a helpful tooltip.

 **tip**

The Silverlight animation that plays in the Insert Silverlight Video dialog is actually a Silverlight video playing using the template selected in the Template drop-down. If you want to see how the controls in a particular template operate, select the template and then hover your mouse over the bottom portion of the preview window. You'll see controls appear and you can interact with those controls just as you would on a web page.

After you've selected the desired options, click Encode. Expression Web encodes the video and inserts the necessary code into your page to display the video.

Inserting Deep Zoom Images

Microsoft Deep Zoom allows for the display of very large images by providing a means of panning and zooming in a smooth fashion. To see a great example of a Deep Zoom image, browse to memorabilia.hardrock.com.

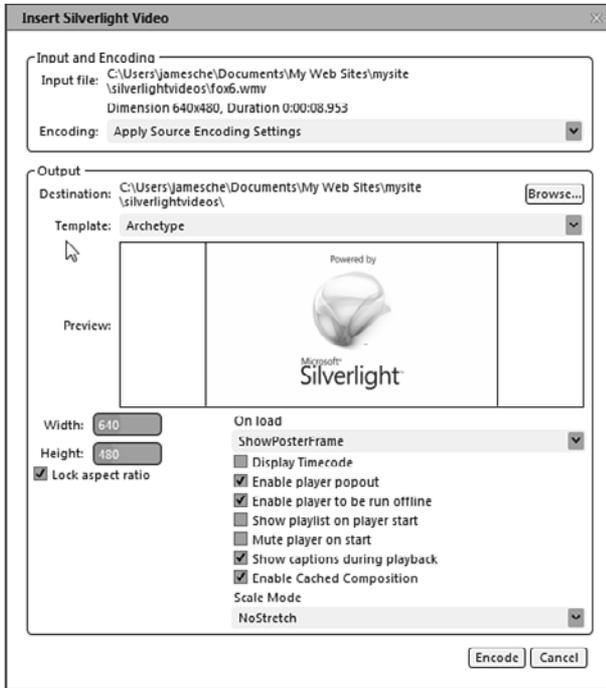
Deep Zoom projects are created using Deep Zoom Composer. You can download a copy of Deep Zoom Composer from Microsoft by searching for it at www.microsoft.com/downloads.

To insert a Deep Zoom image, select Insert, Media, Deep Zoom and select the Deep Zoom output file from your Deep Zoom project. When you do, Expression Web displays the Insert Deep Zoom dialog shown in Figure 9.22.

Expression Web can insert Deep Zoom images that use Silverlight 3, Silverlight 4, or Microsoft Seadragon Ajax. Seadragon allows web designers to use Deep Zoom in his or her site without using Silverlight. If you select the Autodetect option, Expression Web uses Silverlight if it's available and Seadragon Ajax otherwise.

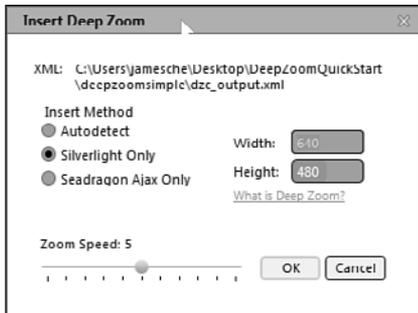
**tip**

For more information on Seadragon Ajax, see livelabs.com/seadragon-ajax.

**Figure 9.21**

The Insert Silverlight Video dialog provides all the settings necessary to customize the appearance and quality of your Silverlight video.

The Insert Deep Zoom dialog also provides controls for resizing your Deep Zoom image and for controlling the speed of the zooming feature of Deep Zoom.

**Figure 9.22**

The Insert Deep Zoom dialog provides settings for including Deep Zoom images on your page.

Inserting Windows Media

Expression Web can insert both Windows Media video and Windows Media audio. The formats that are supported depend on the capabilities of the user's machine as they are viewing your site. For example, if you insert a video file encoded in DivX format, a user can view the video only if DivX is installed on his or her machine.

To insert Windows Media into your page, select Insert, Media, Windows Media Player, and then browse to the desired media file, just as you did with the Flash movie. After you've selected the file, click Open to add the Windows Media Player to your page.

After you've added Windows Media to your page, you can modify the properties of the player by either double-clicking it or by right-clicking and selecting ActiveX Control Properties from the menu to display the Windows Media Player Properties dialog.

The four tabs in the Windows Media Player Properties dialog are explained in the sections that follow.

The General Tab

The General tab (shown in Figure 9.23) enables you to configure general settings for Windows Media.

The General tab contains the following settings:

- **File Name or URL**—The filename or the URL to display in Windows Media Player.
- **Select a Mode**—This drop-down controls the appearance of the Windows Media controls. Expression Web displays a description of the selected setting under the drop-down.
- **Auto Start**—When checked (the default setting), the Windows Media file starts playing automatically when the page is loaded.
- **Stretch to Fit**—When checked, the video displayed in Windows Media Player is stretched to fit the video window. This check box is unchecked by default.
- **Play Full Screen**—When checked, the player opens in full-screen mode.
- **Play Count**—By default, Windows Media plays once and then stops. You can increase the play count to increase the number of times your media plays.
- **Volume Settings**—The Volume Settings section provides controls for adjusting the audio of your Windows Media.



tip

If Windows Media Player cannot find the correct software (called a *codec*) to play back the format you are using, the user is asked if she wants Windows Media Player to try to install the correct codec.

Users of Mac OS (Apple users) might have problems viewing Windows Media files even with the correct codecs installed. Because of that, you might cause problems for some of your visitors when you choose Windows Media format.



tip

Expression Web uses the file-name parameter when adding Windows Media; therefore, the File Name or URL text box will be blank when Expression Web adds Windows Media.

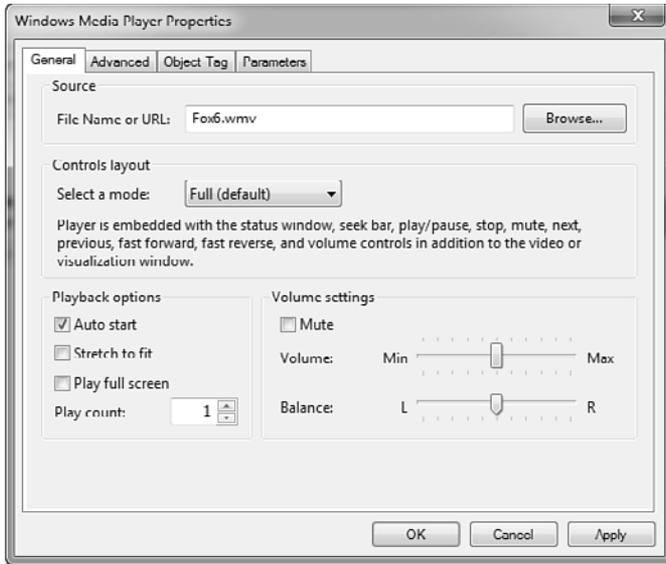


Figure 9.23
The General tab contains general settings for Windows Media.

The Advanced Tab

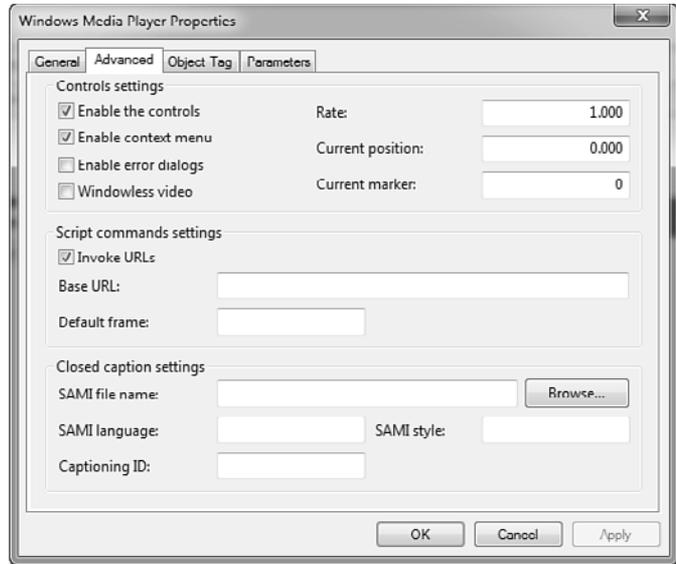
The Advanced tab is shown in Figure 9.24 and contains some of the advanced settings you can configure for Windows Media.

The following settings are available on the Advanced tab:

- **Enable the Controls**—When this box is checked (the default setting), the controls in the Windows Media Player are enabled and can be clicked. When it's unchecked, the controls cannot be clicked.
- **Enable Context Menu**—When checked (the default setting), a contextual menu is displayed when you right-click the Windows Media Player in the web browser.
- **Enable Error Dialogs**—When checked, any errors that occur during playback are displayed in a pop-up dialog. This box is unchecked by default.
- **Windowless Video**—When checked, video is rendered by the Windows Media Player directly inside the client window so that other elements (such as effects or text) can be rendered on top of the video. Checking this option can cause reduced performance. It is unchecked by default.
- **Rate**—Controls the playback rate of the Windows Media. The default value is 1.
- **Current Position**—Controls the position (in seconds) at which the Windows Media begins playing. The default is 0.000, meaning the Windows Media begins playing at the beginning.
- **Current Marker**—Controls the marker at which the Windows Media begins playing. If the Windows Media was not encoded with markers, setting this value to anything other than zero (the default) results in an error.

Figure 9.24

The Advanced tab contains some of the more advanced Windows Media settings.



- **Script Command Settings**—Windows Media files can contain URLs, and if the Invoke URLs check box is checked, those URLs open inside the default browser. The Base URL text box controls the base URL the Windows Media Player will use, and the Default Frame controls the frame that is used for the new window.
- **Closed Caption Settings**—Windows Media Player can use Microsoft's Synchronized Accessible Media Interchange (SAMI) technology to provide closed captions for Windows Media. The Closed Caption Settings section provides settings for implementing this feature.

note

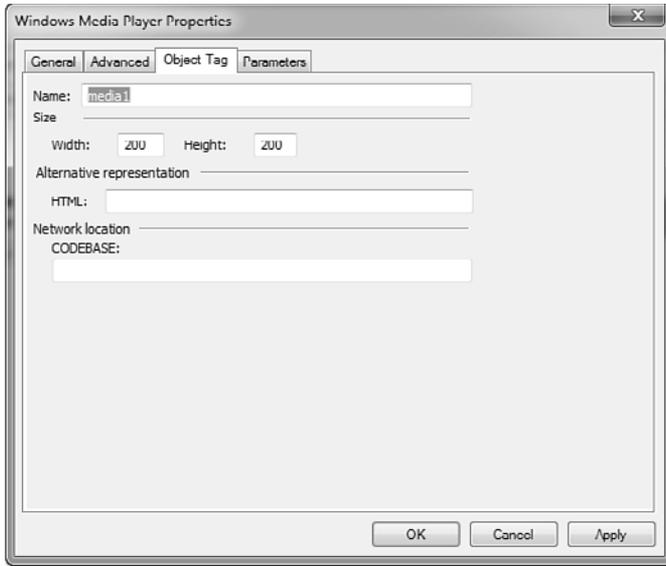
A discussion of implementing closed captions in Windows Media is outside the scope of this book. For more information on SAMI, see msdn2.microsoft.com/en-us/library/ms971327.aspx.

The Object Tag Tab

When Windows Media is inserted, Expression Web uses the `<object>` tag to configure it on your page. The Object Tag tab (shown in Figure 9.25) provides settings for the `<object>` tag.

The following settings appear on the Object Tag tab:

- **Name**—Specifies the name for the Windows Media Player in your page. The name can be used in a script to refer to the Windows Media Player.

**Figure 9.25**

The Object Tag tab contains settings that affect the `<object>` tag used to add Windows Media to your page.

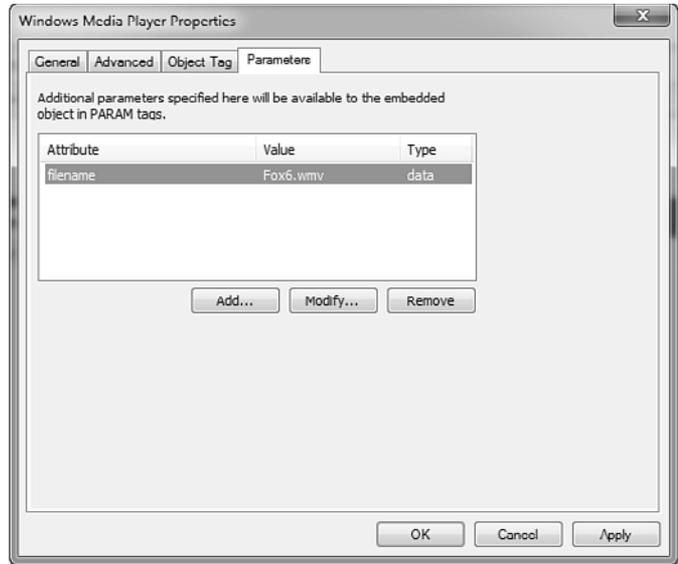
- **Size**—Specifies the size of the Windows Media Player in the page.
- **HTML**—HTML that is entered into this text box is displayed when someone without the Windows Media Player installed visits your page.
- **CODEBASE**—The CODEBASE setting can contain a URL that links to a location where the browser can download the Windows Media Player control, if it's not already installed.

The Parameters Tab

The Parameters tab (shown in Figure 9.26) allows you to add, modify, or remove parameters of the `<object>` tag. As you make modifications in the other tabs, Expression Web adds parameters to the `<object>` tag on the page. Those settings appear in the Parameters tab.

Figure 9.26

The settings you configure on other tabs appear as parameters on the Parameters tab.



Importing Adobe Photoshop Files

Adobe Photoshop is one of the most commonly used graphic design packages for sites, and Expression Web offers Photoshop users some convenient integration features.

By selecting Insert, Picture, From Adobe Photoshop (.psd), you can use any of your Photoshop files in your page without importing the original .psd file into your site. After you select the .psd file from your hard drive or your network, Expression Web displays the Import Adobe Photoshop dialog as shown in Figure 9.27. By default, Expression Web includes all layers in your .psd file, but you can remove the check mark from any layer that you don't want to include.

Expression Web also gives you the choice of importing the compatibility image that Photoshop saves with your .psd file. The compatibility image is an image that includes all layers marked as visible within Photoshop.

After you've chosen the layers you want to include, choose your encoding type (PNG, JPEG, or GIF) and click OK. Expression Web displays the Save dialog so you can save the imported image into your site.

Expression Web maintains the connection between the imported image on your page and the original Photoshop file. If you update the original .psd, and you want the image on your page to be updated with those changes, right-click the image on your page and select Adobe Photoshop (.psd), Update from Source from the menu. Expression Web displays the Import Adobe Photoshop dialog



tip

By right-clicking the image and selecting Adobe Photoshop (.psd), Edit Source, you can launch Photoshop from within Expression Web so that you can quickly edit the .psd file.

as shown in Figure 9.27 so that you can import the modified .psd file into your page and replace the existing image.

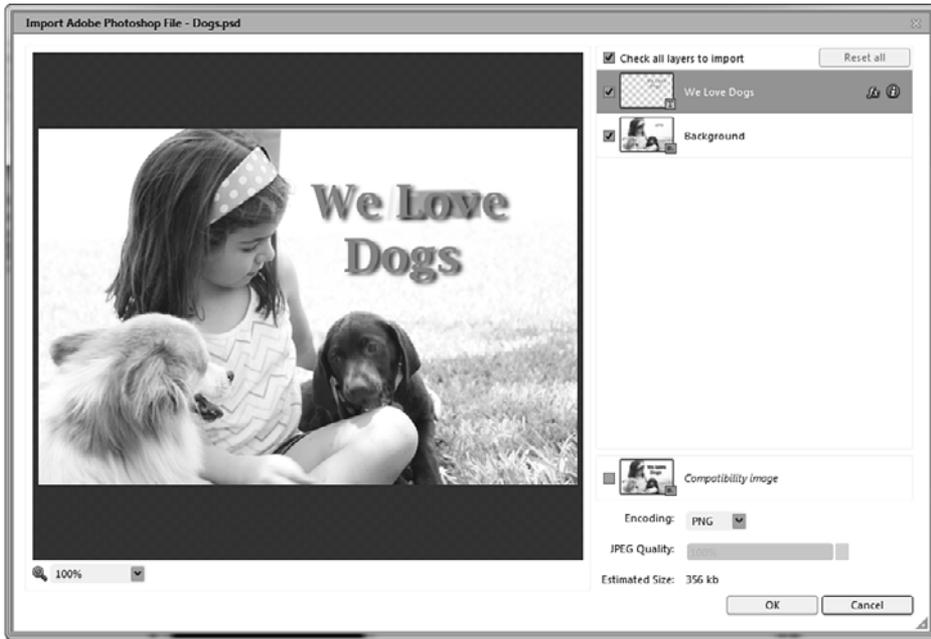


Figure 9.27 The Adobe Photoshop Import feature in Expression Web is an easy way to include your Photoshop creations in your page.

Page Transitions

Page transitions were introduced in Internet Explorer 5.5. They are effects that display when moving from page to page or site to site and are built into the browser. You can easily add page transition effects with Expression Web.

To add a page transition effect, select Format, Page Transition to display the Page Transitions dialog shown in Figure 9.28.



caution

Page transitions work only in Internet Explorer. They will not adversely affect other browsers, but they won't be visible.

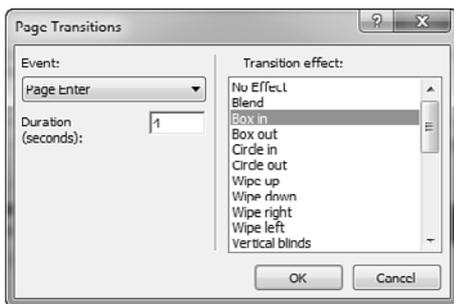


Figure 9.28 Page transitions can be configured easily using the Page Transitions dialog.



Page Transitions Don't Work

If you've added page transitions but nothing happens when you view your site in a web browser, first make sure you are using Internet Explorer 5.5 or greater. Page transitions won't work in other browsers. If you meet that general requirement, chances are that page transitions are turned off in your browser.

To enable page transitions in Internet Explorer, select Tools, Internet Options and click the Advanced tab. In the Browsing section, make sure Enable Page Transitions is checked and click OK.

Page transitions are met with mixed feelings by many site visitors. Depending on the speed of the computer on which the site is being browsed, a transition might take much longer to complete than the time for which it's configured. For example, my wife uses an old, slow 500MHz machine to browse the Internet. Page transitions on that machine regularly take twice as long as they do on a faster machine. A nearly 10-second transition is far too long, and it can be annoying when visiting sites that use transitions for all pages.

Before you add an effect such as page transitions, give careful consideration to how it might affect people visiting your site.

Adding Audio and Video with HTML5

Adding multimedia with HTML5 is easier than ever thanks to the new `<audio>` and `<video>` tags. Browsers that support HTML5 features have built-in support for audio and video, so it's not necessary to use a browser add-on or control to play multimedia files.

The following code sample demonstrates how the HTML5 video tag works.

```
<video id="MyVideo" controls src="MyVideo.mp4"></video>
```

That looks pretty simple, and it is. However, it's not as simple as it might seem at first.

Unfortunately, not all browsers support the same type of audio and video files. For example, H.264 is one of the most common types of video used on the web these days, but neither Firefox nor Opera supports H.264 video files with the `<video>` tag. Audio files bring about that same type of frustration; most browsers support MP3 audio files, but Firefox and Opera don't.

To account for browser incompatibilities, HTML5 allows you to specify multiple source files for audio and video. As long as one of the sources listed is supported by your browser, everything will work fine. The following code sample illustrates how to use multiple source files.

```
<audio id="MyAudio">
  <source src="mysong.mp3" type="audio/mp3" />
  <source src="mysong.ogg" type="audio/ogg" />
  <p>Sorry. Your browser does not support HTML5 audio.</p>
</audio>
```

When a browser encounters this code, it reads the first source file. If the browser supports MP3, it uses `mysong.mp3` as the source file. If the browser doesn't support MP3, it moves to the next file and attempts to use it. If it doesn't support that file type either, a message is displayed notifying the user that the browser doesn't support HTML5 audio.

You can find more information on using the `<video>` and `<audio>` tags at <http://msdn.microsoft.com/en-us/magazine/hh781023.aspx>.

USING FIND AND REPLACE

An Introduction to Find and Replace

Anyone who uses a computer should be familiar with tools that find and replace text. Such tools are typically rudimentary, but a find and replace tool for a web design application has stiffer requirements. Not only does it need to be able to locate and replace text in the body of pages, but it also must be able to locate and replace text that appears in code. Most importantly, it must be able to distinguish between the two. The word “button,” for example, has an entirely different connotation when used in the body of a page than when used in code. If you wanted to replace all HTML buttons with image buttons, you wouldn’t want to also replace the phrase *button-down shirt* with the phrase *image-down shirt*.

The find and replace tool in Expression Web is well-suited to performing intelligent find and replace for sites. It offers features specifically designed for searching HTML code, and it provides a powerful search capability utilizing regular expressions. Regular expressions (sometimes referred to as *regex*) are specialized search strings that match patterns in text or code. As you’ll see in this chapter, the use of regular expressions adds powerful search capabilities that would not be possible otherwise.



note

Regular expressions comprise a language that can be difficult to master. A full discussion is outside the scope of this book. If you’re interested in learning regular expressions, I highly recommend that you read *Sams Teach Yourself Regular Expressions in 10 Minutes* from Sams Publishing.

Finding and Replacing Text

Finding and replacing a simple word or phrase is a straightforward endeavor. In web design, however, things are rarely simple. Suppose you have designed a site that contains hundreds of pages with Social Security numbers (SSNs) on them. Because of new requirements in your company, you are charged with the task of reformatting these SSNs. You need to keep the last four digits and replace the rest of each SSN with asterisks. A simple find and replace just won't suffice, but a regular expression is perfect for such a job.



note

The actual structure of an SSN is more restrictive than the regular expression used here. To keep the regular expression example less complex, I opted for a simpler pattern.

Using Regular Expressions

The most efficient way of working with regular expressions is to separate your search into parts. When looking for an SSN, you need to find three numbers followed by a dash, two numbers followed by another dash, and then four numbers. Additionally, the first digit of the series must be between 0 and 7.

The final regular expression looks like this:

```
[0-7][0-9]^2\[0-9]^2\[0-9]^4
```

If you're unfamiliar with regular expressions, this example might appear to have a complex syntax, but it's actually simple. Let's break it apart so you'll fully understand how the language works.

The first set of numbers you want to match consists of three digits and is known as the *area number*. The first digit must be between 0 and 7 because the Social Security Administration has never issued an SSN with an area number higher than 728.

The regular expression to match this pattern is as follows:

```
[0-7][0-9]^2
```

The first part of this expression, `[0-7]`, indicates that any single digit between 0 and 7 produces a match. The second part of the expression, `[0-9]^2`, indicates that any two digits between 0 and 9 produces a match. The `^` character is the repeat expression character, and it is followed by the number of times the preceding expression should be repeated.

The middle set of numbers you need to match consists of two digits between 0 and 9 and is known as the *group number*. The syntax for the regular expression is `[0-9]^2`. This syntax should now be familiar to you. It means that you want to match a character between 0 and 9 and then repeat that expression two times.

The last set of numbers you want to match consists of four digits between 0 and 9 and is called the *serial number*. The syntax for the regular expression is `[0-9]^4`. The curly braces surrounding this portion of the regular expression are explained in the "Replacing Text" section later in this chapter.

Between each set of numbers is a dash character. A dash is a special kind of character because not only can you write a regular expression to look for a dash in some text, but it is also used in regular expression syntax. (In the regular expression we're using in this chapter, the dash is used to

indicate a range of digits.) To actually find an explicit dash in text, you need to specify that you are looking for an actual dash and not using it as part of the regular expression.

The `\` character, called the *escape character*, does just that. By preceding a character with the `\`, you are telling Expression Web that you want to match that character. Therefore, the regular expression `\-` matches a dash character in text.

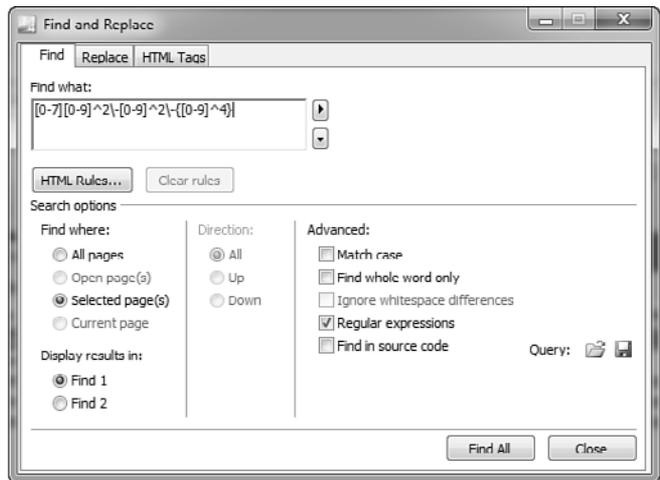
Finding Text

The Find and Replace dialog (shown in Figure 10.1) is made up of three tabs:

- **Find**—Provides tools for locating text within one or more pages
- **Replace**—Provides tools for locating and replacing text within one or more pages
- **HTML Tags**—Provides tools for locating and replacing HTML code

Figure 10.1

The Find and Replace dialog provides all the tools you need to locate and replace both text and HTML code.



To find specific text in one or more pages, open the Find and Replace dialog by selecting Edit, Find. Enter the text you want to search for in the Find What text box, select the desired options, and click Find All to display the search results. You can choose to search from the insertion point up, from the insertion point down, or in all directions by selecting the desired radio button in the Direction section.



tip

Some of the options in the Find Where section might be disabled based on what you currently have open in Expression Web.



Cannot Select Search Direction

If you're searching from the insertion point down, but the radio buttons in the Direction section are all disabled and you can't change the setting, make sure a page is open and that Current Page is selected in the Find Where section. Setting search direction is valid only when you are searching within the current page.

You can specify where to search for the text entered using the radio buttons in the Find Where section of the Find and Replace dialog. The following options are available:

- **All Pages**—Searches for the specified text in all pages in the current site.
- **Open Page(s)**—Searches for the specified text in all open pages.
- **Selected Page(s)**—Searches for the specified text in all selected pages. Pages can be selected in the Folder List panel or in Folders View.
- **Current Page**—Searches for the specified text in the current page only.

➔ *For more information on using the Folder List panel and Folders View, see Chapter 1, “An Overview of Expression Web 4.”*

You can also specify additional options for searching in the Advanced section. Check the Regular Expression check box if the text you have entered is a regular expression. If you want to search source code for the text you have entered, check the Find in Source Code check box. The other options should be self-explanatory.



Find in Source Code Disabled

If you have a page open and Current Page is selected in the Find Where section, the Find in Source Code check box is disabled. If the page is open in Design View, searches are performed on text inside the page. If the page is open in Code View, searches are performed on source code.



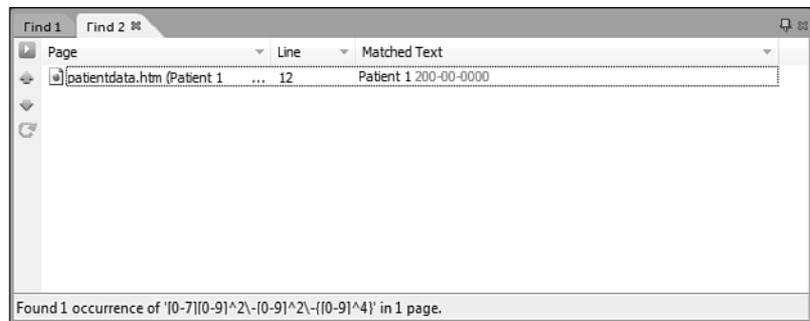
Source Code Searches Don't Work

Suppose you have a page open in Code View. You want to search in source code, and you have Current Page selected; however, even though you're entering HTML code that should be found, no search results are produced. Chances are you have an HTML rule applied to your search. Find in Source Code is always disabled and deactivated when HTML rules are applied. Clear the rule by clicking the Clear Rules button if you want to search in HTML code.

When you click Find All, Expression Web displays the results in the Find 1 panel by default. You can display the results in a second panel (the Find 2 panel) by selecting the Find 2 radio button in the Display Results In section. Figure 10.2 shows the results of a search for SSNs using the regular expression shown in Figure 10.1. The results are displayed in the Find 2 panel, but previous search results in the Find 1 panel can be recalled easily by clicking the Find 1 tab to display the Find 1 panel. This is useful when you want to start a new search but are not yet finished working on results from a previous search.

Figure 10.2

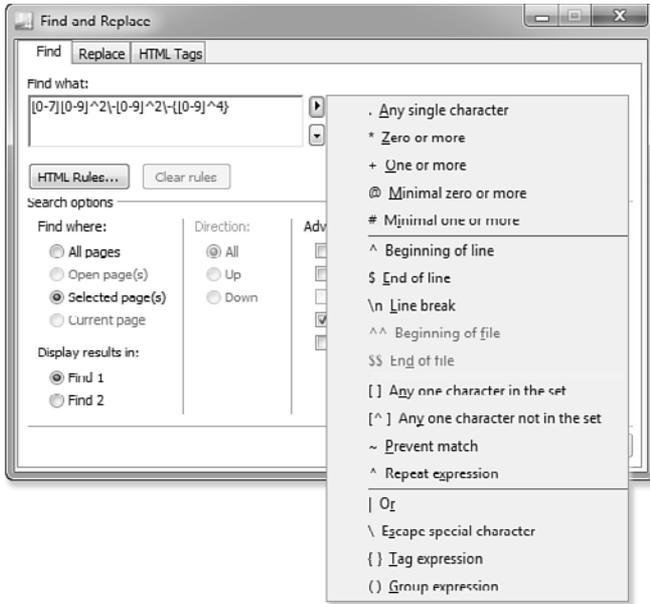
The results of a regular expression search for SSNs are displayed in the Find 2 panel. The ability to use two panels for search results allows you to easily work with two different result sets.



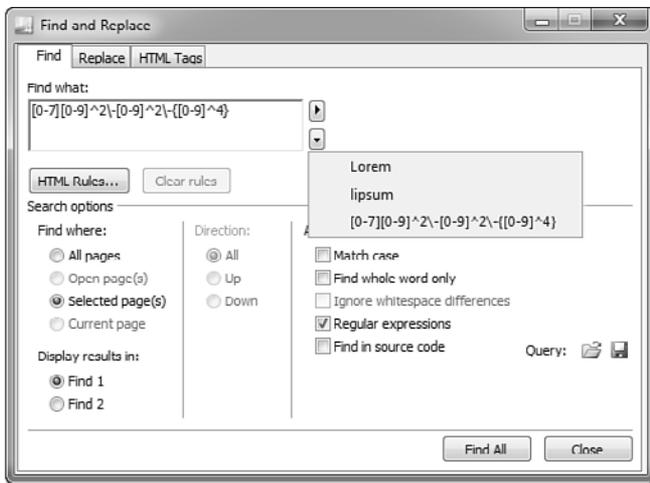
If you need assistance entering a regular expression search, click the right-pointing arrow button to the right of the Find What box, as shown in Figure 10.3. You can easily build simple regular expressions using this method.

Complex regular expressions will likely require manual entry instead of using the Regular Expressions button. Fortunately, Expression Web keeps a list of recently used searches so you can easily recall a complex search later. By clicking the downward-pointing button to the right of the Find What box, you can access a list of previously entered searches, as shown in Figure 10.4.

➡ *For an even better way of saving complex searches, see “Saving Queries,” p. 182.*

**Figure 10.3**

The right-pointing arrow button can make creating regular expression searches fast and easy, but don't expect to find advanced regular expression syntax here.

**Figure 10.4**

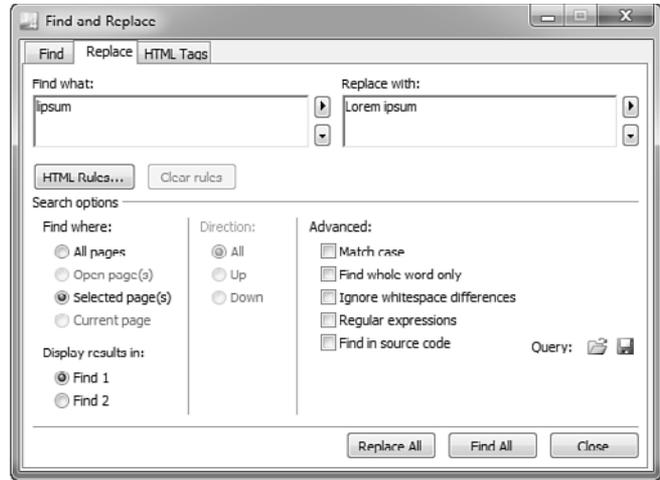
Expression Web maintains a list of recent searches you can recall with the click of a button.

Replacing Text

To replace text, open the Find and Replace dialog by selecting Edit, Replace. If the Find and Replace dialog is already open, you can simply click the Replace tab. Enter the text you want to search for in the Find What text box and the text that should replace it in the Replace With text box, as shown in Figure 10.5.

Figure 10.5

The Replace tab allows you to locate text and replace it with the text you specify.



You can also use regular expressions when replacing text. Remember that the requirements for our Social Security example are that all SSNs should be reformatted so that only the last four digits of the number are displayed. All other digits should be replaced with asterisks. Without regular expressions, you wouldn't be able to perform such a complex replace, but with regular expressions, it's fairly straightforward.

We've already covered the regular expression used to locate SSNs. Here it is again:

```
[0-7][0-9]^2\-[0-9]^2\-[0-9]^4}
```

I've already explained everything in the regular expression with the exception of the curly braces. Curly braces in a regular expression enable you to store the result of the expression inside the braces so it can be used later. An expression inside curly braces is called a *tagged expression*. You can have any number of tagged expressions in a regular expression.

Tagged expressions are used when replacing text using regular expressions. Let's look at a specific example using the SSN replacement we're performing. Consider the following SSN:

```
232-00-2323
```

When our regular expression locates this SSN, it should replace it with the following:

```
***-**-2323
```

Replacing the numbers with asterisks is simple, but you also need to leave the last four digits as they are. You could just change the regular expression so that it located all patterns of ###-## and just replaced them with asterisks. However, some instances of that pattern might not be SSNs. For example, suppose the pages also contain employee numbers and are in the format ###-##. In that case, you would be replacing the employee numbers with asterisks, and that's not what you want.

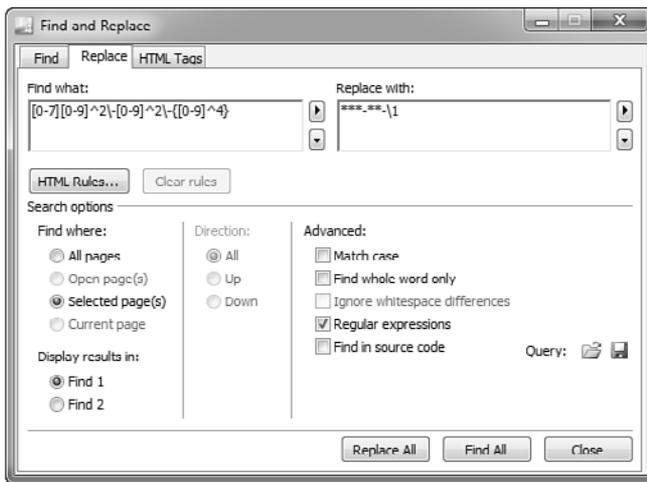
Tagged expressions are the perfect solution to this problem. By using tagged expressions, you can use your regular expression as-is and easily perform the replace operation that is required.

Figure 10.6 shows the Find and Replace dialog ready to perform the SSN replacement. The regular expression in the Replace With text box shows the asterisks that will be used in place of the first five numbers in the SSN. The expression `\1` that appears in place of the last four digits will be replaced with the result of the first tagged expression.

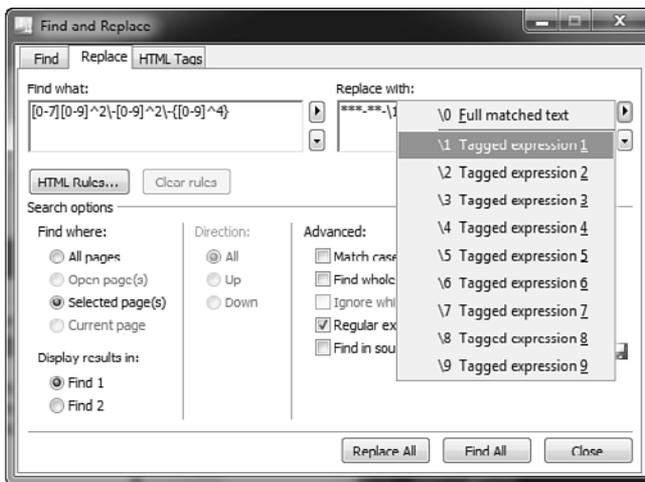
The right-pointing arrow button next to the Replace With text box (see Figure 10.7) provides easy access to tagged expressions. Simply click the right-pointing arrow button and then select the desired tagged expression to have it inserted into your regular expression.

**tip**

As previously mentioned, you can have any number of tagged expressions. The first one is `\1`, the second is `\2`, and so on. An additional tagged expression, `\0`, can be used to refer to the entire string that was matched by a regular expression.

**Figure 10.6**

By using tagged expressions, portions of matched text can be stored for use when replacing text.

**Figure 10.7**

Insert tagged expressions by clicking the Regular Expressions button.

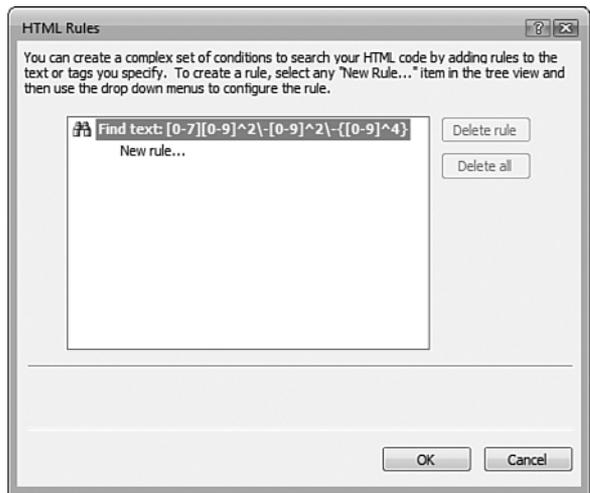
Using HTML Rules in Find and Replace

Let's add another level of complexity to the SSN search. Suppose there are many places where SSN-type patterns are presented in the site, but you only want to find and replace the ones that are actual SSNs. All the actual numbers appear inside `<div>` tags and are styled with a CSS class called `empData`. What may at first seem like an impossible task is actually made easy using HTML rules.

➔ *For more information on CSS styles, see Chapter 17, "Creating Style Sheets."*

To access HTML rules, click the HTML Rules button in the Find and Replace dialog. When you do, the HTML Rules dialog is displayed, as shown in Figure 10.8.

Figure 10.8
The HTML Rules feature adds impressive power to your searches.



Let's configure an HTML rule to replace the SSN found in the sample site in the `Examples\Ch10\Files\Website` folder on this book's website at informit.com:

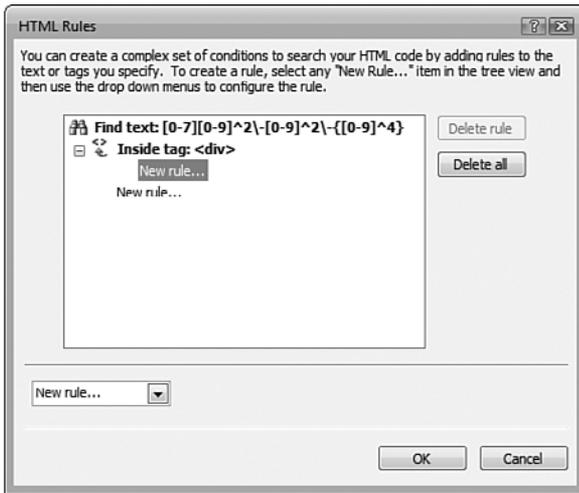
1. Open the site in Expression Web.
2. Open the `default.htm` page.
3. Select `Edit, Replace` to display the Find and Replace dialog.
4. In the Find What text box, enter the following regular expression:

```
[0-7][0-9]^2\[0-9]^2\-[0-9]^4}
```

5. In the Replace With text box, enter the following regular expression:

```
***-**-\1
```

6. Click the HTML Rules button.
7. Select New Rule in the dialog.
8. From the New Rule drop-down, select Inside Tag.
9. From the tag drop-down, select div.
10. Select New Rule directly under the Inside Tag rule, as shown in Figure 10.9.

**Figure 10.9**

You can create nested rules to search for specific patterns within a specified tag. In this case, you apply a new rule to the div tag search.

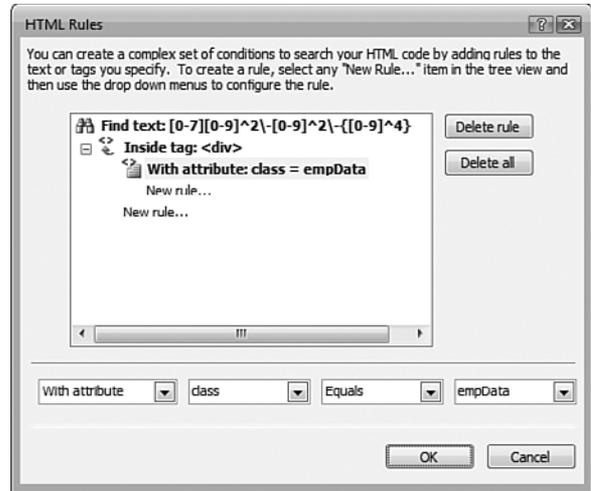
11. From the New Rule drop-down, select With Attribute.
12. From the attribute drop-down, select Class.
13. Click inside the [any value] drop-down and enter empData, as shown in Figure 10.10
14. Click OK to add the new HTML rule.

After the HTML rule has been added, the text of the HTML rule will be displayed next to the Clear Rules button, as shown in Figure 10.11.

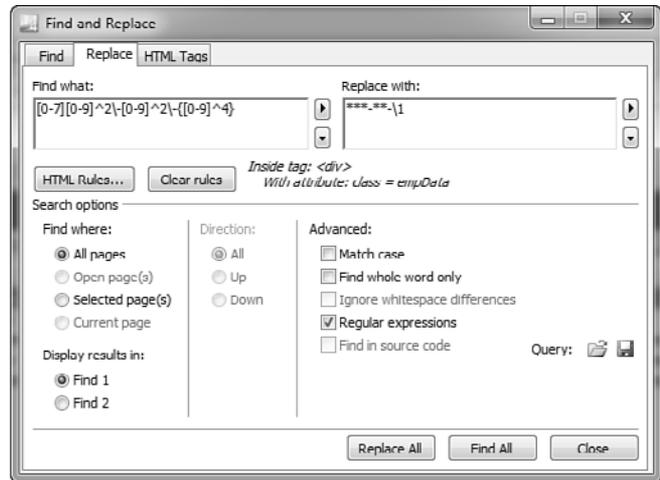
Make sure that the All Pages radio button is selected and click Replace All to perform the replacement.

Figure 10.10

Powerful searches can be created by nesting rules. In this case, you want to find only text in a `<div>` with a CSS class attribute of `empData`.

**Figure 10.11**

Applied HTML rules are displayed in the Find and Replace dialog. To clear a rule, click the Clear Rules button.



Finding and Replacing HTML Tags

In addition to searching for and replacing text within a page, you can search and replace or modify HTML tags using the Find and Replace dialog.

To find and replace HTML tags, select Edit, Find or Edit, Replace and click the HTML Tags tab. Select the desired tag in the Find Tag drop-down and select the replace action from the Replace Action drop-down if desired. Depending on what you select in the Replace Action drop-down, other options are made available.

Let's change the `empData` class in `default.htm` to `employeeData`. Open the site and open `default.htm`. Complete the following steps to replace the `class` attribute's value:

1. Select Edit, Replace.
2. Click the HTML Tags tab.
3. From the Find Tag drop-down, select `div`.
4. In the Replace Action drop-down, select Set Attribute Value.
5. In the Attribute drop-down, select `class`.
6. In the To drop-down, enter `employeeData`, as shown in Figure 10.12.

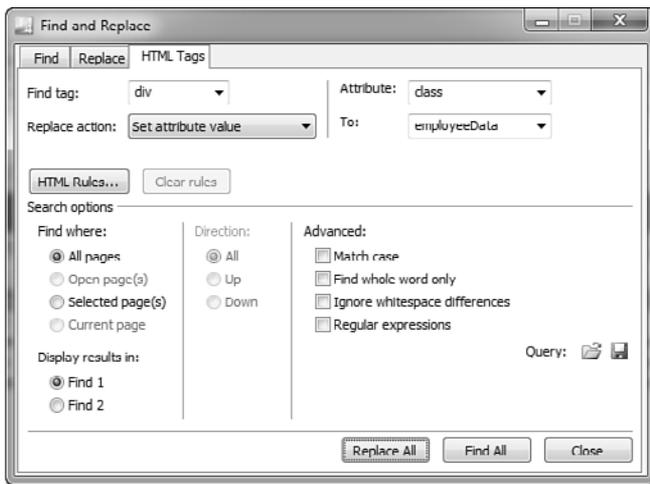


Figure 10.12

Searching for and modifying HTML tags is another powerful feature of the Find and Replace tool in Expression Web.

7. Click Replace All to replace the `class` attribute.

Switch to Code View and look at the `<div>` tag. Notice that the `empData` CSS class has been changed to `employeeData`.

Saving Queries

The SSN example used in this chapter is a fairly simple one. Much more complex searches are possible, and if you add a series of HTML rules to the mix, configuring a search can be time-consuming. That's not a big deal if your search is a one-time thing, but if you find yourself in the position of having to perform a particular search more than once, being able to save the search for later use can save a lot of time.



tip

HTML rules can also be applied when finding and replacing HTML tags for even more powerful searches.

Expression Web can save searches you perform using the Query buttons in the Find and Replace dialog. Click the Save Query button to save a query and the Open Query button to open a previously saved query.

Expression Web queries are saved as .fpq files and are in XML format. The following code shows the XML for the SSN example used in this chapter:

```
<?xml version="1.0" encoding="utf-8" ?>
<fpquery version="1.0">
  <queryparams />
  <find text="[0-7][0-9]^2\[0-9]^2\-[0-9]^4">
    <rule type="insideTag" tag="div">
      <rule type="attribute" attribute="class" compare="=" value="empData" />
    </rule>
  </find>
  <replace text="***.***\1" />
</fpquery>
```

**tip**

Expression Web queries are saved in the AppData\Roaming\Microsoft\Expression\Web 4\Queries folder in the current user's profile path by default.

Editing and Removing Recent Searches

As mentioned previously in this chapter, Expression Web saves a list of previously used searches so you can access them by clicking the Most Recently Used button, shown earlier in Figure 10.4. Expression Web saves a long list of recent searches, and it's likely that you will want to clear them out or edit them from time to time. Expression Web doesn't provide any user interface element for editing or removing recent searches, but you can do it manually by editing the Registry.

Most recently used searches are stored in the following Registry key:

HKEY_CURRENT_USER\Software\Microsoft\Expression\Web\4.0\FindMRUTo make changes or remove recent searches, make sure Expression Web is not running; then perform the following steps:

1. Open a command prompt.
2. From the command prompt, type **regedit** and press Enter.
3. Expand the HKEY_CURRENT_USER section.
4. Expand the Software section under HKEY_CURRENT_USER.
5. Expand the Microsoft section under Software.

**caution**

Be careful when editing the Registry. If you make modifications without knowing what you're doing, you can cause serious problems with your applications or with Windows itself.

6. Expand the Expression section under Microsoft.
7. Click the Web folder once under Microsoft to select it.
8. Select File, Export.
9. Enter **ewd_backup.reg** for the filename and save the file to your Desktop or another accessible area. This creates a backup of the Web Designer section of the Registry.
10. Expand the Web section.
11. Expand the 4.0 section under Web.
12. Click the FindMRU folder once to select it.

You should see Registry keys similar to the ones in Figure 10.13. Note that the names of these keys are in incremental order beginning at 0. To remove all entries from the recent list of searches, select all keys with a numeric value and press Delete to delete them.

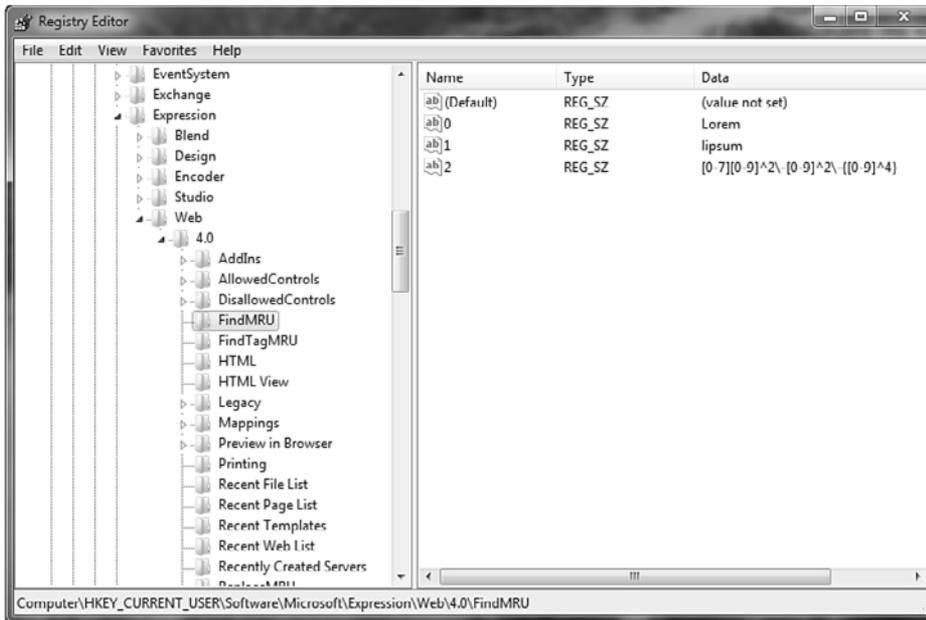
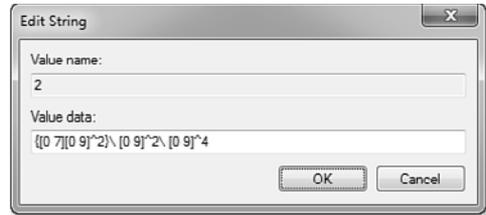


Figure 10.13
Recent searches are stored in the Registry. You can remove or edit them, but be careful and back up the key first!

If you'd like to edit an entry, double-click the entry and modify it as needed. Click OK to commit the change, as shown in Figure 10.14.

If you want to remove one or more entries but not all of them, make sure that any remaining entries are in numerical order beginning at 0. To renumber a specific key, select it and then press F2. Enter the new name and press Enter to commit it.

Figure 10.14
Double-click the name of a specific key to edit it.



If you make any errors while editing the Registry, close the Registry Editor and double-click the .reg file you saved before you began editing the Registry. Click Yes when asked if you want to add the information to the Registry.

As a final note, if you make changes to the recent searches and your changes don't show up in Expression Web, make sure that Expression Web is closed when you are making your changes. If it is closed and your changes still don't take effect, it's likely that an instance of EXPRWD.EXE is running and you just can't see it. Right-click in an empty area of your taskbar, and select Task Manager from the menu. Click the Processes tab and look for EXPRWD.EXE. (You can click the Image Name column to search by that column.) If you find an active instance, select it and click End Process to get rid of it. You should then be able to successfully edit your recent searches.

This page intentionally left blank

CONFIGURING PAGE EDITOR OPTIONS

Accessing Page Editor Options

Expression Web is highly configurable and most of its configuration options are packed into a single dialog—the Page Editor Options dialog. To access the Page Editor Options dialog, select Tools, Page Editor Options.

The first thing you notice is the vast array of options spread out among 11 tabs. Instead of covering these options throughout the entire book, I cover most of them in this chapter so you learn how to use these settings effectively.

Exploring Page Editor Options

As mentioned previously, the Page Editor Options dialog consists of 11 tabs, many of which are packed full of options. We cover each of these tabs (with the exception of the AutoThumbnail tab and the Code Formatting tab, which are covered elsewhere in the book) in this section.

The General Tab

The General tab (shown in Figure 11.1) contains general options for Expression Web divided into five sections.

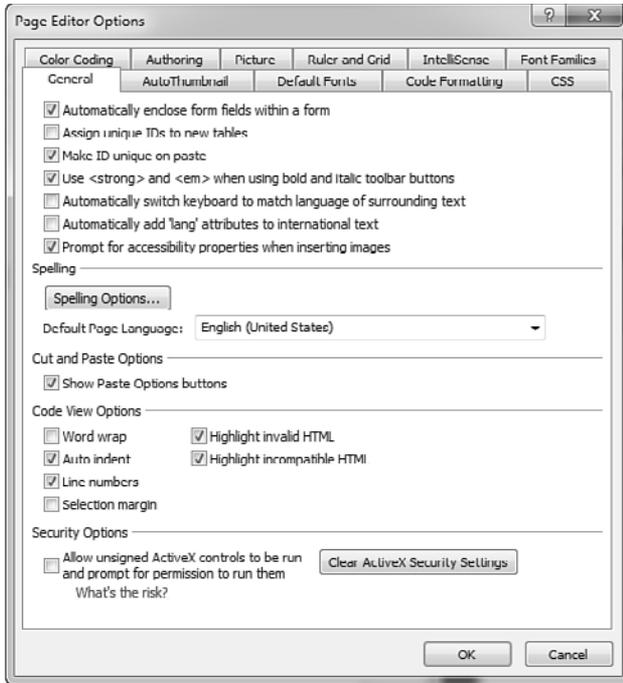


Figure 11.1

All general options for Expression Web are located on the General tab.

The following options are available in the first section on the General tab:

- **Automatically Enclose Form Fields Within a Form**—By default, the first form field you insert on a page is automatically wrapped in an HTML `<form>` tag. You can override this behavior by unchecking this box.
- **Assign Unique IDs to New Tables**—If you want Expression Web to add an ID attribute to all new tables, check this box. Existing tables are not affected.
- **Make ID Unique on Paste**—When this option is checked, Expression Web ensures that no duplicate control IDs are created when a control is pasted onto a page.
- **Use `` and `` When Using Bold and Italic Toolbar Buttons**—By default, Expression Web uses the `` and `` tags to apply bold and italic formatting, respectively. This method is preferred to the alternative of using `` and `<i>` tags, both of which are obsolete.
- **Automatically Switch Keyboard to Match Language of Surrounding Text**—Windows has the capability to use different virtual keyboard layouts for different languages. You can check this box to cause Expression Web to automatically switch your keyboard based on the language of the text surrounding the insertion point.



tip

Some controls (such as form controls) use a name attribute instead of an ID attribute. Expression Web always creates a unique name attribute regardless of whether the Make ID Unique on Paste check box is checked.

- **Automatically Add 'lang' Attributes to International Text**—If you enter text in a language other than the language specified for the page, Expression Web adds a 'lang' attribute for the text that you enter. For more information on the 'lang' attribute, see www.w3.org/TR/html401/struct/dirlang.html#adef-lang.
- **Prompt for Accessibility Properties When Inserting Images**—Expression Web prompts you for alternative text and long descriptions when inserting images. If you'd prefer not to be prompted, uncheck this box.

➔ For more information on accessibility settings, see Chapter 12, “Maintaining Compatibility and Accessibility.”

The second section of the General tab provides access to spelling options, most of which are configured by clicking the Spelling Options button, as shown previously in Figure 11.1. When the Spelling Options button is clicked, the Spelling Options dialog shown in Figure 11.2 is displayed so you can configure how Expression Web checks spelling.

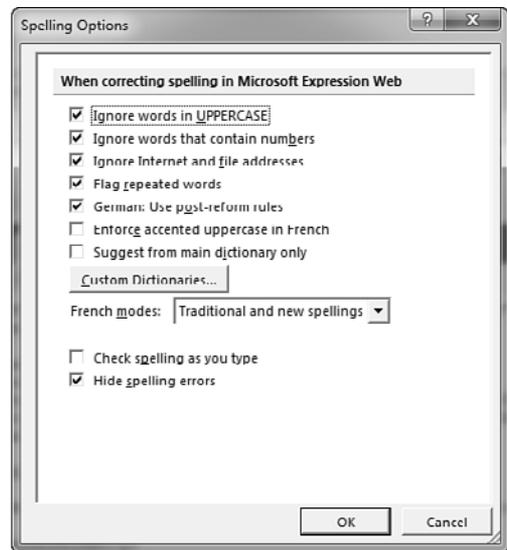


caution

Turning off the Prompt for Accessibility Properties When Inserting Images feature makes it more likely that you will forget to add alternative text or long descriptions to your images. If you omit these attributes from your images, your site will not pass accessibility standards.

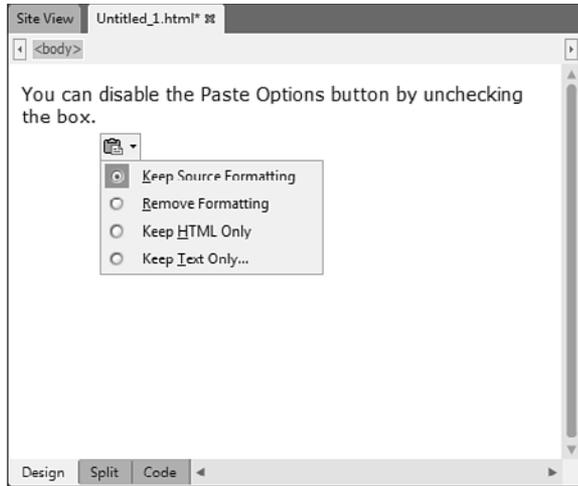
Figure 11.2

The Spelling Options dialog provides plenty of options to customize how Expression Web checks your spelling.



You can also select a language for spell-checking by clicking the Default Page Language drop-down on the General tab and selecting the language of your choice.

The third section of the General tab contains one check box: Show Paste Options Buttons. This check box is checked by default and causes Expression Web to display the Paste Options button when you paste content onto a page, as shown in Figure 11.3. You can disable the Paste Options button by unchecking the box.

**Figure 11.3**

The Paste Options button makes it easy to control content pasted into a page. You can disable the button by unchecking the Show Paste Options Buttons check box on the General tab of the Page Editor Options dialog.

The fourth section of the General tab contains options for Code View. The following options are available:

- **Word Wrap**—This check box controls whether word wrap is enabled in Code View.
- **Auto Indent**—This check box controls whether HTML elements are automatically indented underneath the parent element.
- **Line Numbers**—This check box controls whether line numbers are displayed.
- **Selection Margin**—When this check box is checked, a left margin is added in Code View to make selecting code easier.
- **Highlight Invalid HTML**—By default, Expression Web highlights any invalid HTML in a page. By unchecking this box, you can turn off this behavior.
- **Highlight Incompatible HTML**—By default, Expression Web highlights any HTML that is incompatible with the current schema. By unchecking this box, you can turn off this behavior.

The final section of the General tab contains a security setting for ActiveX controls. Expression Web runs Microsoft Silverlight, Windows Media content, and Adobe Flash content without any prompt. However, by default, if you open a page with any other type of ActiveX control, Expression Web prompts you to run the control unless it is signed. By unchecking the Allow Unsigned ActiveX Controls to Be Run and Prompt for Permission to Run Them check box, you can prevent Expression Web from ever running an unsigned ActiveX control.

When you choose to run an unsigned ActiveX control, Expression Web keeps track of the controls you have trusted. If you want to clear the list of trusted controls, click the Clear ActiveX Security Settings button.

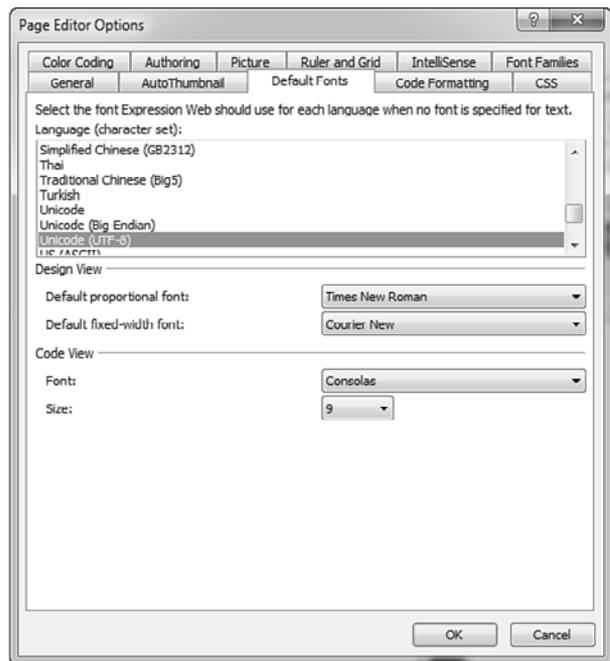
AutoThumbnail Tab

The options available in the AutoThumbnail tab are covered in detail in Chapter 9, “Using Graphics and Multimedia.”

Default Fonts Tab

The Default Fonts tab controls the font that Expression Web uses for a particular language when no font is configured in HTML or CSS code (see Figure 11.4).

Figure 11.4
You can control the default font in Expression Web by using the settings in the Default Fonts tab.



Code Formatting Tab

The options available in the Code Formatting tab are described in Chapter 4, “Using Page Views.”

CSS Tab

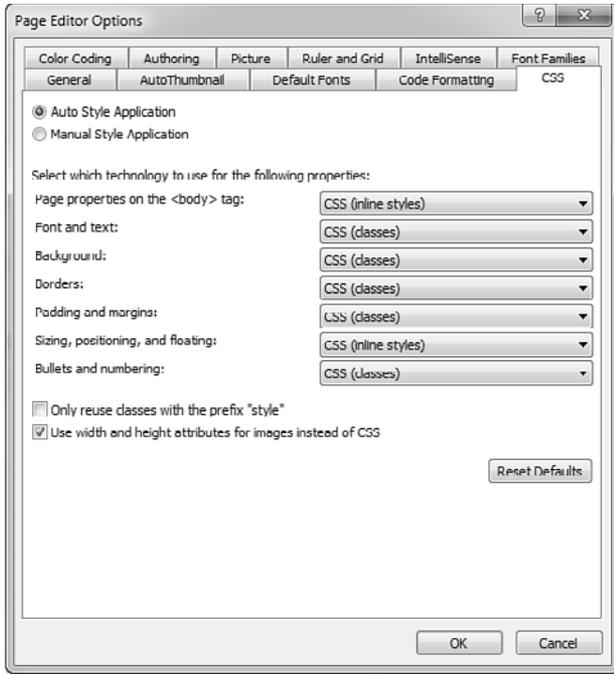
The CSS tab allows you to control how Expression Web uses CSS styles in your pages (see Figure 11.5).

➔ *For more information on using CSS with Expression Web, see Chapter 18, “Managing CSS Styles.”*



caution

Keep in mind that any font you configure in the Default Fonts tab might not be the font used when visitors to your site see your pages. Unless you explicitly specify a font and the site visitor has that font installed on his computer, the browser decides which font to use.

**Figure 11.5**

You can override Expression Web's default CSS behavior using the CSS tab.

By default, Expression Web automatically adds CSS code to your page when you apply formatting. This behavior is controlled by the following radio buttons in the CSS tab:

- **Auto Style Application**—Expression Web automatically applies CSS styles to formatted text using the CSS technology selected in the series of drop-downs. This is the default setting.
- **Manual Style Application**—Expression Web displays the Style Application toolbar shown in Figure 11.6 to allow you to manually choose how to apply styles.

**Figure 11.6**

The Style Application toolbar allows you to manually select how styles are applied.

In addition, there are a couple of check boxes to control how Expression Web deals with CSS styles:

- **Only Reuse Classes with the Prefix “Style”**—CSS classes that are added automatically by Expression Web always have a prefix of “style.” By checking this box, you can ensure that Expression Web will not modify any CSS styles that were not auto-generated. This option is available only when the Auto Style Application option is selected.

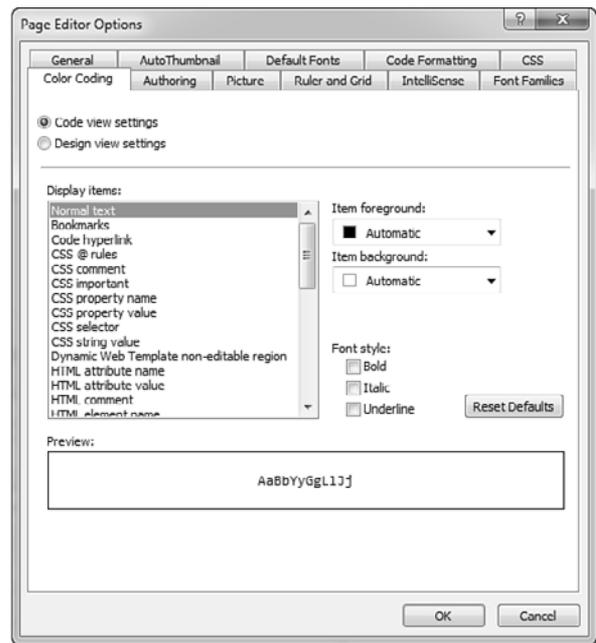
- **Use Width and Height Attributes for Images Instead of CSS**—By default, Expression Web uses the `width` and `height` attributes to specify the width and height of images. By unchecking this box, you can force Expression Web to use CSS styling when specifying the height and width of images.

Color Coding Tab

The Color Coding tab provides precise control over not only text elements, but also user interface elements in Expression Web (see Figure 11.7).

Figure 11.7

You can have highly detailed control over color coding using the Color Coding tab.



By selecting either the Code View Settings or the Design View Settings radio button, you can control which items appear in the Display Items list.

Authoring Tab

The Authoring tab, shown in Figure 11.8, lets you choose what kind of document Expression Web creates by default and also controls schemas for HTML documents.

The Default Document Type drop-down configures the type of document that Expression Web creates by default when you click the New Document button on the toolbar or press `Ctrl+N` on your keyboard. You can choose from HTML, ASPX, CSS, XML, or text file.

The Default HTML File Extension radio buttons allow you to select the file extension that Expression Web uses by default for new HTML files.

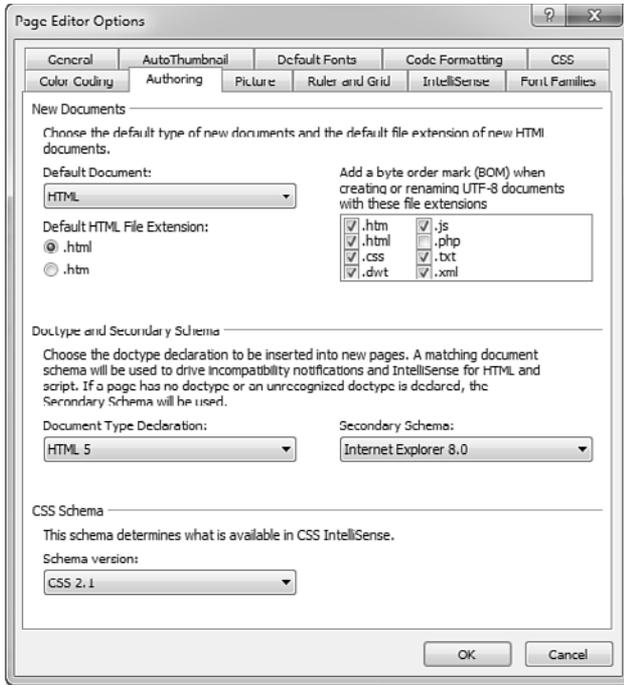


Figure 11.8
Document schemas are configured in the Authoring tab.

The Authoring tab also provides settings for configuring a byte order mark (BOM) for specific file extensions. The first version of Expression Web caused problems for PHP developers because it added a BOM to all PHP pages. Expression Web 4 will not add a BOM to PHP files by default, and it adds control over the BOM for specific file types.

The Document Type Declaration drop-down controls the DOCTYPE declaration included at the top of your pages. The DOCTYPE controls how Expression Web determines what code is valid. If code in the page isn't valid or compatible with the selected schema, Expression Web warns you with a warning symbol in the status bar as shown in Figure 11.9.

The Schema Version drop-down list contains configuration options for the CSS schema of the document. This setting affects only what appears in IntelliSense for CSS in the document. By selecting the CSS schema you want to target, you ensure that IntelliSense in Expression Web doesn't display invalid CSS values.



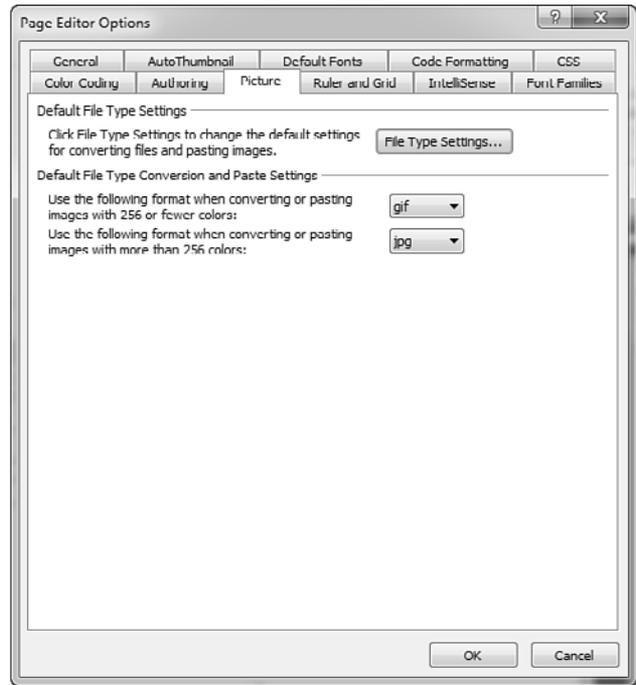
Figure 11.9
Expression Web uses the document schema to determine when invalid code exists in a page. If it finds invalid code, you are notified.

Picture Tab

The Picture tab configures how Expression Web works with images (see Figure 11.10).

Figure 11.10

You control how Expression Web works with image files by using the Picture tab.



The File Type Settings button displays the Picture File Type dialog, as shown in Figure 11.11. This dialog lets you easily configure settings for both GIF and JPEG images.

➔ *For more information on image file formats, see Chapter 9, “Using Graphics and Multimedia.”*

Ruler and Grid Tab

The Ruler and Grid tab, shown in Figure 11.12, controls the appearance of the grid and ruler that can be displayed in Design View. You can also choose the unit used for both the grid and the ruler.

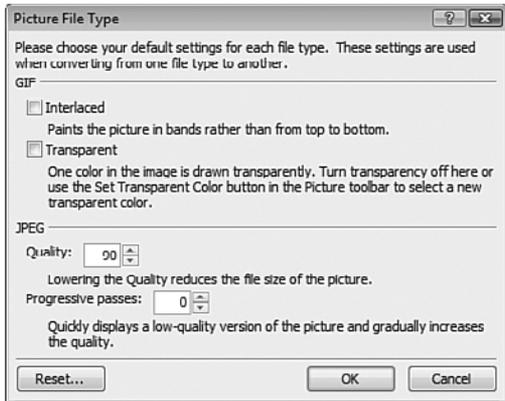


Figure 11.11
GIF and JPEG options are configured in the Picture File Type dialog.

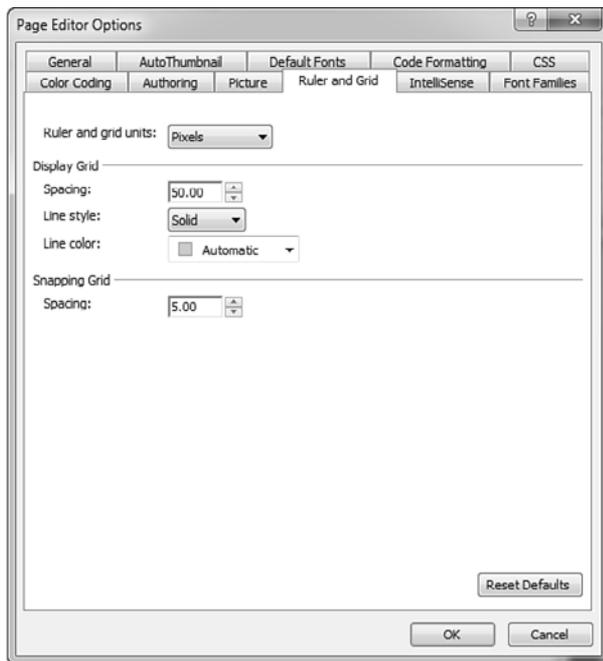


Figure 11.12
Control the appearance of the grid and ruler using the Ruler and Grid tab.

The Display Grid section controls the grid that is visible when selecting View, Ruler and Grid, Show Grid. The Snapping Grid section allows you to configure the spacing for the Snap to Grid feature that is accessed by selecting View, Ruler and Grid, Snap to Grid.

➔ For more information on the grid and ruler, see Chapter 4, “Using Page Views.”

IntelliSense Tab

The IntelliSense tab controls what IntelliSense displays in Code View (see Figure 11.13). You can turn off specific elements in IntelliSense by unchecking the desired check boxes. First introduced in Expression Web 3, PHP IntelliSense options can also be configured from this tab.

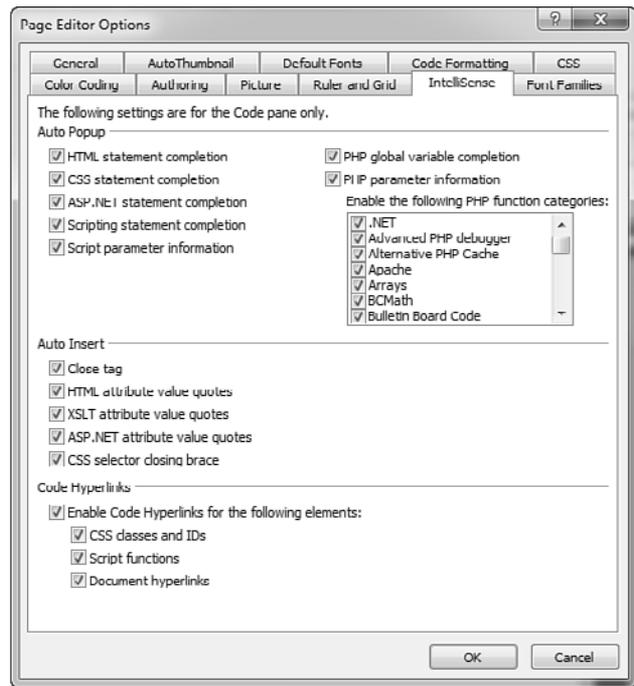


note

Available only in Code View, IntelliSense displays a context-sensitive drop-down list of choices as you enter code.

Figure 11.13

The IntelliSense feature available in Code View can be controlled using the options available in the IntelliSense tab.



Font Families Tab

The Font Families tab, shown in Figure 11.14, lets you easily configure one or more font families for selection in the Font drop-down on the Common toolbar, as shown in Figure 11.15.

➔ *For more information on using font families, see Chapter 3, “Creating Pages and Basic Page Editing.”*

You can build a new font family by selecting fonts from the Add Font list and clicking Add. After you’ve created the desired font family, you can arrange the fonts by clicking Move Up or Move Down. You also can remove the font family by selecting it and clicking Remove.

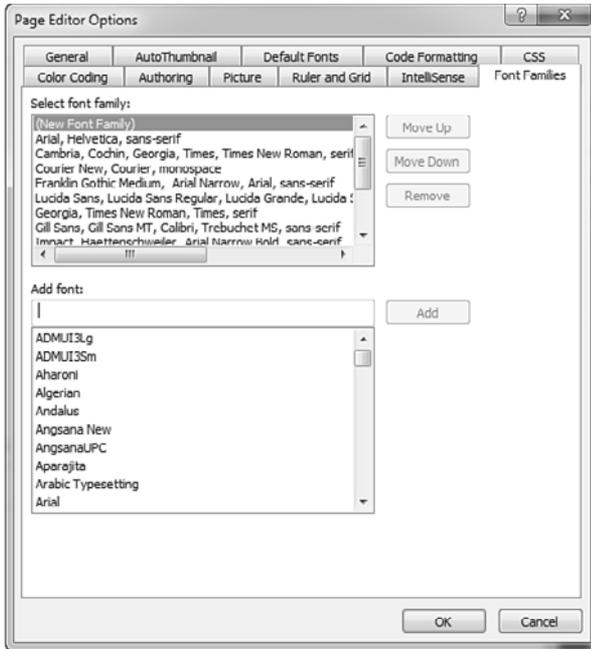


Figure 11.14

The Font Families tab is an easy way to configure and arrange font families in Expression Web.

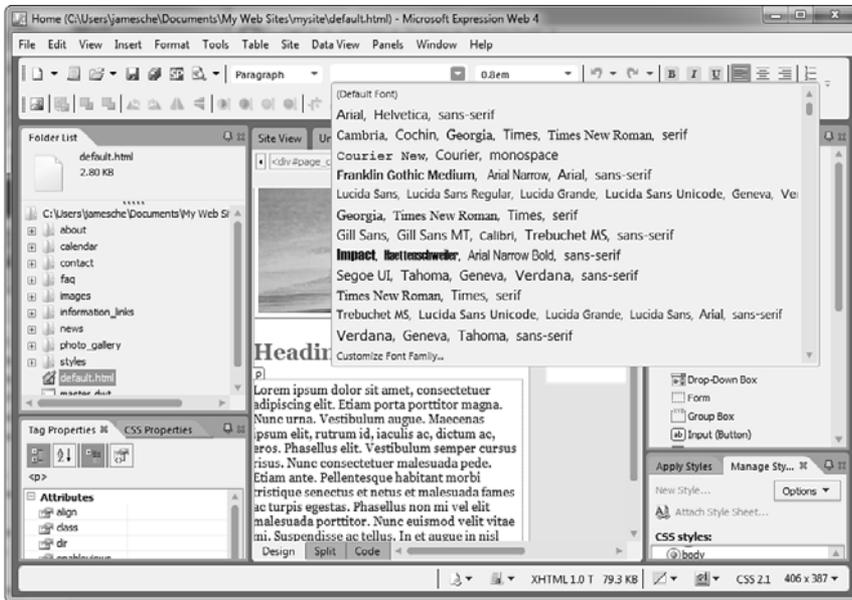


Figure 11.15

Font families are selected from the Font drop-down on the Common toolbar.

Experiment to Learn More

Numerous options affect the way code is generated by Expression Web. Most of these options won't affect code that has already been generated. Because of that, it's important for you to understand how each option affects code so you can make good decisions.

Some of the options covered in this chapter cause subtle changes in the way Expression Web creates code. For example, changes to the Code Formatting or CSS properties may not cause obvious changes in the way Expression Web generates code. The best way to fully understand these options is to create a new page and use it to test how different options affect the code Expression Web creates. By limiting your testing to small amounts of code, changes in Expression Web's code will be more apparent.

If you've experimented with options in the Page Editor Options dialog and want to revert to the original settings, you can click the Reset or Reset Defaults button that exists on many tabs. Note that each tab's button resets the options only on that particular tab.

This page intentionally left blank

MAINTAINING COMPATIBILITY AND ACCESSIBILITY

An Introduction to Accessibility

It's amazing to think that despite the fact that the World Wide Web is in its infancy, access to it has become an integral part of our lives. Access to the Web is quickly becoming not just a convenience, but a necessity.

This sense of necessity has largely driven the need for computer applications and sites to be accessible to people who might have hearing loss, low vision, or other disabilities. In fact, in 1998, Congress enacted the Workforce Investment Act, which consisted (in part) of an amendment to the Rehabilitation Act of 1973. That amendment was titled Section 508, and it has become synonymous with accessibility in the world of technology.

Government agencies and most educational institutions are required to abide by Section 508 requirements. Section 508 specifies requirements not only for sites, but also for multimedia, video, telecommunication products, and more. It is far-reaching legislation that affects most web designers.

Accessibility doesn't just refer to the ability of people with vision problems to experience a site via a screen reader. It also applies to proper color choice so that those with various types of color blindness can read your site and so on. It even prohibits design techniques that can cause screen flicker beyond a certain frequency. All in all, 16 rules comprise Section 508 standards.

Designing for Accessibility

There are many points to consider when designing your site to be accessible. Expression Web provides some user interface elements designed to keep you in compliance with accessibility requirements. Many other techniques require attention to detail and a touch of common sense when developing your site.

Accessible Hyperlinks

When creating a hyperlink in Expression Web, you'll have the option of also specifying a ScreenTip by clicking the ScreenTip button in the Insert Hyperlink dialog, as shown in Figure 12.1.

When you click the ScreenTip button, the Insert Hyperlink ScreenTip dialog is displayed, as shown in Figure 12.2. Enter the text for the ScreenTip and click OK. When a user hovers over the hyperlink with his mouse, the text configured as the ScreenTip pops up next to the mouse. Additionally, the text you enter is read by a screen reader for those with visual disabilities.

note

Section 508 is not the only standard for accessibility, but it's the only government-enforced standard. The W3C has devised the Web Content Accessibility Guidelines (WCAG) in an effort to enhance the accessibility of sites. Expression Web recognizes both Section 508 and WCAG. However, Expression Web only checks your site against WCAG 1.0 and not the updated WCAG 2.0.

note

For full details on all Section 508 rules regarding sites, see www.section508.gov.

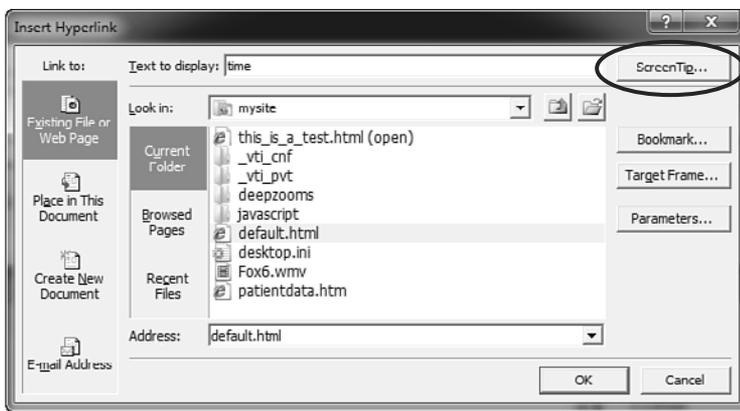


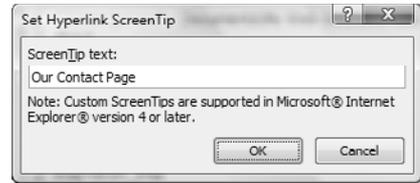
Figure 12.1
Configuring ScreenTips for hyperlinks is important for accessibility. Expression Web makes it easy.

note

The ScreenTip feature in Expression Web is implemented using the `title` attribute of the hyperlink.

Figure 12.2

Enter the text for your ScreenTip. It will then be available to a screen reader for those with visual disabilities.



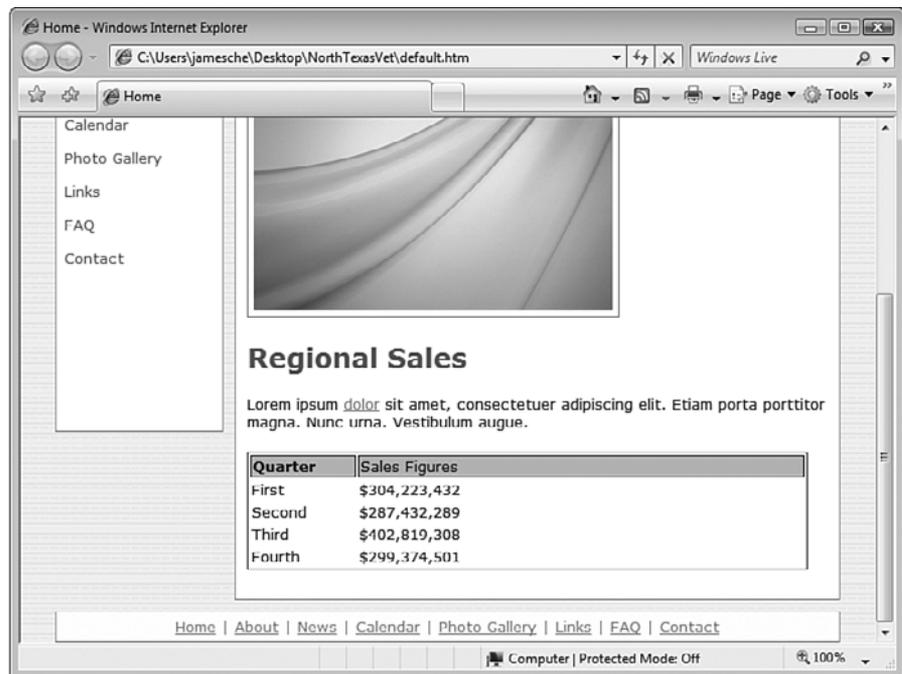
➔ For more information on inserting hyperlinks, see “Creating Hyperlinks,” Chapter 3, p. 55.

Accessible Tables

When creating a table, it's often tempting to describe the table in the site's text and then present the data without headers. However, Section 508 requires that all tables containing data must also contain header information. The table shown in Figure 12.3 is compliant with Section 508 guidelines.

Figure 12.3

All data tables must have header information. This table meets that requirement.



➔ For more information on creating tables, see Chapter 5, “Using Tables.”

Accessible Frames

Frames pages are a challenge for screen readers because a screen reader sees a frames page as two or more separate pages. You should follow a few general guidelines to make frames pages more accessible.

Make sure each `<frames>` tag in the frameset contains a `title` attribute that clearly defines the purpose of the page. For example, the following `<frames>` tags aid in accessibility:

```
<frame title="Site Navigation" name="navframe" src="nav.htm" />
<frame title="Main Site Content" name="main" src="main.htm" />
```

You should also be sure you include a proper DOCTYPE declaration on your frameset page. The following DOCTYPE declaration should be used on a frameset that conforms to the XHTML 1.0 standard:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

Finally, make sure you include a `<noframes>` section in your frameset.

➔ *For more information on using frames, see Chapter 6, “Using Frames.”*

Other Accessibility Considerations

There's no possible way to cover all the accessibility topics in this chapter. However, in addition to the common issues we've already discussed, there are some general considerations to keep in mind as you develop a site.

Be sure to always provide some alternative representation for images, navigation elements, Adobe Flash animations, and any element implemented with client script. Expression Web provides user interface elements for ensuring that you meet the necessary requirements in this area. For example, when you insert an image onto a page, Expression Web displays the Accessibility Properties dialog shown in Figure 12.4.

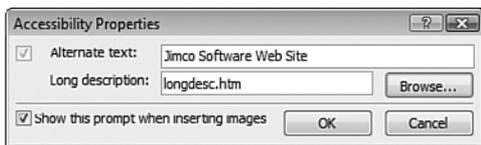


Figure 12.4
The Accessibility Properties dialog serves as a reminder to enter attributes required for accessibility.

Another item web designers often overlook is that a site should be viewable without the application of any style sheets. For example, consider a site with white text inside a table styled with a black background using a style sheet. Because Internet Explorer uses white as a background color by default, the text for such a site would not be visible to anyone without CSS support. Not only is that a counterproductive design, but it also falls outside the Section 508 requirements.

The bottom line is that you should always think about accessibility when designing your sites. Many accessibility rules are based on common sense. However, no matter how much effort you put into making your sites accessible, you are bound to miss something. Thankfully, Expression Web provides an Accessibility Checker feature so you can locate accessibility problems and correct them before you deploy a site.

Using the Accessibility Checker

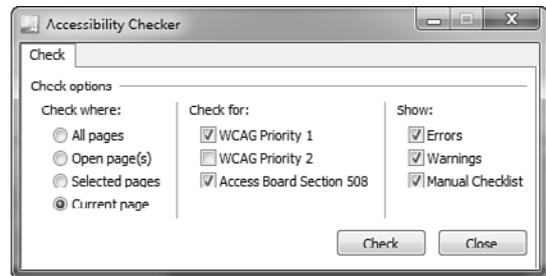
The Expression Web Accessibility Checker allows you to run reports on how well your site holds up to accessibility standards. The Accessibility Checker can check one or more pages (or your entire site) against WCAG and Section 508 requirements and provide a comprehensive report of the results.

Checking Accessibility

To access the Accessibility Checker, select Tools, Accessibility Reports to display the dialog shown in Figure 12.5.

Figure 12.5

The Accessibility Checker is a flexible tool for finding accessibility problems in your site.



The Check Where section of the Accessibility Checker dialog lets you check all pages, all open pages, selected pages, or the current page for accessibility problems. The Check For section allows you to choose which standard(s) you want to check your pages against. You can choose WCAG Priority 1, WCAG Priority 2, and Section 508 requirements.

The Show section provides a series of check boxes that control how much information is displayed in the accessibility report. The following options are available:

- **Errors**—Represents problems that cause the page to fail accessibility requirements. Any errors must be corrected if the site is to fall within accessibility standards.
- **Warnings**—Represents areas of a page that might be problematic based on the content. You should review these areas for possible accessibility problems.

note

Priority 1 results are problems that must be corrected for a page to remain compliant with accessibility standards. A priority 2 result should be corrected for maximum accessibility, but not doing so does not cause your page to fail validation.

- Manual Checklist**—Lists the general requirements for the selected standards. These are somewhat similar to reminders that are designed to ensure you are aware of the requirements imposed by the selected standards. Because the items on this list cannot be checked automatically, you will want to check them manually.

After you've made your choices in the Accessibility Checker, click Check to check the page(s). When the check is complete, Expression Web displays the results in the Accessibility panel, as shown in Figure 12.6.

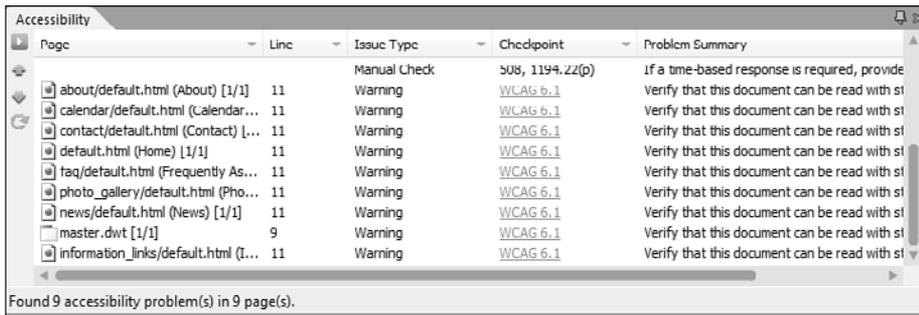


Figure 12.6
The Accessibility Checker's results are displayed in the Accessibility panel.

Working with the Accessibility Panel

Additional options in the Accessibility panel are available by right-clicking a particular item. For example, for additional details on an item, right-click the item and select Problem Details. Expression Web displays the Problem Details dialog with details on that item (see Figure 12.7). This is useful not only to get a better understanding of the problem, but also to provide hints on how to correct the issue.



Figure 12.7

The Problem Details dialog can provide helpful hints on how to resolve issues that appear in the Accessibility report.

If a particular item appears because of a WCAG checkpoint, additional information on that specific item can be obtained by either clicking the WCAG link in the Checkpoint column (shown previously in Figure 12.6) or by right-clicking the item and selecting Learn More from the menu. Doing so opens your browser and takes you to the W3C site, where you can read about the specific requirement, as shown in Figure 12.8.

To correct a problem or warning displayed in the Accessibility panel, double-click the entry. Expression Web opens the applicable page and selects the code that caused the error or warning, as shown in Figure 12.9, so that it can be easily corrected.

Figure 12.8

Expression Web provides direct links to W3C documentation on particular items in the Accessibility panel.

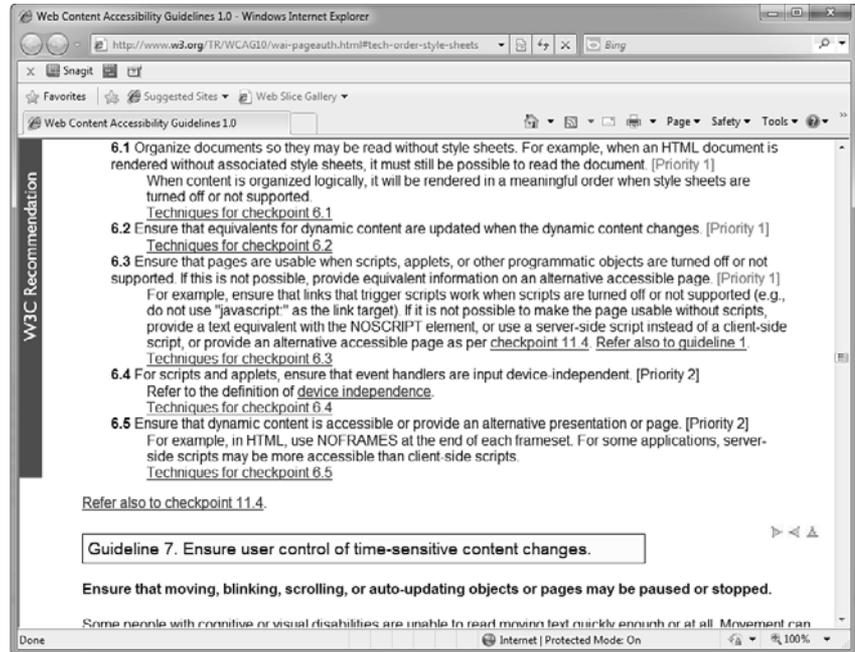
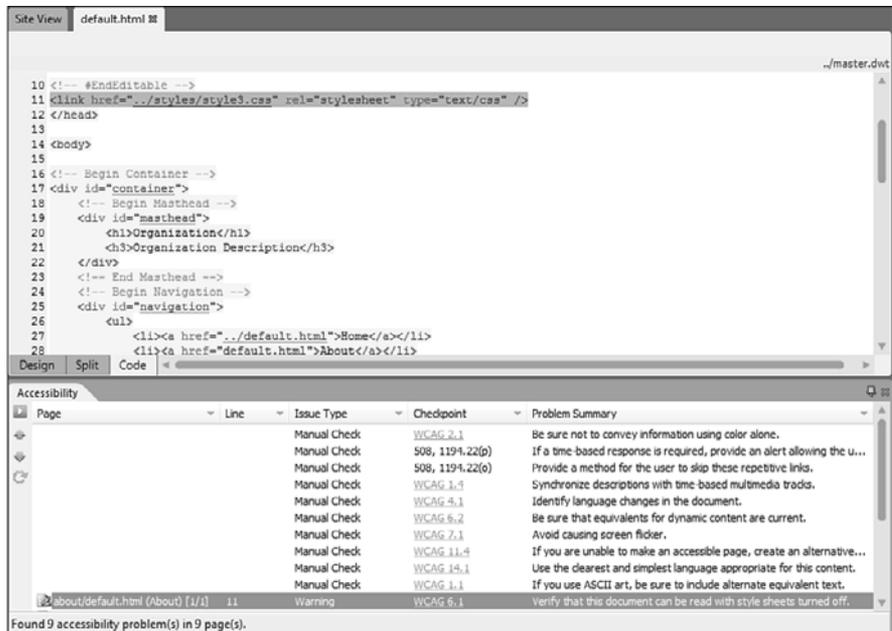


Figure 12.9

Locating problems in code is easy. Just double-click the item in the Accessibility panel to open the page and highlight the offending code.



Generating Accessibility Reports

In some cases, saving the results of the Accessibility Checker can be convenient. Perhaps you are not the person responsible for correcting all the errors and warnings, or maybe you want to correct the problems over time. Expression Web lets you copy selected Accessibility Report results so that you can paste them into a new page for printing or saving.

To generate a page with details of selected Accessibility Report results, select the entries you're interested in. Right-click and select Copy Selected Results from the menu as shown in Figure 12.10. You can then paste the entries into a new page. The resulting page is shown in Figure 12.11.

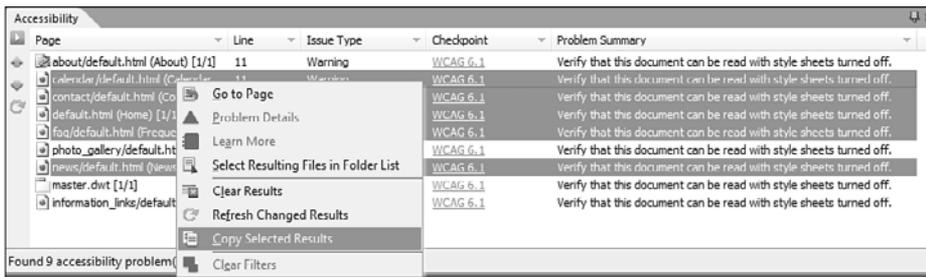


Figure 12.10
A printable report can be generated by copying results from the Accessibility panel.

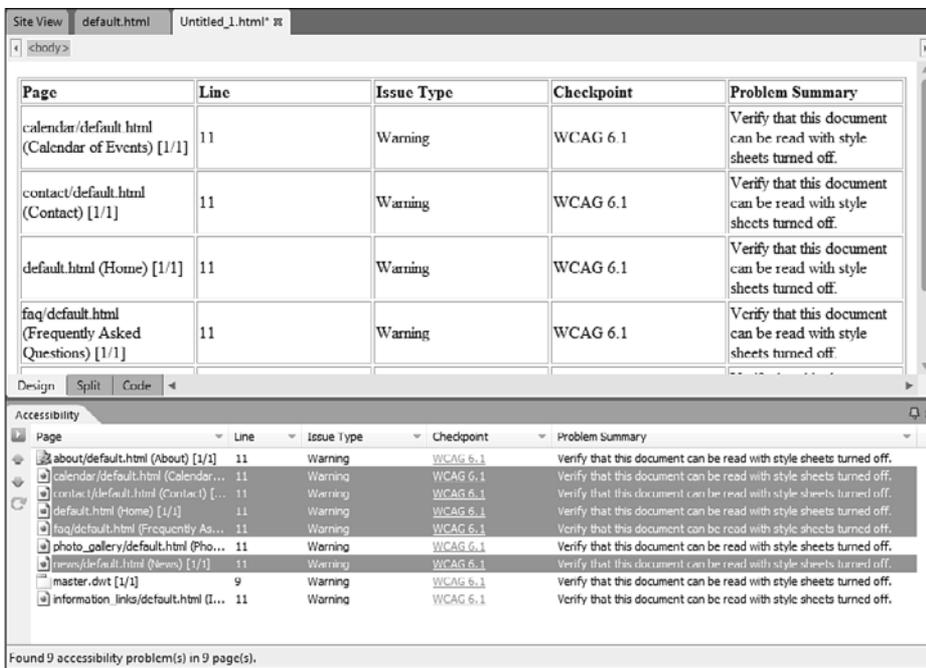


Figure 12.11
Accessibility Report results can be pasted into a new page for easy printing or saving.

Not all of us are required to abide by Section 508, but all of us should. Considering the effort web designers take to ensure that their sites render correctly in all browsers, it only makes sense that equal effort should go into ensuring that your sites are accessible to those with disabilities.

Designing for Compatibility

Here's a trivia question for you: What was the name of the first web browser? For many of you, Mosaic probably came to mind. Although Mosaic is commonly thought of as the first browser, it was not. The first web browser was invented by Tim Berners-Lee in 1989, and it was called WorldWideWeb.

WorldWideWeb was nothing like what comes to mind when we think of a web browser today. It didn't have the capability to render graphics. Even so, it was an extraordinary accomplishment, and—as history can testify—it transformed the way we access information.

In 1993, NCSA Mosaic 1.0 was released, but it also lacked the capability to display images. That capability wasn't made available until 1994, when Mosaic 2.0 was released.

In 1994, Netscape Communications released Netscape 1.0. A year later, Microsoft released Internet Explorer 1.0, and the race was on. By 1997, Microsoft had released Internet Explorer 4.0, which contained a solid implementation of the Document Object Model (DOM), allowing web developers to begin using JavaScript to manipulate pages. In the meantime, Netscape had released version 3.0 and the Netscape 3.0 Gold Edition, which contained a what-you-see-is-what-you-get (WYSIWYG) web design tool called Netscape Composer. Add Microsoft's FrontPage 97 (another WYSIWYG web design tool available at the time) into the mix and the stage was set for a compatibility nightmare.

What Is Browser Compatibility?

The concept of browser compatibility can be traced directly to the browser wars of the 1990s. As Netscape and Microsoft battled it out for browser superiority, web developers became casualties of that war. Code that looked great in one browser looked entirely different on another browser. The design divergence wasn't limited to Microsoft versus Netscape design decisions. Pages that looked great in one version of a particular browser might fall apart on a different version of the same browser.

As the war heated up, it became clear that web designers would not get relief from browser developers. Instead, it was up to web designers to “fix” the problem. Web development hacks began to pop up in forums frequented by designers. If you were a web developer during this painful period, you no doubt remember how difficult it was to keep up with all the latest techniques.

A few years ago, I could have easily filled an entire book on compatibility issues. Fortunately, that is no longer the case. Today's web design tools and web browsers are more standards-compliant. The hacks that web designers used in the past are often no longer necessary. To avoid compatibility



note

Recent browsers and design tools are almost entirely standards-compliant, but plenty of people are still using older browser versions. Keep that in mind when designing your site. See the “Should You Design for Older Browsers?” sidebar for my take on how to deal with this situation.

issues, today's web designer can focus on the standards and be fairly certain that her site will appear as expected for most people.

Should You Design for Older Browsers?

There has long been a debate as to whether web designers should ensure that pages display well on older browser versions. In some cases, your client might dictate how you approach this debate. After all, if someone is paying you to design a site, he likely will make the choice about designing for older browsers. However, if you're designing your own site or being asked for input from a client, you have a choice to make: Should you design to the standards or design to older browser versions?

In my opinion, designing to standards is a better choice. As you can clearly see by reading the opening section to this chapter, the Web evolves quickly. If you spend effort designing for specific browser versions, your work will quickly become outdated as people move to newer versions. It's simply a race you cannot win.

By focusing instead on web standards, you ensure that your site works well today and into the future. Browser developers spend considerable development cycles ensuring browsers are backward compatible. The standards of today will work well in browsers for many years to come.

Compatibility Features in Expression Web

Expression Web has many features designed to help you create standards-compliant sites. The most important feature offered by Expression Web is its capability to create standards-compliant code automatically. Because of that, you are free to work on your site without worrying about what's going on under the hood. However, what if you're working on a site that was originally created in another application? Fortunately, Expression Web offers features to help you identify and correct inherited code problems as well.

Identifying Code Problems

Expression Web has a few ways of notifying you when a code problem exists. Among these are the status bar, marking invalid code in Code View, and compatibility reports.

The status bar displays an icon when it finds incompatible code or a code error (see Figure 12.12). For example, if an HTML table is missing the closing `</table>` tag, Expression Web informs you that a code error has been detected. Other errors can be subtler, such as improperly closed tags in an XHTML-compliant document.



tip

The icon indicating incompatible code appears only when in Code View or Split View. The Code Error icon appears in all views when a code error is present.

Figure 12.12

In Code View, icons appear on the status bar when a code problem is found. (Both code errors and code incompatibilities.)



➔ For more information on identifying and correcting code errors in Code View, see Chapter 4, “Using Page Views.”

To determine whether code problems exist, Expression Web uses several methods to determine the formatting and layout rules applicable to the current page.

doctype

The doctype declaration is a statement contained within the first line of code in a page. It defines the rules that are applicable to the document. The following doctype specifies that a document should follow the XHTML 1.0 Transitional rules:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Expression Web displays the current doctype in the status bar. In Figure 12.12 (shown previously), the Status Bar indicates that the doctype is set to XHTML 1.0 Transitional.

➔ For more information on using Code Snippets, see Chapter 3, “Creating Pages and Basic Page Editing.”



note

XHTML is the latest approved standard in web design. The current XHTML standard is almost identical to the HTML 4.01 standard, with the addition of XML formatting and elements. (The target date for approval of HTML5 is 2014.)

A full discussion of XHTML is beyond the scope of this book. For more information on XHTML, read *Special Edition Using HTML and XHTML*, available from Que Publishing.

The Schema

If there is no doctype declaration in your document, Expression Web uses the schema of the document to determine whether problems exist. The schema is defined in Expression Web and can target a standard (for example, HTML 4.01 or XHTML 1.0) or a browser (for example, Internet Explorer 5.0 or Internet Explorer 8.0).



tip

Most doctype declarations contain a URL to the definition file (.dtd file) that defines the rules for the document type. Expression Web does not read the definition file to determine the rules of the document, but your web browser will often format the document differently if the URL is missing.

Needless to say, if you decide you want to design your page to be compatible with an older schema such as the Internet Explorer 5.0 schema, you're going to have a tough time. Much of the code Expression Web creates when you format page elements is CSS code, and older schemas cause Expression Web to complain heavily about such code.

Another drawback to using older schemas is that Expression Web will continue to create code that uses CSS elements and other features that likely aren't compatible with older schemas. If you insist on using older schemas, you should prepare yourself to do a fair amount of code editing and hand coding to replace the code that Expression Web generates.

To alter the doctype and schema settings, select Tools, Page Editor Options, and click the Authoring tab. The doctype and schema can be set using the doctype and Secondary Schema drop-down, as shown in Figure 12.13. Note that you can also select the CSS schema in this dialog, but doing so affects only what is displayed in IntelliSense.



caution

Expression Web will not alter the code it creates based on doctype or schema settings. Specifying a doctype or setting a schema only affects the rules Expression Web uses to flag certain code as invalid or incompatible.

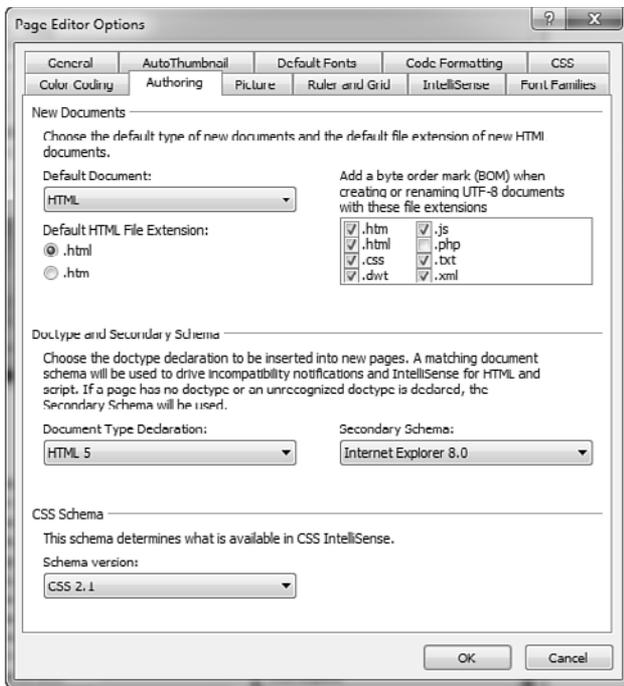


Figure 12.13

The doctype and schema are configured in the Page Editor Options dialog.

The Quirks of a Missing doctype

What happens if you omit the doctype declaration? Expression Web reverts to what is known as *quirks* rendering mode. Quirks rendering mode is a special mode built into most browsers. It is designed to alter the way certain elements are rendered in a page to attempt to work around known rendering bugs. When your browser is operating in quirks rendering mode, you might see unexpected results when your page is displayed.

Rendering problems experienced by web designers are often caused by missing doctype declarations. If you open a page and see Quirks displayed in the status bar in Expression Web, you need to add a doctype declaration so the browser knows what standard the page is using.

The easiest way to add a doctype declaration is to use the Code Snippets functionality of Expression Web. Simply press Ctrl+Enter and select the desired doctype from the list of Code Snippets.



doctype Not Added to Page

If you've selected a specific doctype in the Page Editor Options dialog but there isn't a doctype on the page when you check Code View, don't worry; it's not a bug.

The doctype that you select in the Doctype drop-down in the Page Editor Options dialog controls what gets added to new pages you create. It does not affect existing pages. If you want to add a doctype to an existing page, you must do it manually.

Marking Invalid Code

When Expression Web encounters invalid or incompatible code, it underlines the code in red. If you hover your mouse over the underlined code, a ScreenTip informs you of the problem, as shown in Figure 12.14.

➔ *For more information on working in Code View, see Chapter 4, "Using Page Views."*

When Expression Web encounters a code error (such as mismatching tags), it highlights the offending code in yellow. Just as with underlined code, you can hover the mouse over the highlighted code for details of the problem.



tip

You can quickly display the Page Editor Options dialog by clicking the doctype, schema, or CSS schema displayed in the status bar.

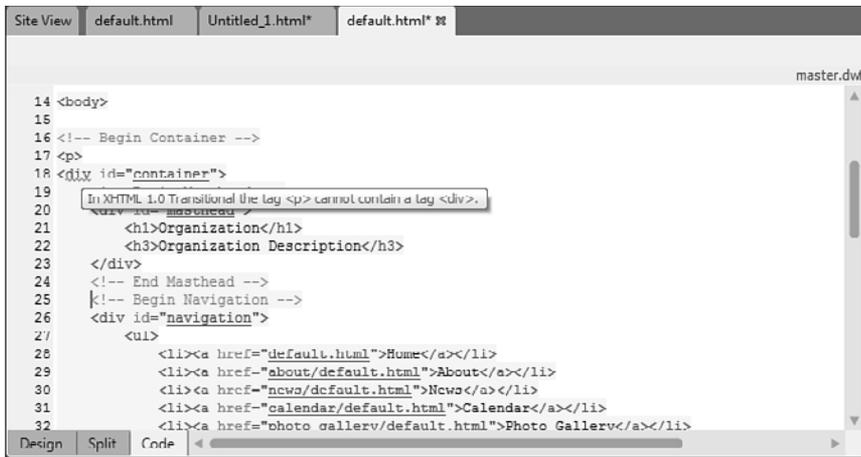


Figure 12.14
Expression Web
makes finding code
errors fast and easy
by highlighting them
in Code View.

Using Reports to Find Problems

Expression Web has three reporting options specifically designed to aid in locating problems with code:

- **Accessibility Reports**—Also called the Accessibility Checker, this report locates any code problems affecting the accessibility of your site.
- **Compatibility Reports**—This report analyzes your page code and shows problems with code that doesn't comply with the standards you've specified.
- **CSS Reports**—This reports on CSS usage and shows any problems with CSS code in your site.

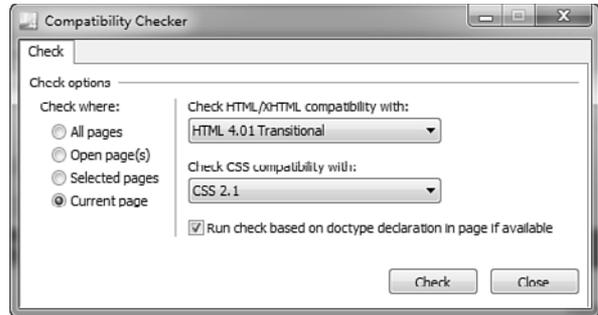
CSS reports are covered in-depth in Chapter 18, “Managing CSS Styles.” Therefore, we won't go into any detail on the reports here.

To access compatibility reports, select Tools, Compatibility Reports to display the Compatibility Checker dialog, as shown in Figure 12.15. You'll have the option of checking all pages, all open pages, selected pages, or the current page using the Check Where options. Pages can be checked against selected HTML/XHTML standards using the Check HTML/XHTML Compatibility With drop-down, and CSS code can be checked against a CSS standard specified in the Check CSS Compatibility With drop-down.

If you have a doctype declaration on your pages, you can check the Run Check Based on Doctype Declaration in Page if Available check box and the Compatibility Checker checks your pages based on the doctype. When you use this option, the setting in the Check HTML/XHTML Compatibility With drop-down is ignored if a doctype declaration is on your page.

Figure 12.15

The Compatibility Checker makes it easy to find problems in your code.

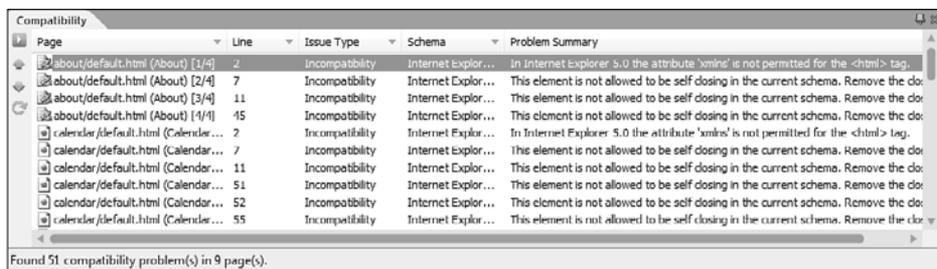


It's important to realize that checking the Run Check Based on Doctype Declaration in Page if Available check box does not automatically negate the setting in the Check HTML/XHTML Compatibility With drop-down. In fact, you might find yourself in a situation where you have inherited a site that contains some pages with a doctype declaration and some pages without one. In such cases, it's often convenient to check against the doctype in the page when present, but check other pages against a particular standard in preparation for adding a doctype later. It's in these types of situations that the flexibility of the Compatibility Checker is truly appreciated.

The results of the Compatibility Checker are displayed in the Compatibility pane, as shown in Figure 12.16. Each column in the Compatibility pane can be sorted by clicking the column header. Double-clicking an item in the list takes you to the code that is causing the problem so you can correct it. As you make corrections, you can use the buttons along the left edge of the Compatibility pane to refresh the results, rerun the compatibility reports, and navigate between compatibility errors.

Figure 12.16

The Compatibility pane displays the results of a compatibility check in an easy-to-filter view.





Code View Errors and Compatibility Checker Don't Match

If you see one problem in the Compatibility Checker, but in Code View, Expression Web is highlighting many errors that don't show up in the Compatibility pane, it's likely that you are using a standard in the Compatibility Checker that does not match the standard chosen in the Page Editor Options dialog. You can avoid this kind of problem by always including a `doctype` declaration on your pages.

By clicking the small arrow at the right edge of a column in the Compatibility pane, the results can be filtered using either the drop-down menu that appears or by selecting Custom from the menu and entering filtering criteria in the Custom AutoFilter dialog. To remove the filter you've applied, right-click the column header and select Clear Filters, as shown in Figure 12.17.

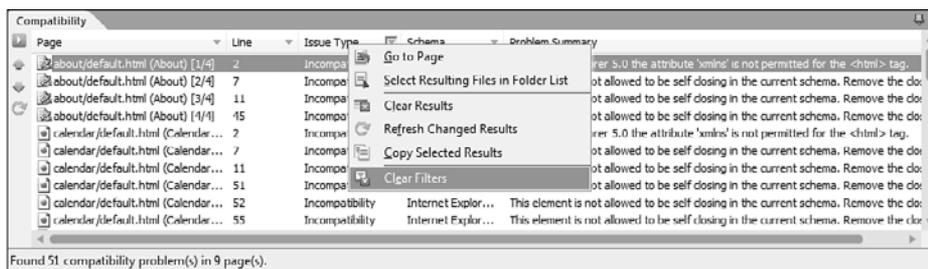


Figure 12.17 Filters can be removed using the context menu that appears when you right-click a column header.

Seeing Color

Many of us take for granted that everyone sees color the same way we do. In fact, many people do not. There are many forms of color blindness, and each of them can affect the way your site appears. However, because most of us see color accurately, ensuring our site appears correctly to those with a form of color blindness is a tough job. Vischeck can make it a lot easier.

Vischeck (www.vischeck.com/vischeck/) offers a site where you can enter a URL and generate a view of what the URL looks like to people with various forms of color blindness. You can also upload an image file and Vischeck provides a view of that image as it appears to someone with color blindness.

If you'd prefer, you can download a plug-in from Vischeck (free of charge) that allows you to run Vischeck's algorithm on images from within Adobe Photoshop or another image-editing application that supports Photoshop plug-ins.

Choosing colors for your site is a critical task. You can avoid selecting colors in a vacuum by using Vischeck to ensure that your color choices make your site accessible.

USING SUPERPREVIEW

An Overview of SuperPreview

In the early days of web design, it wasn't uncommon for a page to render very differently from one browser to the next. Web standards were still largely in flux and proprietary code was widespread. Because of these conditions, web designers would typically either spend a considerable amount of time implementing hacks to fix rendering problems or would design for one particular browser flavor and recommend that site visitors use that specific browser.

Web standards have solidified greatly over the years, and modern browsers are typically compliant with those standards. However, these conditions don't mean the end of compatibility concerns. Some users are still using older browsers, and some of those browsers can introduce some serious layout problems.

SuperPreview

Expression Web includes SuperPreview, an extremely powerful tool for identifying and correcting layout problems with your site. SuperPreview has many benefits over traditional tools for previewing pages, but perhaps the most revolutionary is the capability to preview a page using several versions of Internet Explorer without having to install multiple versions and the ability to preview your page in Safari on a Mac using the remote browser service.

SuperPreview enables you to view a preview of a page in a baseline browser and a comparison browser. Figure 13.1 shows a SuperPreview preview of the Jimco Books site using Internet Explorer 6 as the baseline browser and Firefox 3.6.6 as the comparison browser. SuperPreview's built-in rendering engine for Internet Explorer 6 makes it possible to see exactly what your page looks like in Internet Explorer 6 without having it installed on your system.



Figure 13.1 SuperPreview is a powerful tool for checking your page's appearance and layout in multiple browsers.

How SuperPreview Generates a Preview

When you launch SuperPreview, it first determines which browsers should be available for preview. If you have Internet Explorer 6 installed, SuperPreview only provides a preview of Internet Explorer 6. If you have Internet Explorer 7, SuperPreview makes Internet Explorer 6 and 7 available for preview. If you have Internet Explorer 8, SuperPreview makes Internet Explorer 6, 7, 8, and Internet Explorer 8 in IE7 compatibility mode available for preview.

In addition to these IE versions, SuperPreview also makes Firefox available as a choice if you have Firefox installed. SuperPreview also supports Safari on the Mac using a remote browser service.

➔ *For more information on previewing using remote browsers, see "Using Remote Browsers," p. 230.*

To generate a preview using SuperPreview, first select the baseline browser by selecting one of the browser options from the left pane; then select a comparison browser from the right pane. Enter a URL to preview in the Location box as shown in Figure 13.2 and press Enter.

note

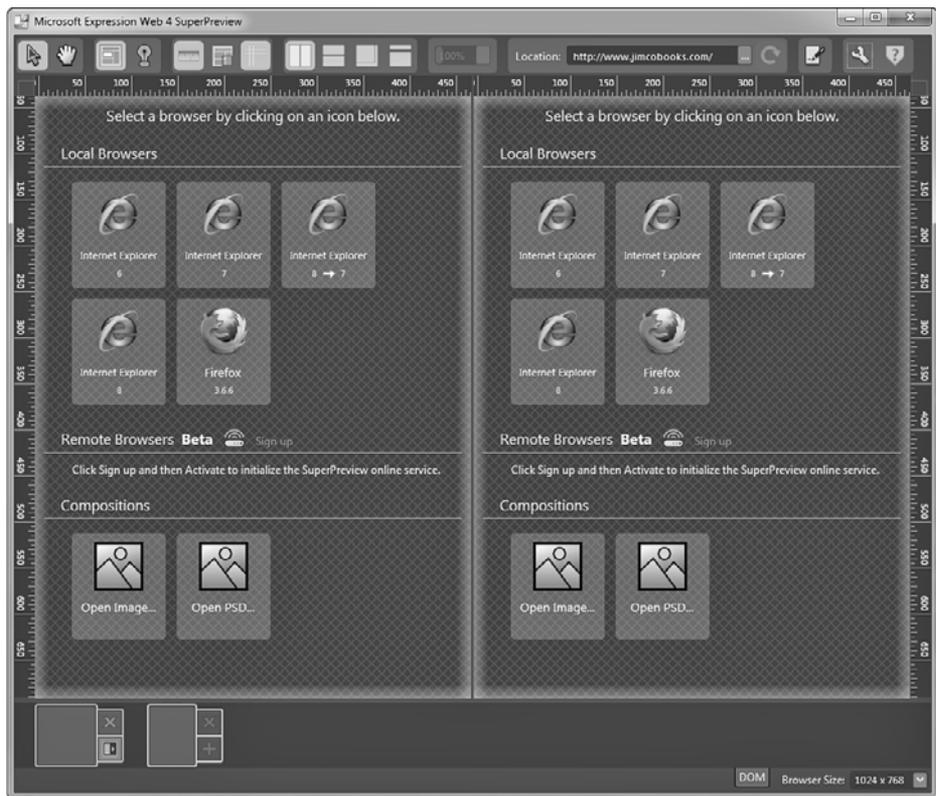
Microsoft has released two different versions of SuperPreview. SuperPreview for Internet Explorer is a free, standalone application and supports only Internet Explorer. SuperPreview with Expression Web adds support for Firefox and remote browsers and is integrated with Expression Web. However, it can also be launched separately using the Microsoft Expression Web 4 SuperPreview icon that is installed in your Start menu when Expression Web is installed.

When SuperPreview builds its preview, it runs the URL you specify through the actual rendering engine for the browsers you've chosen. In addition to loading the page, SuperPreview also executes all script in the page through the OnLoad event so that any client-script designed to change page rendering is recognized by the preview that is generated. The result is a 100% accurate preview of your page. In other words, if you select Internet Explorer 6 and generate a preview, you can be assured that what you see is exactly what visitors using Internet Explorer 6 will see when your page is viewed.

**tip**

It's important to realize that SuperPreview is not an interactive web browser. SuperPreview essentially generates a snapshot of a page. Therefore, interactive UI elements such as menus and links don't work in SuperPreview.

Figure 13.2
The left pane in SuperPreview shows options for the base-line browser, whereas the right pane shows the comparison browser.



The SuperPreview Interface

Before we go too much further into using SuperPreview, it might help to look at the SuperPreview user interface. After we've reviewed all of the interface tools available, we'll look at how you can use those tools to identify rendering problems in your pages.

Pointer Modes

The leftmost two toolbar buttons control the *pointer mode*, meaning how the mouse pointer works in SuperPreview. The first button changes the pointer mode to Selection mode. With Selection mode enabled, clicking inside the preview selects the element on which you click. This is useful to examining the specific properties of a page element.

The next button sets the pointer mode to Panning mode. With this mode selected, you can drag the page you are previewing so you can view areas of the page that don't fit within the confines of the preview window.

Pointer modes are mutually exclusive. In other words, clicking the Selection mode button deactivates Panning mode, and vice versa.

DOM Highlighting

When operating in Selection mode, clicking any page element selects that element. SuperPreview uses one of two highlighting modes to identify which DOM element is selected. Highlighting modes are selected using the Box Highlighting Mode and the Lights-Out Highlighting Mode buttons, as shown previously in Figure 13.2.

When Box Highlighting mode is selected, clicking a DOM element draws a border around the element. When Lights-Out Highlighting mode is selected, clicking a DOM element highlights the selected element and causes all other DOM elements to appear dimmed, as shown in Figure 13.3.

**tip**

When you change the DOM Highlighting mode, your pointer mode automatically switches to Selection mode.

UI Helpers

The next section of the toolbar contains three user interface helpers. The Toggle Ruler Visibility button toggles the display of the rulers along the edges of the preview window. The Toggle Thumbnail Visibility button toggles a small thumbnail view of the page in the lower-right corner of each preview. The Toggle Guide Visibility button toggles the display of the guides.

Figure 13.3
Lights-Out
Highlighting
mode lets you
easily identify
exactly which
DOM element
is selected. In
this figure, the
Welcome banner
is selected.



Layout Modes

The Layout Modes section of the toolbar contains buttons to control how your previews are laid out.

The Vertical Split Layout button shows the baseline and comparison browser previews side-by-side. This is the default setting. The Horizontal Split Layout button displays the baseline and comparison browser previews one on top of the other. The Overlay Layout button provides a view of the baseline browser superimposed on top of the comparison browser as shown in Figure 13.4.

Finally, the Single Layout button allows you to view either the baseline browser or the comparison browser in its own window.

Preview URL

The URL you are previewing is specified using the Location text box. However, you can also click the ellipsis button and select a file on your local disk to preview. This is useful in cases where you want to view a preview of a file in a disk-based site before publishing it to the Internet.



tip

Another quick and easy way to preview a page from a disk-based site in SuperPreview is to right-click the file in Expression Web's Folder List panel and select Display in SuperPreview from the menu. The benefit to this method is that ASP.NET and PHP pages display using the Microsoft Expression Development Server so you see the page the way viewers of your site will see it.

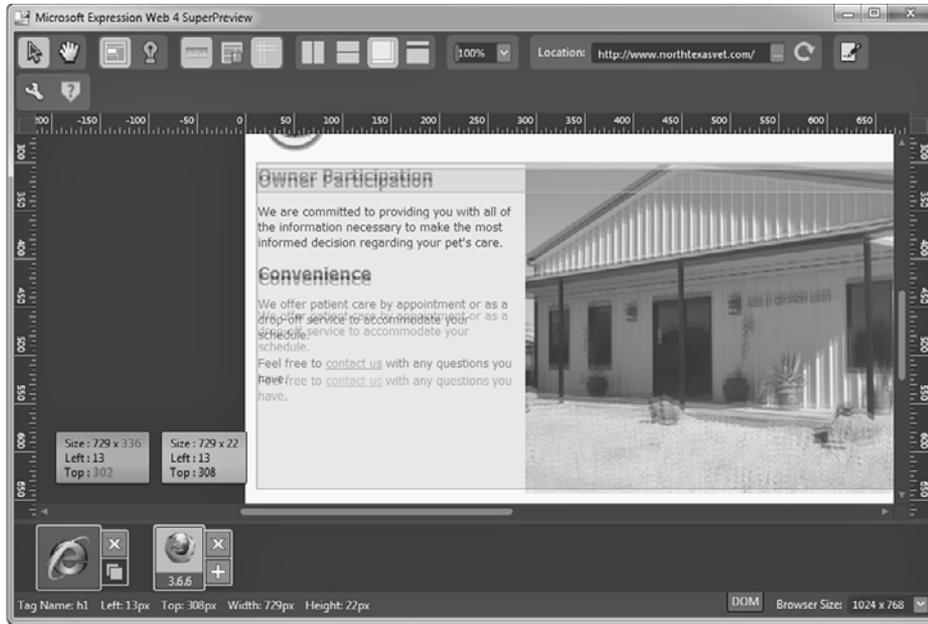


Figure 13.4 Overlay Layout is perhaps one of the most powerful and impressive features in SuperPreview.

Baseline and Comparison Browser Selectors

The baseline and comparison browser selectors allow you to choose which browser you want to use for your baseline browser and comparison browser. Each has a Close button in the upper-right corner so you can choose a different browser if you want. Additionally, the baseline browser selector offers a button under the Close button that alternates the baseline browser between the left and right preview panes. The comparison browser selector provides an Add button underneath the Close button that makes choosing multiple browsers as a comparison browser easy.

If multiple browsers are selected as the comparison browser, you'll see multiple browser selector buttons available for the comparison browser. In Figure 13.5, both Internet Explorer 8 and Internet Explorer 6 are being used as a comparison browser. To switch the preview pane between IE8 and IE6, simply click the corresponding version on the comparison browser button.

DOM Tab

The DOM tab expands a window at the bottom of the SuperPreview window that shows the DOM tree for the currently previewed page. Using the DOM tab, you can easily locate the actual DOM element that is responsible for any rendering irregularities.

Figure 13.6 shows the DOM tree expanded after clicking the DOM tab. Notice that the currently selected element in the preview window is also selected in the DOM tree.

Figure 13.5 Multiple browsers can be chosen for the comparison browser. You can easily switch between them by clicking the appropriate button.

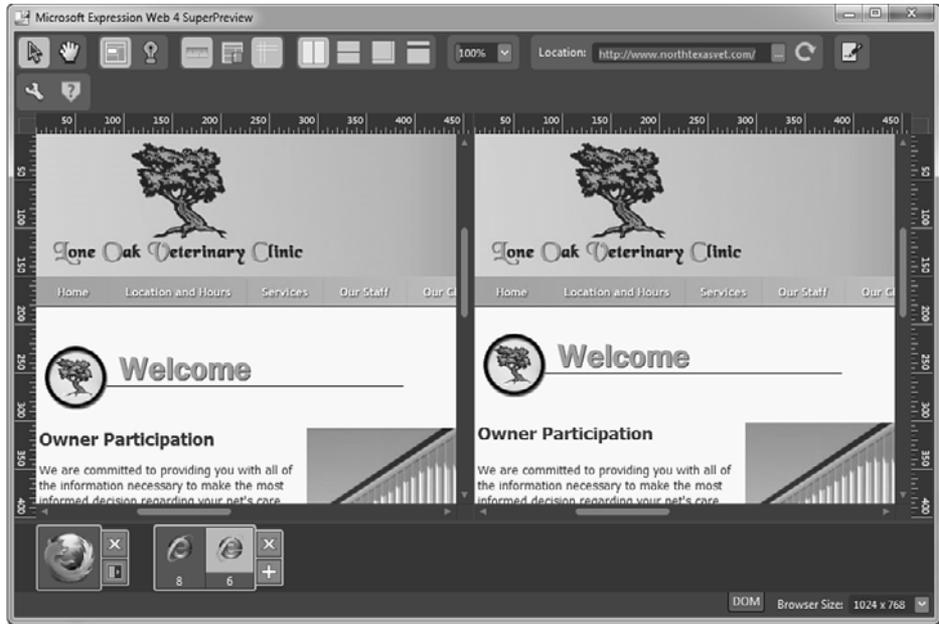


Figure 13.6 The DOM tree provides a quick and easy way to identify the code responsible for rendering problems.



Browser Size Drop-Down

The Browser Size drop-down contains a list of popular resolution settings and allows you to preview your page using whatever resolution you choose. If your desired resolution is not listed, you can choose Custom Size from the drop-down as shown in Figure 13.7, and you can then define your own size.



Figure 13.7

The Browser Size drop-down makes previewing your page in a specific resolution as simple as a click.

After you select a new browser size, SuperPreview regenerates any previews it is currently displaying.

Now let's use SuperPreview to investigate a rendering problem in a page so you can begin to fully realize the power and potential of this tool.

Using SuperPreview to Preview Layout

You've probably already realized that SuperPreview can be an incredibly valuable tool when you know that your page renders differently in a specific browser version. However, what you might not realize is that SuperPreview can help you identify subtle rendering differences you might not have even been aware of. Let's look at just such an example as we discover how SuperPreview can help to identify why a page renders slightly differently in two different browsers.

Setting Up the Previews

The first thing you need to do when using SuperPreview is choose your browsers. For this particular test, we're going to select Firefox 3.6.6 as the baseline browser and Internet Explorer 8 in IE7 compatibility mode as the comparison browser. To do this, simply click Firefox 3.6.6 in the left preview pane and Internet Explorer 8 → 7 in the right preview pane. Your SuperPreview dialog should look like the one shown in Figure 13.8.

note

To select Firefox 3.6.6, you must have Firefox 3.6.6 installed. To select Internet Explorer 8 → 7, you must have Internet Explorer 8 installed.

Figure 13.8 SuperPreview is now configured with Firefox 3.6.6 as the baseline browser and Internet Explorer 8 in IE7 mode for the comparison browser.



Generating Previews

In the Location box, enter **www.jimcobooks.com** and press Enter. SuperPreview generates a preview of the page using each of the browsers that were selected as shown in Figure 13.9.

You might immediately notice that the banner ad on this page is a little off between the two previews. However, you may not notice many other small rendering differences. To see these, click the Overlay Layout button. After you do, you'll see that several page elements on this page are positioned differently in Firefox 3.6.6 and in IE8 running in IE7 compatibility mode.

Fixing Rendering Problems

First, the banner ad on the page appears to be slightly lower on the page in Firefox than in Internet Explorer. To get a better idea of what might be causing this problem, switch to Lights Out Highlighting mode and click the banner ad. The SuperPreview window should look like the one shown in Figure 13.10.

The additional whitespace at the top of the banner in Firefox is obvious, but notice that the Size, Left, and Top measurements displayed in SuperPreview are identical in both browsers. That tells you that the rendering problem is not with the banner ad itself, but likely with another element's position in relation to the banner ad.

note

It might interest you to know that I identified this rendering problem and how to resolve it while writing this chapter using SuperPreview. However, I left it broken so you would be able to investigate the issue while reading this book.



Figure 13.9
This SuperPreview example shows both an obvious rendering problem and some problems that aren't immediately apparent.



Figure 13.10
Locating rendering problems is made easier with the combination of Lights Out Highlighting mode and Overlay Layout.

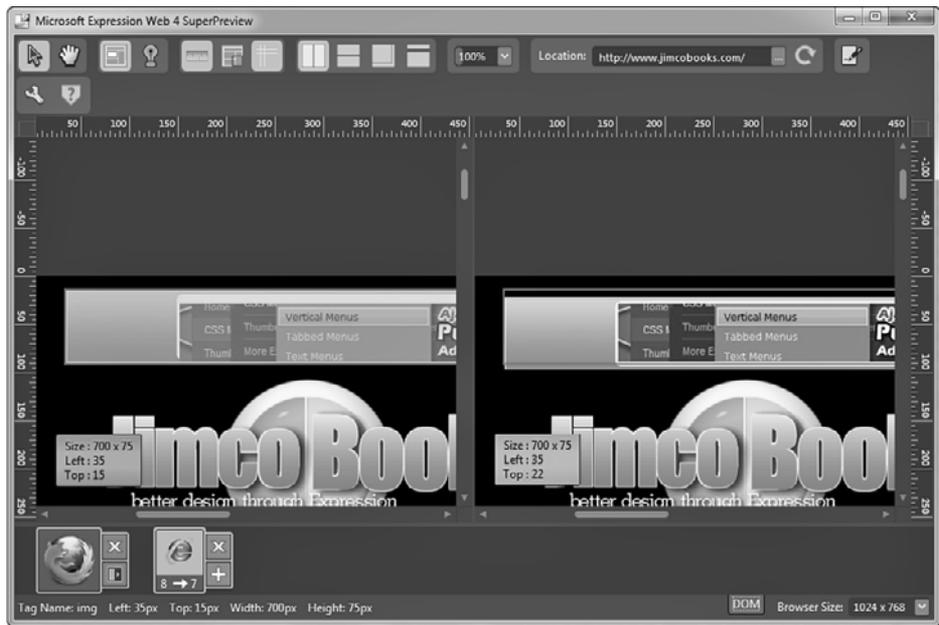
Click anywhere on the page other than on the banner ad to deselect it. Click the orange graphic that surrounds the banner ad to select it. Notice now that the Top measurement shows a difference between Internet Explorer and Firefox (see Figure 13.11).

We now know that the orange border around the banner ad is the source of the rendering problem, but to find out exactly what might be causing this problem, we need to investigate the code that underlies the page element. Fortunately, SuperPreview provides tools for investigating the DOM as well.

Click the DOM tab (shown previously in Figure 13.2) to display the DOM for the page as shown in Figure 13.12. Notice that the element we've selected is an `` element, but it's wrapped in a `<div>` with a class of `bannerAd`.

If you click the `<div>` in the DOM tree, you'll notice that the size of the `<div>` is different in Firefox than it is in Internet Explorer (see Figure 13.13). This is the source of the problem. By modifying the `bannerAd` CSS class and specifying height, width, and top CSS properties, this rendering problem can easily be corrected.

Figure 13.11
The Top measurement shows a difference between the two browsers.



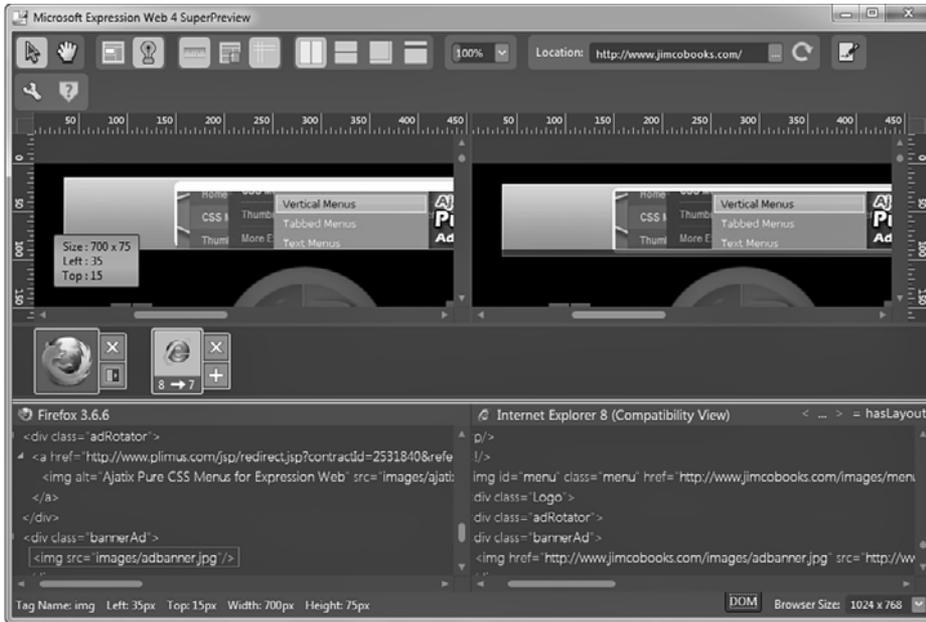


Figure 13.12 The DOM tab shows a synchronized view of the page's DOM so you can locate the code behind your preview.



Figure 13.13 SuperPreview lets you easily identify the source of this rendering problem. Here we see that the `<div>` is sized differently in Firefox and Internet Explorer.

As you can see by this example, SuperPreview offers some incredibly powerful tools for identifying exactly why rendering problems exist. Some rendering problems aren't as easy to resolve as this one, but identifying the source element of the problem is often one of the most difficult steps in resolving rendering issues, and SuperPreview provides power and flexibility web designers have not had before.

Using the Snapshot Panel

In some cases you might want to preview a page quickly in a particular browser. In such cases, you can use Expression Web's Preview in Browser feature, but if you want a quick preview without having to take the extra step of previewing in your browser, the Snapshot panel offers a great solution. Using the Snapshot panel, you can see a preview using the SuperPreview preview engine inside a panel integrated into the Expression Web interface.

To activate the Snapshot panel, select Panels, Snapshot. By default, the Snapshot panel is displayed at the bottom of the Expression Web interface, as shown in Figure 13.14, but you can drag the panel and dock it on any side just as you can with any other panel in Expression Web.

You can switch between different browsers in the Snapshot panel by selecting your desired browser from the drop-down in the upper-left of the Snapshot panel.



tip

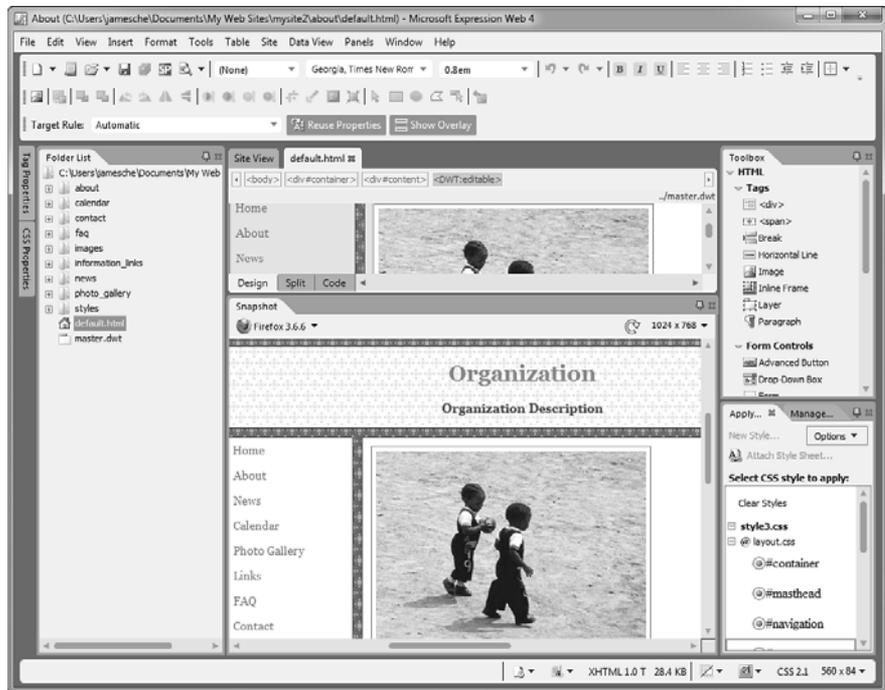
SuperPreview even supports .psd files (Adobe Photoshop files) when you click the Open Image button, so there's no need to export your Adobe Photoshop mock-ups to another format before using them with SuperPreview.



note

The Snapshot panel works only with local sites. If you have a remote site open using FTP or HTTP, the Snapshot panel will not work.

Figure 13.14
The Snapshot panel offers a preview using the SuperPreview engine right within the Expression Web interface.



If you have Service Pack 2 or later installed, the SnapShot panel offers an interactive mode that allows you to preview your page along with any dynamic content. In other words, if your page contains a JavaScript menu, the menu works in the SnapShot panel exactly the same way it works in the browser. However, there is a catch. Interactive mode is only available for the version of Internet Explorer that is installed on your system and for Firefox version 3.x.

Using Remote Browsers

One of the new features in Expression Web 4 is the remote browser functionality of SuperPreview. This feature currently provides a preview using Firefox 4, Firefox 5, Safari 4, Safari 5, and Chrome.

To use the remote browser feature, you must first activate your computer by clicking the Sign Up For SuperPreview Online Service button on the toolbar, shown previously in Figure 13.2. When you do, you are asked to enter your email address, to which Microsoft sends an activation link. Once you receive the email, click the link in the email message to validate your email address. You can then click Activate in the Online Service Activation dialog as shown in Figure 13.15.



tip

Another advantage of the Snapshot panel is the capability to preview pages in IE6 without having IE6 installed.



note

The SuperPreview Online Service is not available in the trial version of SuperPreview.

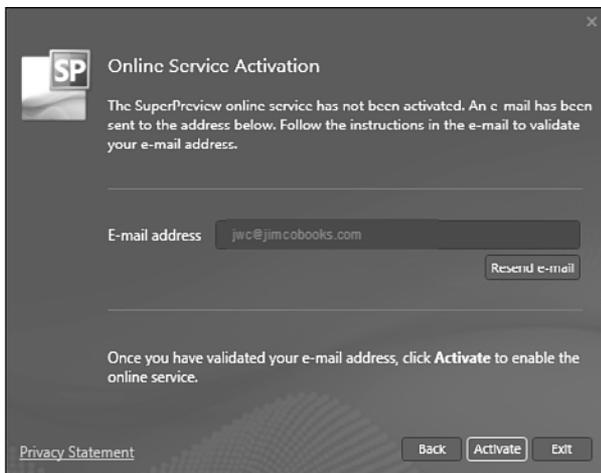


Figure 13.15

The Online Service Activation dialog is used to activate the SuperPreview remote browser feature.

After you've activated the remote browser service, you'll see Safari (Mac) listed in the Remote Browsers section of SuperPreview as shown in Figure 13.16.

You can control online service options by clicking the Options button on the toolbar, shown previously in Figure 13.2. The SuperPreview Options dialog (shown in Figure 13.17) makes it possible to check the status of the SuperPreview online service and to deactivate or delete your online service account. You can also choose to package web pages for previewing in remote browsers.



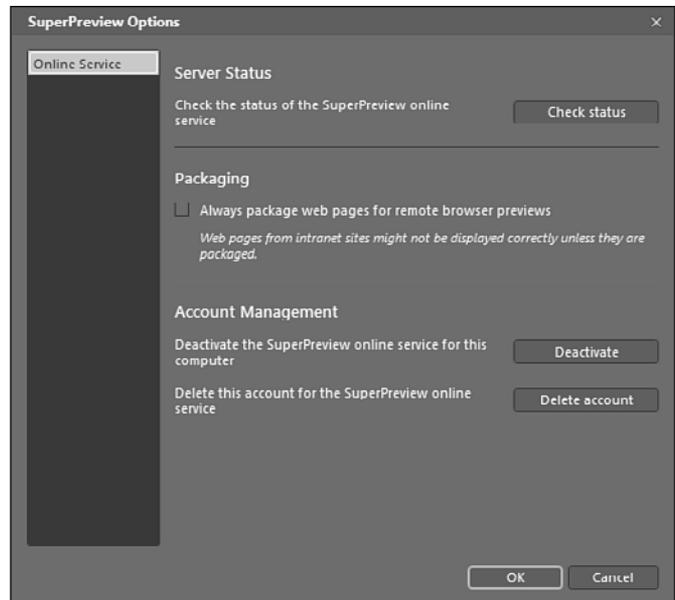
tip

You can toggle whether the remote browser service is available using the Enable or Disable Preview with Remote Browsers toolbar button.

Figure 13.16
Additional browsers appear as a choice once you've activated the remote browser service.



Figure 13.17
The SuperPreview Options dialog provides settings for the SuperPreview Online Service.



Packaging web pages for remote previewing is important in cases where the online service might not have access to files necessary to build a preview of your page. For example, if your site exists on an intranet site or on your local computer, the online service will likely not have access to images, CSS files, and so on that are necessary for building a preview. By checking the Always Package Web Pages for Remote Browser Previews check box as shown in Figure 13.17, you can ensure that your previews are accurate.

As you've seen in this chapter, the powerful preview tools offered by SuperPreview substantially reduce the pain involved in dealing with designing for multiple web browsers. As Microsoft continues to evolve this product, I have no doubt but that we'll see even more amazing capabilities for web designers.

Building Layouts with SuperPreview

One of the more common methods of web design is to build a mock-up of a site in a tool such as Expression Design or Adobe Photoshop and then to develop the code and CSS necessary to replicate that mock-up. SuperPreview can be an important part of ensuring that the code you are creating is cross-browser compliant.

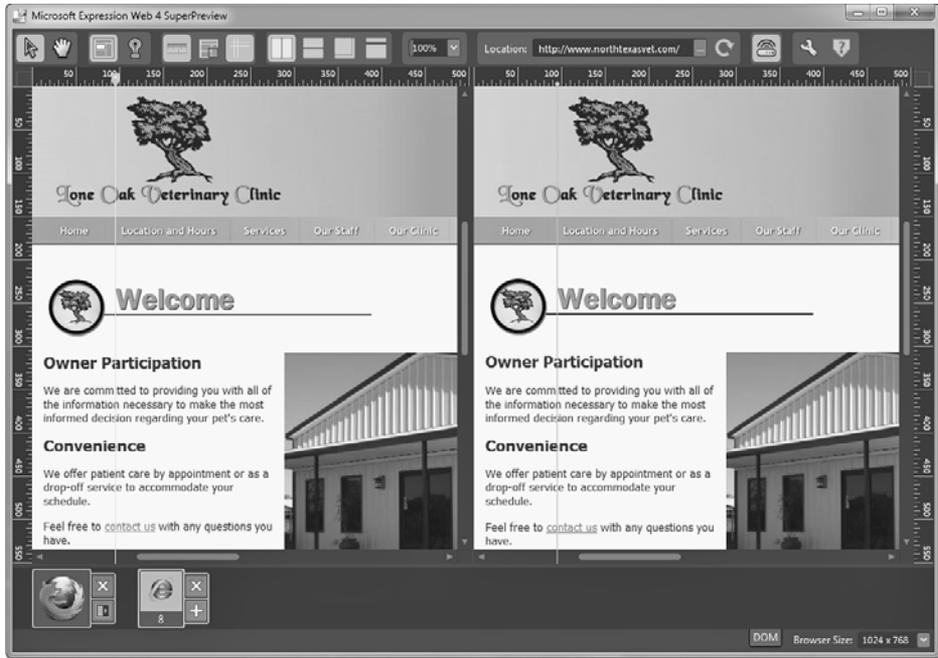
To use SuperPreview to assist in building sites that use a mock-up, click the Open Image button in one of the preview panes and browse to your mock-up. Next, select your desired browser in the other SuperPreview preview pane and then switch to Overlay Layout. Your browser preview will be superimposed over your mock-up, and you'll be able to quickly and easily determine whether your page is rendering correctly.

Another great feature in SuperPreview that aids in building layouts is the inclusion of Photoshop-like guides. If you click and drag from one of the rulers onto the preview area, a guide can be placed either horizontally or vertically on the page so that precise layouts are made easier. As shown in Figure 13.18, SuperPreview shows the current position of the guide as you are dragging.

To remove a guide, simply drag it back to the ruler. Alternatively, you can use the Toggle Guide Visibility button to hide any guides you've placed.

Using these tools, SuperPreview stands poised to revolutionize how web designers troubleshoot and resolve rendering issues between browsers, and I think it's one of the most exciting new features in Expression Web.

Figure 13.18
Photoshop-like guides make checking positioning of precise layouts easy.



This page intentionally left blank

PUBLISHING A SITE

What Is Publishing?

Even the best site is worthless unless people can see it. You can be the cream of the web designer crop, but if no one can access your site, it doesn't matter a bit. Therefore, you have two choices when it comes to designing a site with Expression Web: You can either design the site live on a web server that hosts your domain, or you can create your site offline and then publish it to a live server.

Developing a site offline and then publishing it to a live server is always the preferred choice. If you develop directly against a live site, any problems you encounter while developing your site (and there will be some) are visible to everyone who visits your site. You could, of course, put up one of those graphics that you see on many sites that say "Under Construction," but the way I see it, if a site is under construction, it should be on a development computer, not on a live web server. After all, an artist doesn't paint a masterpiece while the canvas hangs on a museum wall. A professional web designer always develops a site offline and then publishes it to the Internet or an intranet.

Publishing is simply the process of copying web content from one place to another. It might be content copied from one directory to another on the same machine, one site to another on the same machine, or from one machine to a different machine. You can also publish content to any of the numerous storage devices available today, such as a removable hard drive, a Universal Serial Bus (USB) key, or even a portable music player or your cell phone! If you can copy a file to a device or location, you can publish to it.

Server Options for Publishing

As shown in Figure 14.1, Expression Web offers many options for publishing content:

- FTP/SFTP/FTPS
- FrontPage Server Extensions (via HTTP or HTTPS)
- WebDAV
- File System

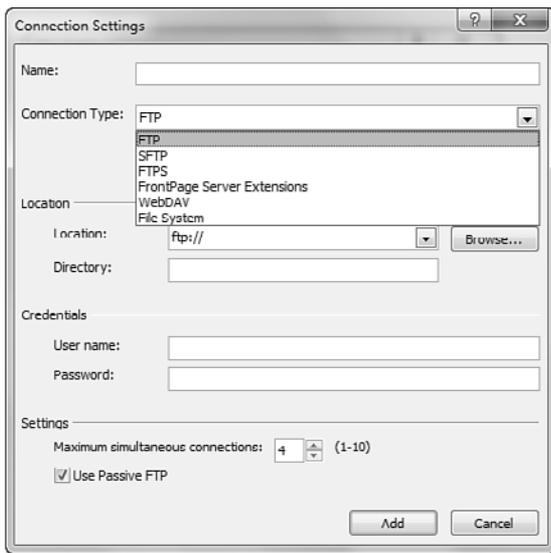


Figure 14.1
The Connection Settings dialog's Connection Type drop-down enables you to select your publishing method.

One of these methods is sure to be suitable for your purposes. In fact, you'll likely have a choice between a couple of them, so you'll need to understand the pros and cons of each option to make an educated decision when the time comes to publish your site.

FTP

FTP has been around a long time and is widely used by many web developers for transferring web content. In fact, the first FTP standard appeared in 1971, long before the advent of the World Wide Web.

FTP is generally not secured. That means someone can intercept your username and password. However, Expression Web also supports the use of both FTPS and SFTP so that you can publish securely.

FTP is a reliable method of transferring files between computers. However, there are some things to keep in mind when using FTP. The two types of FTP connections are passive FTP and active FTP.

note

Check with your hosting company for FTPS or SFTP support. Not all hosting companies offer both, and many hosting companies offer neither.

If you experience trouble when using FTP (especially when publishing through a firewall), enabling passive FTP usually resolves the issue.

To enable passive FTP in Expression Web, select either FTP or FTPS in the Connection Type dropdown and check the Use Passive FTP check box as shown previously in Figure 14.1.



Content Missing After Publishing

If you've published your content using FTP but nothing appears when you browse to your site, you haven't lost everything. Chances are you just uploaded information to the wrong directory. Your host has a specific directory that contains your site's files. Most of the time, you are automatically placed into that directory when you log in. However, in some cases, you need to specify the directory within Expression Web in the FTP Directory text box.

It's also possible that the name of your home page (that is, `index.htm`) might not be correct for your site. If you don't use the correct home page name for your site, you end up with an error if you try to browse to your site without specifying a filename.

Ask your host for the directory and the home page name you should be using when publishing.

FrontPage Server Extensions

The FrontPage Server Extensions are a set of files that is installed onto a web server to add functionality to a site. The FrontPage Server Extensions have been around for years. However, since Microsoft discontinued FrontPage, some hosting companies are beginning to stop support for the FrontPage Server Extensions.



note

Microsoft still supports the FrontPage Server Extensions on Windows Server 2003. Microsoft also engaged Ready to Run software to develop and support a version of the FrontPage Server Extensions for Windows Vista and Windows Server 2008. FrontPage Server Extensions are not supported at all on Windows 7 or on Windows Server 2008 R2.

When to Choose the FrontPage Server Extensions

Sometimes you won't have any choice but to use the FrontPage Server Extensions. If you can't connect to a remote server using one of the other connectivity methods provided by Expression Web, you have to use the FrontPage Server Extensions. You also need to use the FrontPage Server Extensions if you are using an Expression Web component (such as the built-in form handlers) that requires FrontPage Server Extensions for functionality on the server.

Outside those situations when you have no other choice, I recommend that you do not use the FrontPage Server Extensions. Although a version of the FrontPage Server Extensions is available for IIS 7, Microsoft did not develop it and does not support it.

The FrontPage Server Extensions use either the HTTP or secure HTTP (HTTPS) protocol to publish site content. These same protocols are used when browsing sites. Because the FrontPage Server Extensions use the same protocols as your web browser, they can easily publish to remote servers even if you are behind a firewall. Almost all firewalls are configured to allow HTTP and HTTPS traffic to freely pass through. The FrontPage Server Extensions take advantage of that fact when publishing content.



caution

Some Expression Web web components rely on the FrontPage Server Extensions to operate. If you are using any of these components, you need to select the FrontPage Server Extensions option when publishing.



Username and Password Not Accepted

The FrontPage Server Extensions recognize different levels of access to a site. If you are entering the correct username and password information and publishing has never worked, it's possible that your host has not granted you proper permissions. Contact your host and tell them to ensure you have permission to author against your site.

The connection settings in Figure 14.2 show the options available when choosing to publish a site using the FrontPage Server Extensions.

Connection Settings

Name: Jimco Production Site

Connection Type: FrontPage Server Extensions

The remote site server supports FrontPage Server Extensions or SharePoint Services.

Location

Location: http://www.jimcooftware.com Browse...

Credentials

User name: jim

Password:

Settings

Use Encrypted Connection (SSL)

Add Cancel

Figure 14.2
The Connection Settings dialog with settings for FrontPage Server Extensions shown.

After the FrontPage Server Extensions have been installed on the web server, they must be configured for your site by an administrator of the web server. During the configuration process, the FrontPage Server Extensions add folders to your site that begin with `_vti`. These folders contain

configuration information that the FrontPage Server Extensions use to keep track of changes to your site.

The FrontPage Server Extensions offer functionality at both design time and runtime. In addition to offering the capability to connect to remote sites and publish content, they also provide components that generate HTML and client script while you are developing the site, as well as components that provide special functionality when the site is browsed.

To publish with the FrontPage Server Extensions, you need to have the FrontPage Server Extensions installed on the remote site. You don't need to install the FrontPage Server Extensions on the local site.

 **note**

Expression Web adds `_vti` folders to your site if you are working with a disk-based site. While the `_vti` folders used with a disk-based site serve the same purpose, they are not actually FrontPage Server Extensions files.



FrontPage Server Extensions Aren't Installed

If your host promises you that the FrontPage Server Extensions are installed, but when you try to publish you keep getting a message saying that the FrontPage Server Extensions aren't installed, the permissions on the remote site might not be set correctly. This can prevent Expression Web from being able to tell whether the FrontPage Server Extensions are installed. If Expression Web can't tell that the FrontPage Server Extensions are installed, it assumes they aren't.

Use Wireshark or Fiddler and see what you can turn up. If you see something that looks out of the ordinary, send it to your host.

➔ For more information on using Fiddler, see “Using Fiddler to Troubleshoot HTTP Publishing Errors,” p. 249.

➔ For more information on using Wireshark, see “Using Wireshark to Troubleshoot HTTP Publishing Errors,” p. 250.

When you publish with FrontPage Server Extensions, you will usually publish to the same address you would use when browsing the remote site in your web browser. For example, if I were publishing a local copy of my site from `c:\WebSites\JimcoSoftware` to my public site using the FrontPage Server Extensions, I would enter a destination address of `www.jimcosoftware.com` as shown previously in Figure 14.2.

WebDAV

WebDAV is really just an extension of the HTTP protocol and therefore shares many of the same benefits offered by the FrontPage Server Extensions. In fact, the options available in the dialog for publishing a site in Expression Web via WebDAV are identical to those available when publishing using the FrontPage Server Extensions.

There are many benefits to using WebDAV:

- Firewalls generally are not an issue because WebDAV extends HTTP, the protocol used when browsing the Web.
- WebDAV supports strong authentication so you can ensure the security of your data.
- WebDAV supports encryption for another layer of protection for your data.
- WebDAV includes support for file versioning and resource locking so that one user won't overwrite another user's work.

WebDAV is already a part of your operating system. In fact, when you create a shortcut to an HTTP location in My Network Places, Windows uses WebDAV to create that link if the FrontPage Server Extensions are not installed on the remote server.

To use WebDAV to publish your Expression Web site, your hosting company or system administrator must provide support for WebDAV. I have had great success using WebDAV on IIS 7 and Expression Web, but Microsoft's implementation of WebDAV isn't complete. Even so, you can open a remote site and publish your site using Microsoft's implementation of WebDAV and Expression Web.



note
Many people in the technical community seriously believe that WebDAV will eventually replace all the publishing protocols in wide use today. In fact, Microsoft has released a WebDAV authoring extension for IIS 7 that works perfectly with Expression Web.



FrontPage Server Extensions Are Used Instead of WebDAV

The FrontPage Server Extensions and WebDAV are mutually exclusive. That means that if the FrontPage Server Extensions are installed on the web server, you can't publish using WebDAV. As soon as you try to publish using HTTP, the web server expects you to use the FrontPage Server Extensions.

If you are getting an error about the FrontPage Server Extensions when publishing via WebDAV, it indicates that the FrontPage Server Extensions are installed on the site and you cannot use WebDAV to publish. Contact your host and let them know that you want to remove the FrontPage Server Extensions and use WebDAV instead.

File System

The File System method of publishing is used most often in situations where both the local and remote locations are on the same network. For this method to work, you must have a connection to the remote server that allows for access via a Universal Naming Convention (UNC) path.

A UNC path consists of a server name and a share name using the syntax `\\server\share`. You also can set up a mapped drive so you can refer to a UNC share using a drive letter. UNC paths and mapped drives use the same underlying architecture.



caution
Mapped drives are user-specific. If you log in to a computer and map a drive, other users who log in to that computer will not see that mapped drive. If you map a drive with the intention of allowing other users of the same computer to publish to the mapped drive, keep in mind that you need to map the drive for each user individually.

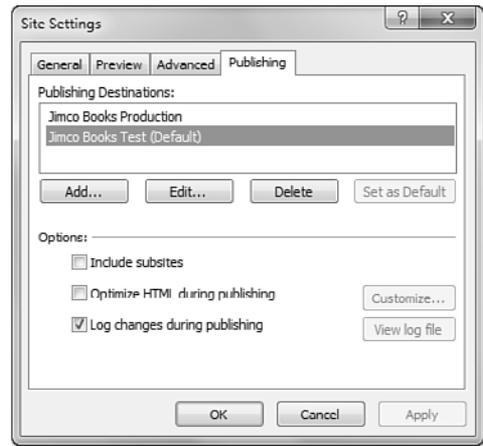
Publishing Content

Now that you have a rundown of all the methods available when publishing a site in Expression Web, let's look at how to publish content to a site.

Configuring a Publishing Destination and Publishing a Site

Before you publish your site, you need to configure one or more publishing destinations. Select Site, Publishing Settings to display the Site Settings dialog, as shown in Figure 14.3.

Figure 14.3
The Publishing tab on the Site Settings dialog lets you easily configure publishing destinations for your site.



Click Add to add a new publishing destination using the Connection Settings dialog shown previously in Figure 14.1. After you've entered the necessary information for your particular connection type, click Add.

Any publishing destinations you create are specific to the site currently opened in Expression Web. Therefore, you can easily create one destination for a test location to show clients or to test your site and another destination for the live site.

After you've created a publishing destination, select the Publishing tab to switch to Publishing view.

If you want to publish to the publishing destination selected in the Connect To drop-down, click the Connect to the Current Publishing Destination link. Otherwise, select the desired publishing destination and Expression Web connects automatically.

Once connected, Expression Web displays the currently opened site in the left pane and the destination site in the right pane (see Figure 14.4).



tip

The View drop-down makes seeing just the files you are interested in while in Publishing view easy.

Expression Web uses icons to indicate changed or unmatched files, as shown in Figure 14.5. A changed file is one that has changed since the last publish. An unmatched file is a file that exists in the current site and not in the destination site, or vice versa.

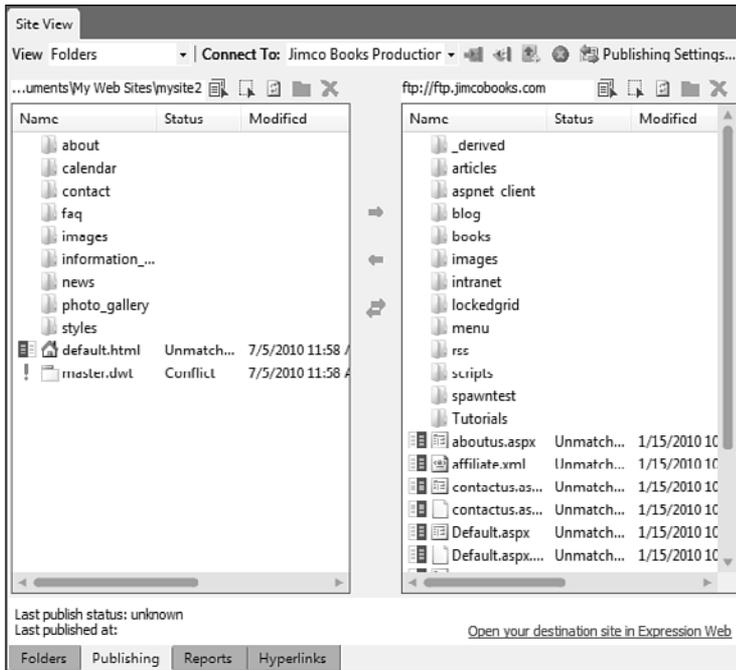


Figure 14.4

The Publishing view shows both the current site and the destination site side-by-side.

To publish changed or unmatched files, click one of the blue arrows in the border between the current site and the destination site. Clicking the arrow that points to the right publishes files to the destination site, the arrow that points to the left publishes to the current site from the destination site, and the arrow that points in both directions synchronizes the current site and the destination site.

While Expression Web publishes your site, it displays the current progress in the Publishing Status pane as shown in Figure 14.6. After publishing, you can use the Failed, Completed, and Log tabs at the bottom of the Publishing Status pane to view the details of publishing.

Figure 14.5

Expression Web uses icons to indicate unmatched or changed files. In this case, `aboutus.aspx` and `Default.aspx` are changed files and `news.aspx` is an unmatched file.

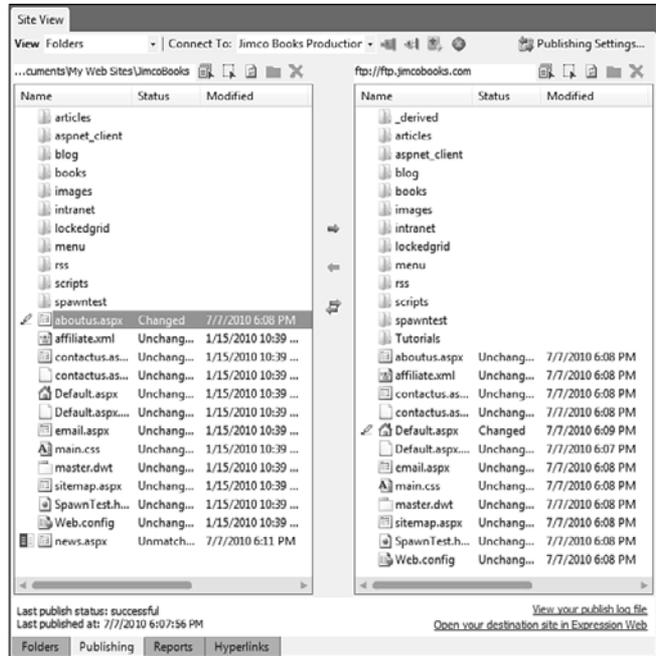
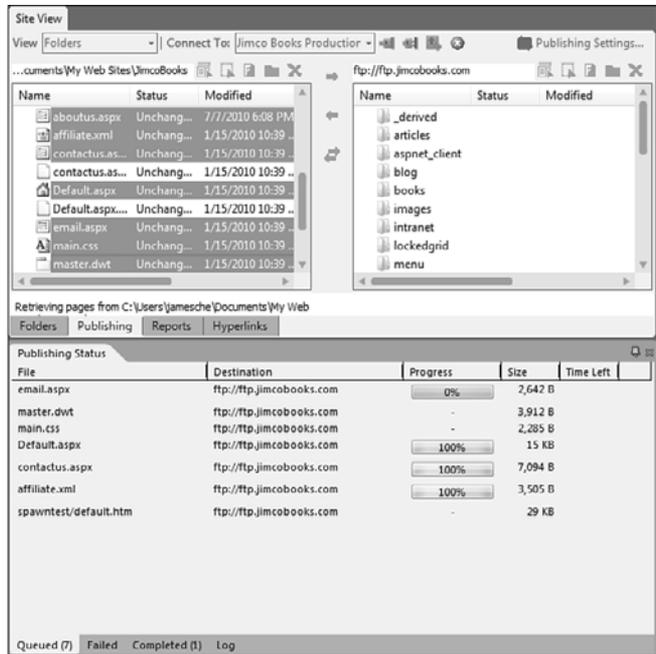


Figure 14.6

The Publishing Status pane shows the status of each file that is publishing.



Publishing Selected Files and Synchronizing Files

In addition to using the Publishing tab to publish files in your site, you can also use the context menu in the Folder List or in Site view.

After you select one or more files in the Folder List or in Site view, right-click and select Publish Selected Files (shown in Figure 14.7) to publish the file or files. Expression Web publishes to the destination that is selected on the Publishing tab.

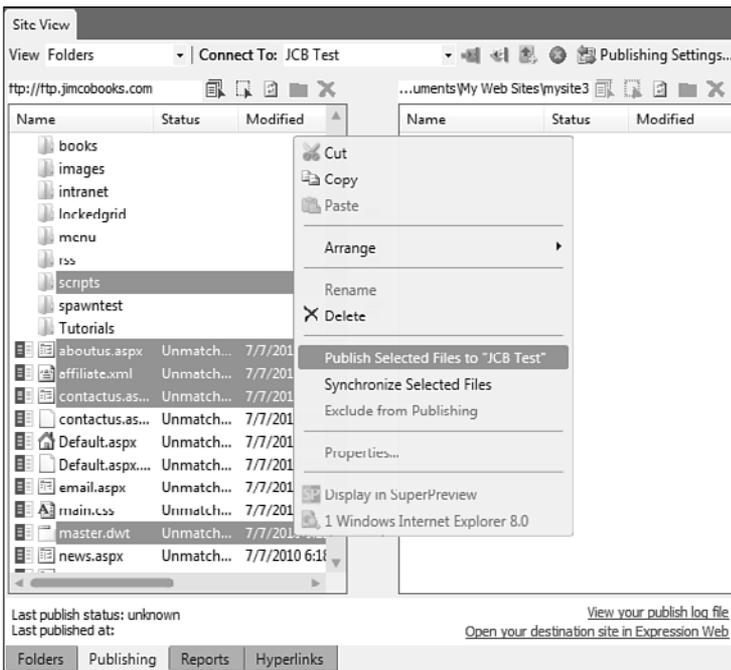


Figure 14.7

The Publish Selected Files menu option makes it easy to publish one or more files quickly.

If you select one or more files in the Publishing tab, you can synchronize the selected files by right-clicking and choosing Synchronize Selected Files from the context menu shown in Figure 14.7.

Optimizing HTML During Publishing

In addition to publishing your files, Expression Web can also optimize your HTML. The main purpose of this feature is to remove code that might not be needed on the remote web server.



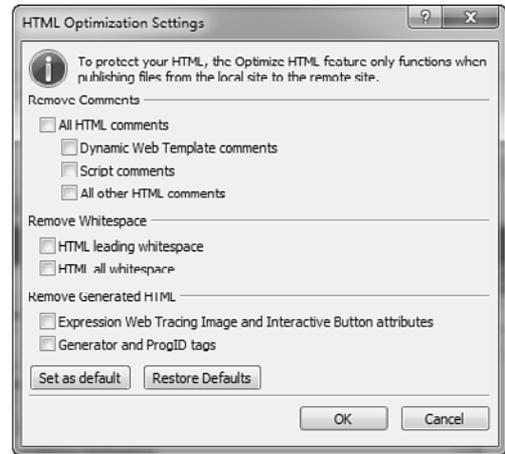
tip

You can also select one or more folders and use the context menu to publish or synchronize all of the files in the selected folders.

You can configure what type of HTML optimization takes place using the HTML Optimization Settings dialog as shown in Figure 14.8. To access the HTML Optimization Settings dialog, click Publishing Settings in the upper-right while in Publishing view and click the Customize button next to the Optimize HTML During Publishing check box shown previously in Figure 14.3.

Figure 14.8

Expression Web can remove any unnecessary code so your pages are as small as possible for better load times, but don't expect miraculous increases in speed with this feature.



Many options available for HTML optimization:

- **All HTML Comments**—Removes all HTML comments in code. HTML comments are designated by the `<! -` and `< / / - - >` characters. The closing comment character might also simply be `- - >`.
- **Dynamic Web Template Comments**—Expression Web adds comments to pages that use Dynamic Web Templates so editable regions can be identified. This option removes those comments.
- **Script Comments**—This option removes comments added to client-side scripts.
- **All Other HTML Comments**—This option removes comments that are not covered in any of the previous categories.
- **HTML Leading Whitespace**—Removes the leading whitespace from your lines of code.
- **HTML All Whitespace**—Removes all HTML whitespace, including whitespace between lines.
- **Expression Web Tracing Image and Interactive Button Attributes**—Removes attributes that are added for the purpose of tracing images and also removes interactive button attributes. These attributes are used only for editing purposes, so they aren't needed on live files.



tip

Many hosting companies that support the FrontPage Server Extensions for publishing also blame them for problems while publishing. This is another reason for choosing FTP over the FrontPage Server Extensions if you can. FTP problems are uncommon, but if you do have them, almost all hosting companies are competent in troubleshooting them.

- **Generator and ProgID Tags**—Removes the Generator and ProgID META tags that are added by Expression Web.



HTML Optimization Doesn't Affect Web Files

Many of the comments and other special code that gets cleaned out as a function of HTML optimization are actually needed by Expression Web while you are creating your pages. Therefore, Expression Web optimizes the HTML only when you publish your site. The local files remain unchanged.

Troubleshooting HTTP Publishing

Rule number 1 of publishing is this: If you're going to have a problem with a hosting company, it's going to happen when you are publishing your content. Most hosting companies do a great job of ensuring that people can always browse your site, but publishing is more complex and requires a more precise configuration.

The second rule of publishing is that most hosting companies will deny responsibility when it comes to publishing problems. That means you are going to have to do your own troubleshooting.

To troubleshoot publishing problems, you need to install software that can capture information between your computer and the remote computer. The two tools I use for doing this are Wireshark and Fiddler, both of which are free tools and available for download on the Internet.

Wireshark is a network protocol analyzer (see Figure 14.9). That's a fancy way of saying it captures all traffic that flows to and from the network card on your computer. It features a great filter so you can zero in on just the traffic you want to see. When dealing with Expression Web publishing issues, you're going to be interested in HTTP or FTP traffic. Wireshark is available from www.Wireshark.org.



note

We only cover HTTP publishing in this section. However, if you're using WebDAV or the FrontPage Server Extensions to publish, the information presented here will be helpful in troubleshooting any problems that might occur.



note

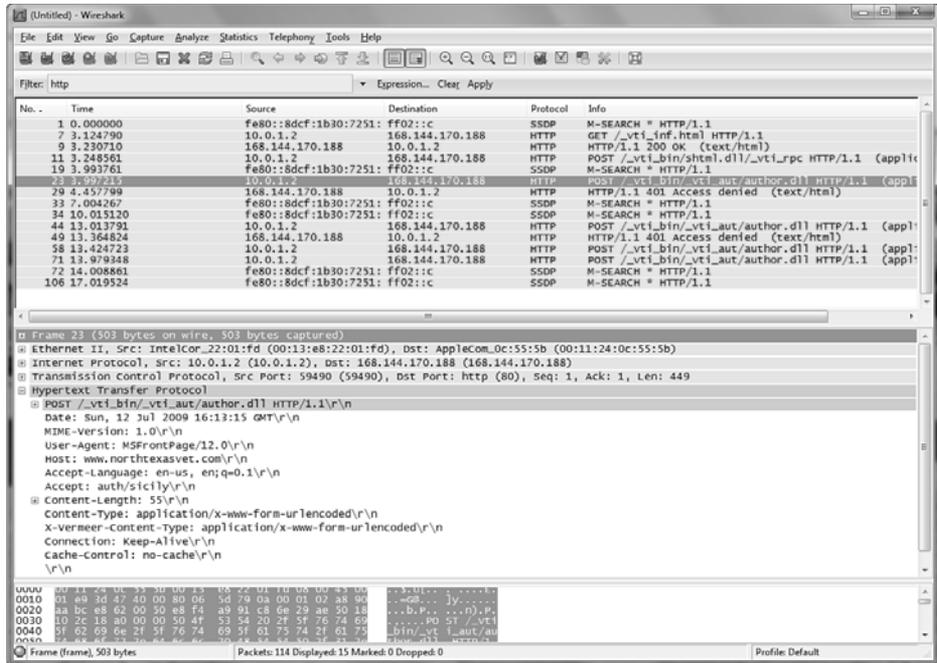
None of the techniques I discuss in this section work if you are using SSL (HTTPS) because all traffic that flows over an SSL connection is encrypted. You'd be able to see the traffic, but you wouldn't be able to decipher it into anything meaningful.



tip

A 401 status code means access was denied. You can see a full list of HTTP status codes at <http://support.microsoft.com/kb/318380/en-us>.

Figure 14.9 Wireshark is a powerful network capture tool that is free and available via download from www.wireshark.org. Here I've captured the network traffic while opening a site using the FrontPage Server Extensions.



The other tool I use is Fiddler (see Figure 14.10), which is an HTTP debugging proxy. That's a fancy way of saying it logs all HTTP traffic on your machine. Fiddler has the added benefit of being able to log local traffic that doesn't travel over your network card. Fiddler is available from www.fiddlertool.com.

HTTP Authentication Traffic

Most of the problems you'll encounter during publishing are authentication problems, meaning that the remote server doesn't accept your username and password. To troubleshoot this type of problem, you need to first understand what happens when things are working correctly.

To publish a site, you need to be able to write content to the remote site. For the web server to verify that it should allow you to write to the content of the remote site, it needs to authenticate you. It does that by sending a 401 status code in response to your attempt to publish.



tip
In most cases, if Windows Integrated authentication fails, the web server automatically tries basic authentication. Chances are you won't even know if or when this happens.



note
Windows Integrated authentication is naturally Windows-specific. Basic authentication, however, is used in non-Windows web servers, such as Apache.

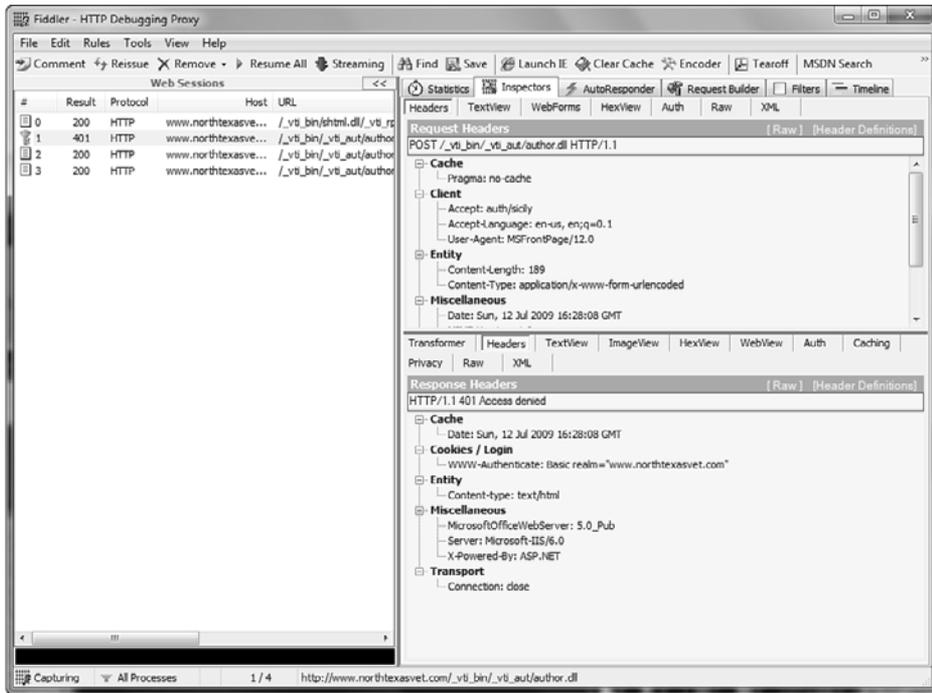


Figure 14.10 Fiddler targets HTTP traffic specifically. It's a great tool to use when troubleshooting publishing using HTTP.

The client (in this case, Expression Web) responds to the 401 status code in one of two ways, depending on the type of authentication being used. You'll generally use one of two authentication methods:

- **Windows Integrated Authentication**—Windows Integrated authentication is authentication that's built in to Windows itself. If your web server indicates that it can accept Windows Integrated authentication, Expression Web attempts to authenticate you using the user logged on to your computer. If that fails, you are prompted for a username and password.
- **Basic Authentication**—Basic authentication always prompts for a username and password.

Windows Integrated authentication uses encrypted data to authenticate you to the remote site. In fact, one of the benefits of Windows Integrated authentication is that it's a secure authentication mechanism. However, it was not designed to work on the Internet. In many cases, if your host is using Windows Integrated authentication, you'll be repeatedly prompted for a username and password and publishing will fail.

Basic authentication doesn't use encrypted credentials. Instead, your username and password are encoded using Base64 encoding before they are sent across the wire. Base64 encoding is a common encoding method used to transmit data across networks. It was actually developed for sending binary data, but it works well for sending text as well.

It's important to keep in mind that Base64 encoding provides nothing in the way of security. A quick Google search turns up scores of utilities for decoding Base64-encoded data. You shouldn't think of this as a flaw in the method used by basic authentication. It's actually the way it was designed to work.

Using Fiddler to Troubleshoot HTTP Publishing Errors

Fiddler is a convenient tool to use for troubleshooting HTTP publishing because it targets only HTTP traffic. To start Fiddler, select it from the Start menu in Windows or use the Fiddler menu item in Internet Explorer, as shown in Figure 14.11.

Figure 14.11
Fiddler installs a button into Internet Explorer so you can always access it quickly when you need it.

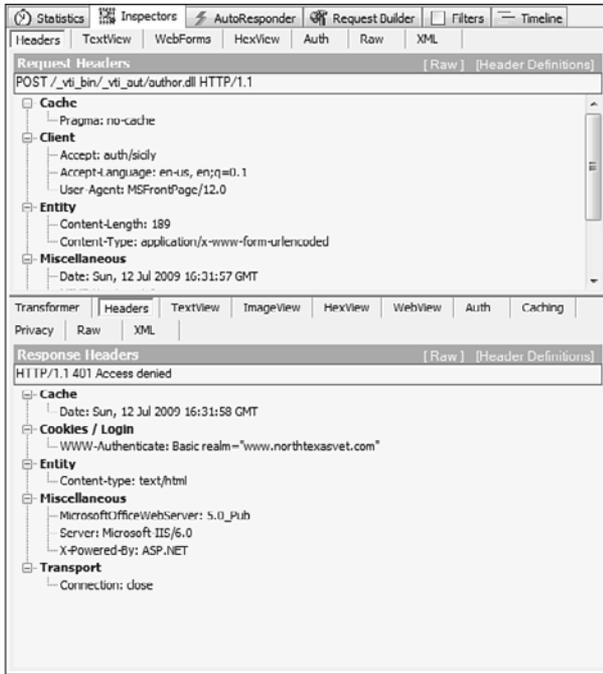


After you start Fiddler, it begins capturing HTTP traffic, as shown previously in Figure 14.10. At the right side of the Fiddler interface, you can see different views of the data. By clicking the Session Inspector tab, you can view the local computer traffic in the top half of the display and the remote traffic in the bottom half. A series of buttons appears above both panels, allowing you to change the format of the data that is displayed. When troubleshooting publishing issues, it's probably most helpful to click the Headers button for both panes, as shown in Figure 14.12.



tip

Because Wireshark captures all network traffic, it can also be used to troubleshoot FTP publishing. Fiddler captures only HTTP traffic and cannot aid in troubleshooting FTP.

**Figure 14.12**

Fiddler can break local and remote traffic into two panes. In this case, you can see that the server is using basic authentication.

You can use the information that Fiddler provides to help in troubleshooting publishing issues by analyzing the authentication information. You want to look for the 401 status sent by the remote server and select the line just below it in Fiddler, as shown in Figure 14.13. You should then see your local machine respond with the authorization information.

If you see that you are not sending authorization information to the server, you have a problem on your end. However, if you see that you are sending information to the server, you're either sending the wrong credentials or there's a problem somewhere other than on your end.

Using Wireshark to Troubleshoot HTTP Publishing Errors

Wireshark is a much more powerful tool than Fiddler because it captures all network traffic on your computer. However, with that power comes additional complexity. For that reason alone, I usually choose Fiddler over Wireshark for simple troubleshooting, but you might find that Wireshark offers features that are useful to you.

To use Wireshark, select Capture, Interfaces to display a list of network interfaces you can capture. Browse to any site and look for the interface that shows an increasing number of packets (see Figure 14.14). Then click the Capture button.

After you've started the capture, begin your publishing in Expression Web. When you encounter a problem, switch back to Wireshark and select Capture, Stop.

Figure 14.13
Fiddler clearly shows when you are sending authorization information to the remote server. I've intentionally blurred the authentication information here for security purposes.

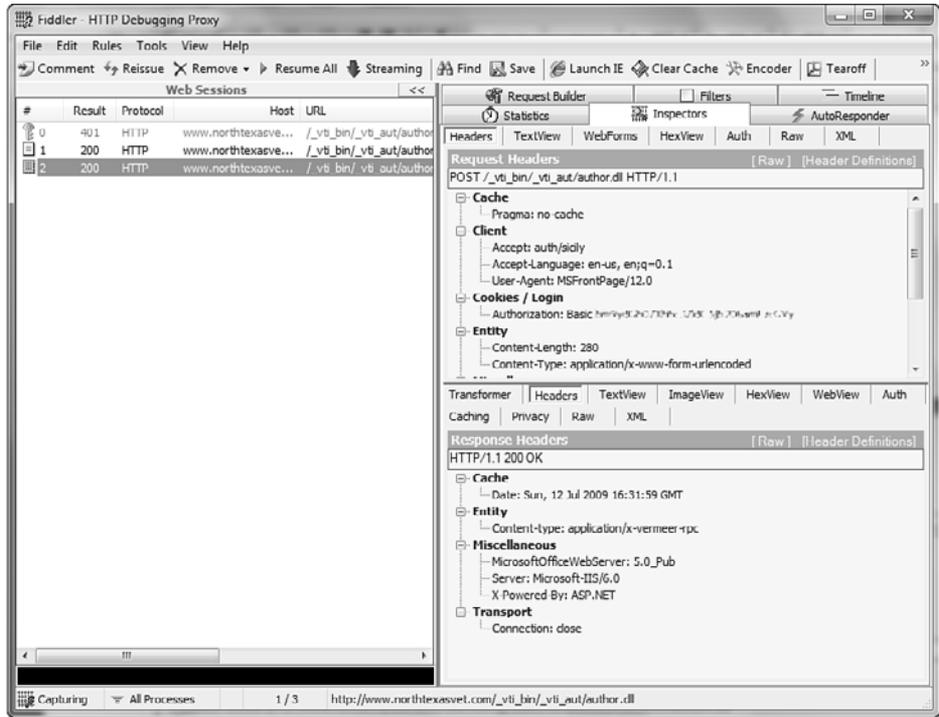
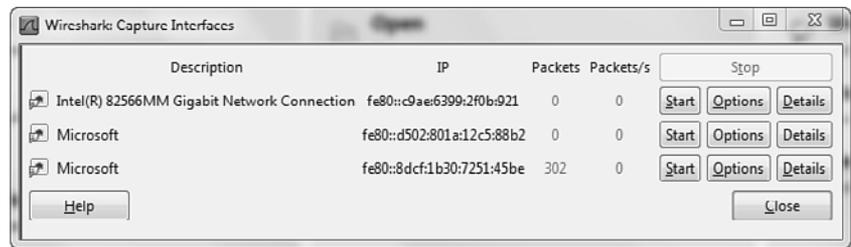


Figure 14.14
You might want to capture the interface that is connected to the Internet. Look for the interface with an increasing number of packets.



Wireshark Captures Nothing

You should be able to capture network traffic against your wireless network card, but you might need to disable promiscuous mode. In the Capture Interfaces dialog shown previously in Figure 14.14, click the Options button next to the interface you are capturing and then uncheck the Capture Packets in Promiscuous Mode check box in the Capture Options dialog.

Wireshark captures all traffic on the network, so there's a good chance it will capture a significant amount of data that is not related to your work in Expression Web. Fortunately, you can filter the captured data easily by simply entering **http** in the Filter text box, as shown in Figure 14.15.

The data is collected in Wireshark in a similar format to the data collected in Fiddler. Once again, you should look for Expression Web's response to the 401 status code sent by the web server.

note

As of IIS 7 and the release of FastCGI, running PHP on a Windows server running IIS 7 provides application performance that is comparable to the performance that PHP enjoys on Apache web servers.

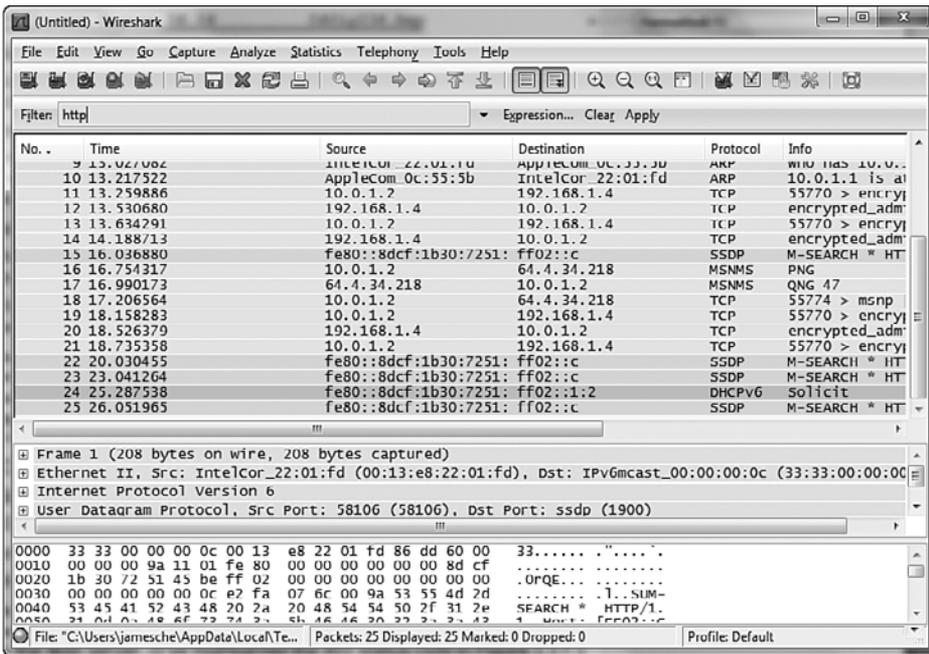


Figure 14.15
Filtering in Wireshark is a simple task using the Filter text box.

Another powerful feature of Wireshark is its capability to display a conversation between Expression Web and the remote web server in a readable format. By right-clicking any of the captured data and selecting Follow TCP Stream from the menu, Wireshark displays the entire conversation (see Figure 14.16). You can even save this information and send it to your host to demonstrate the problem.

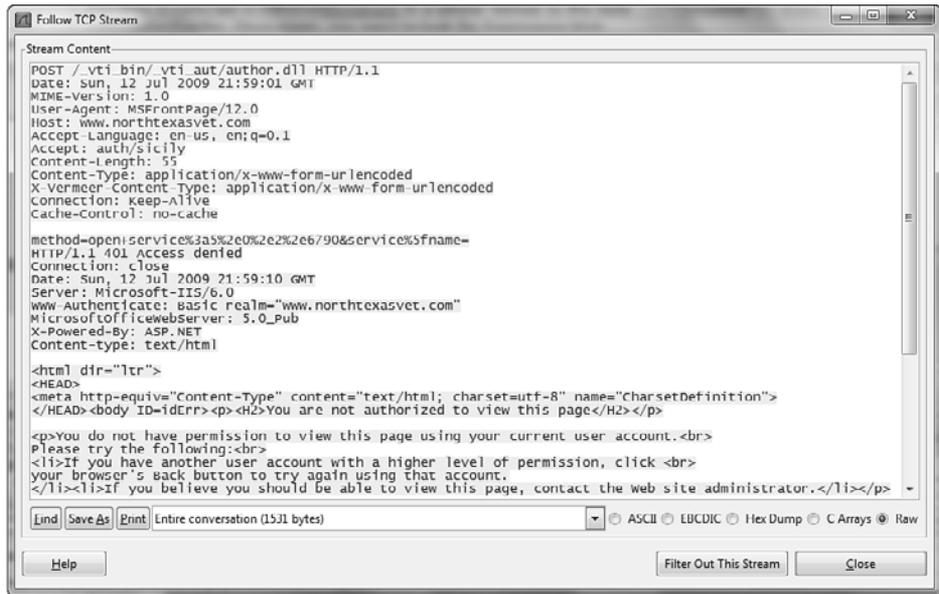
caution

If you decide to host your own site, check with your Internet provider to make sure that doing so isn't a breach of its terms of service.

 **note**

Windows Vista and Windows 7 also have a limited number of connections, but they're designed in such a way as to not generate errors when that limit is reached. Instead, they allow the user to wait in a queue until a connection is available. Because of this, it's possible to host a fairly busy personal site on Windows Vista or Windows 7 without any problems.

Figure 14.16
An enormous benefit to Wireshark is its capability to display captured data as an entire conversation.



By using the data captures from Wireshark or Fiddler, you should have all you need to isolate publishing problems and possibly convince a stubborn host that the problem isn't on your end.

Hosting Your Site

One of the most commonly asked questions by web designers is one that might appear to be simple: "What should I consider when choosing a host for my site?" Many options are available for hosting, and most of them are more affordable than you might think.

First, outline your requirements. For example, are you going to need a database? If so, do you need a high-performance system such as Microsoft SQL Server or a lower-level system? Most hosting companies offer support for SQL Server for a nominal monthly fee. However, if you don't need that kind of power, you can probably save some money by choosing a file-based database such as Microsoft Access. You also might want to investigate using SQL Server Express Edition. This high-performance, file-based system is available free from Microsoft, but keep in mind that some hosting companies do not support the Express Edition of SQL Server.

You also need to try to determine how much traffic you expect. Most hosting companies charge for a specific amount of bandwidth. After you surpass that bandwidth limit, you have to pay a premium for additional bandwidth. This is especially important if you plan to offer large files for download such as video or audio files.

Another important decision to make is whether to choose a hosting company that runs Windows servers. It seems to be all the rage these days to try to convince people to stay away from Windows Server. I encourage you to not fall for those who push their own agenda without any factual information. Make your decision based on the technology you want to use. If you want to use ASP.NET, you need a host that runs Windows Server. If you want to use PHP, you have a choice of a Windows or non-Windows host.

If you just want to host a simple site without any fancy features and you do not expect a lot of traffic, you might want to consider hosting the site yourself. If you're a technology guru, you can get Linux and Apache running on a spare computer lying around the house and get good performance. Setting up Internet Information Services (IIS) on Windows Server is easier, but it requires a more powerful computer.

You can use Windows XP Professional, Windows Vista, or Windows 7 Home Premium Edition or better to host your own site, but these have a limitation on the number of connections that can be serviced. Keep in mind, too, that a single person browsing a site is going to be opening multiple connections. Chances are you will reach the connection limit at two or three people.

If you do choose to host your own site, you might want to invest in a router that has Dynamic Domain Name System (DNS) capability. (Most routers these days have this feature.) Dynamic DNS allows you to configure a static host name (such as `mySite.dyndns.org`) that always goes to the IP address of your computer. Because most Internet service providers (ISPs) assign dynamic IP addresses, your IP address will change periodically. A router that supports Dynamic DNS can update a Dynamic DNS service (such as `www.dyndns.com`) with your new IP address whenever it changes. More information on Dynamic DNS is available at www.dyndns.com/services/dns/dyndns/. The service is free of charge and works great with most routers.

SITE MANAGEMENT AND REPORTING

Site Settings

Although Expression Web can certainly be used to create and edit single pages, it really shines when used to create and manage entire sites.

Several configuration settings in Expression Web are designed to affect an entire site. These settings are available in the Site Settings dialog shown in Figure 15.1.

To access the Site Settings dialog, select Site, Site Settings. There are four tabs in the Site Settings dialog, each of which I discuss in the following sections.

General Tab

The General tab displays general information about the site and allows you to change the name of the site and configure the use of metadata.

➔ *For more information on metadata, see Chapter 2, “Creating, Opening, and Importing Sites.”*

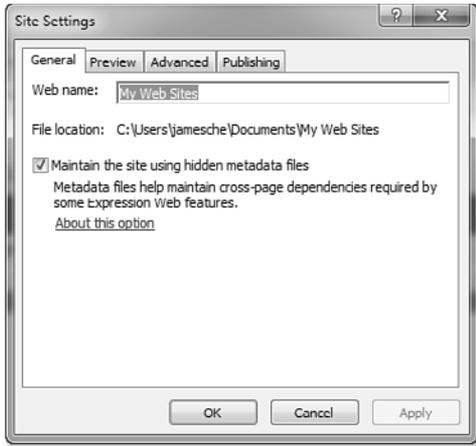
When you change a site’s name in the General tab, the folder name changes as well. If the site is server-based, the URL for the site changes, too.



note

If a remote site open in Expression Web is using the FrontPage Server Extensions, you might experience a delay of several seconds before the Site Settings dialog appears. The FrontPage Server Extensions must read the site’s entire configuration before the dialog is displayed.

This delay is not experienced with sites that aren’t using the FrontPage Server Extensions.

**Figure 15.1**

The Site Settings dialog contains all the settings that apply to an entire site.

If you have a disk-based site open in Expression Web, you'll see a **Maintain the Site Using Hidden Metadata Files** checkbox on the General tab. This checkbox controls whether or not Expression Web uses special files called *metadata files* to store information about your site and its pages. Metadata files are required for some features such as Dynamic Web Templates and Expression Web's ability to update hyperlinks when you move or rename pages.

Preview Tab

The Preview tab configures how pages should be previewed, as shown in Figure 15.2. You can choose to preview using a site's path (including using the Microsoft Expression Development Server for ASP.NET or PHP pages) or a custom URL.

➔ *For more information on using and configuring the Microsoft Expression Development Server, see Chapter 33, "Using the Microsoft Expression Development Server."*

If you are using the Microsoft Expression Development Server to serve PHP pages, you can also use the Preview tab to configure the version of PHP that should be used. You should make sure you are using the same version of PHP here that you will be using on the production server.

➔ *For more information on using PHP with Expression Web, see Chapter 32, "Using PHP."*

Using a custom URL is convenient when you need to use a different URL for previewing than the URL used to open the site in Expression Web. For example, if you are using a disk path in Expression Web for a site located on a web server, you can enter an HTTP path for the custom URL that maps to the disk location used by Expression Web. When pages are previewed in Expression Web, they are opened using the URL you specify.

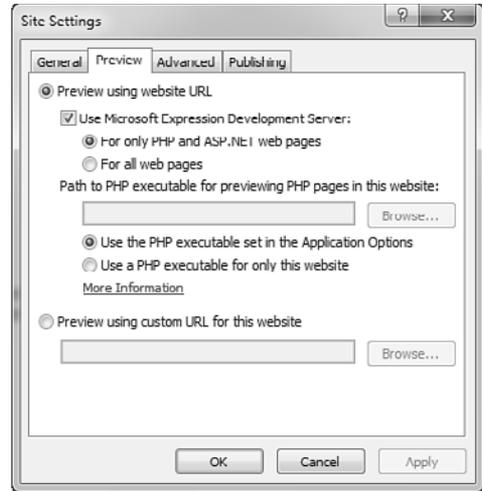


tip

If you are using PHP on a Windows server, use IIS 7 or later if possible because it provides much better PHP performance than other versions of IIS.

Figure 15.2

The Preview tab configures how a site is previewed in your browser.

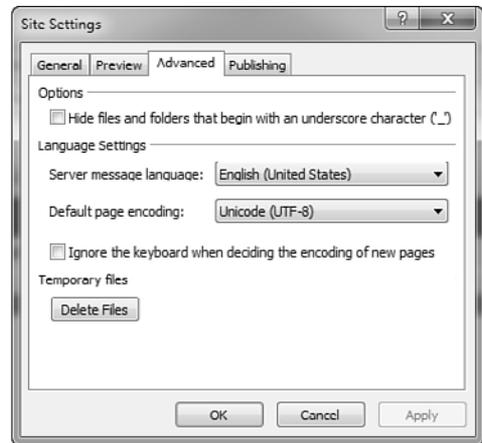


Advanced Tab

The Advanced tab (shown in Figure 15.3) determines whether files and folders that begin with an underscore are displayed in Expression Web, configures the language settings for your site, and allows for the deletion of temporary files.

Figure 15.3

Use the Advanced tab to specify whether files and folders that begin with an underscore are visible, change language settings for your site, and clean up temporary files.



By default, files and folders that begin with an underscore are visible in the Folder List and other Expression Web views. (This does not include metadata in `_vti` folders. They are never visible in Expression Web.) If you want to hide these files and folders (often done to prevent accidental changes to them), check the Hide Files and Folders that Begin with an Underscore Character ('_')

check box and click OK. When you do, Expression Web asks whether you want to refresh the site. Click Yes, and files and folders that begin with an underscore will be hidden in Expression Web.

The setting in the Server Message Language drop-down applies only to FrontPage Server Extensions messages. If the site opened in Expression Web is not a FrontPage Server Extensions-based site, the Server Message Language drop-down will contain only English as an option.

➔ *If you make a change to page encoding and it's not reflected in your pages, see the "Page Encoding Doesn't Affect Pages" Troubleshooting note, p. 263, later in this chapter.*

The Default Page Encoding drop-down configures the character set specified in the content-type META tag added to pages. The browser uses page encoding to determine which character set to use. When the character set is incorrect, the page does not display correctly.

Expression Web uses a default setting of utf-8 (Unicode), which supports all languages and is the correct encoding for XHTML documents.



tip

One of the symptoms of corrupt temporary files is a site that won't open. Many times, this kind of problem results in constant disk activity until the Windows swap file fills up and you run out of disk space. If you run into a problem like this and can't open a site to delete temporary files, you can manually delete the files. They are located in C:\Users\

➔ *For more information on XHTML, see Chapter 12, "Maintaining Compatibility and Accessibility."*

The Delete Files button deletes temporary files that are created for Expression Web to keep track of your site. Although these files don't take up much room on your hard drive, they can become corrupted and cause strange behavior. Deleting them periodically is a good idea for this reason.

Publishing Tab

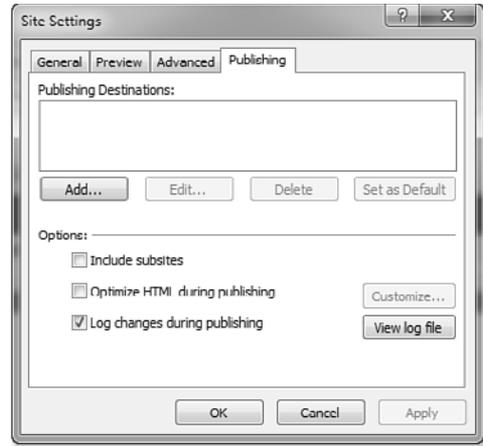
The Publishing tab configures the destinations for publishing and settings used when publishing your site (see Figure 15.4).

The Publishing Destinations section enables you to configure publishing connections that are available in Publishing view. You can add, edit, delete, and set the default publishing destination here.

➔ *For more information on publishing connections in Expression Web, see Chapter 14, "Publishing a Site."*

By default, Expression Web does not publish subsites. If you want to publish subsites along with your site, check the Include Subsites check box.

Figure 15.4
The Publishing tab configures all the settings used when you publish a site in Expression Web.



Expression Web can also optimize your code when publishing; enable this by checking the Optimize HTML During Publishing check box. To customize how Expression Web optimizes HTML, click the Customize button.

➔ *For more information on customizing HTML optimization settings, see Chapter 14, “Publishing a Site.”*

Finally, you can choose to have changes that are made during publishing logged to a log file by checking the Log Changes During Publishing check box. You can then view the log file by clicking the View Log File button.

Site Reports

One of Expression Web’s greatest strengths is in the area of site management. The large array of reports available makes managing a site easier by providing indicators of problems and giving details on other aspects of your site.

The Site Summary view provides a link to some of the reports in Expression Web and also summarizes information about the site. To view the Site Summary (shown in Figure 15.5), select View, Site, Site Summary.

The Site Summary doesn’t list all of the reports available to you. To access all the reports in Expression Web, select the Site Summary drop-down in Site Summary and choose a report from the menu.

The following reports are available in Expression Web:

- **All Files**—Displays all files in the site in a single view. This report also shows the title, location, and other properties of each file.
- **Recently Added Files**—Displays all files recently added to the site. As shown in Figure 15.6, you can select a specific number of days to display files added within that number of days.

Site View

Site Summary

Name	Count	Size	Description
All files	0	0KB	All files in the current site
Pictures	0	0KB	Picture files in the current site (GIF, JPG, BMP, etc.)
Unlinked files	0	0KB	Files in the current site that cannot be reached by starting from your home page
Linked files	0	0KB	Files in the current site that can be reached by starting from your home page
Slow pages	0	0KB	Pages in the current site exceeding an estimated download time of 30 seconds
Older files	0	0KB	Files in the current site that have not been modified in over 77 days
Recently added files	0	0KB	Files in the current site that have been created in the last 30 days
Hyperlinks	0		All hyperlinks in the current site
Unverified hyperlinks	0		Hyperlinks pointing to unconfirmed target files
Broken hyperlinks	0		Hyperlinks pointing to unavailable target files
External hyperlinks	0		Hyperlinks pointing to files outside of the current site
Internal hyperlinks	0		Hyperlinks pointing to other files within the current site
Style Sheet Links	0		All Style Sheet Links in the current site
Dynamic Web Templates	0		All files that are associated with a Dynamic Web Template.
Master Pages	0		All files that are associated with a Master Page.

Folders Publishing Reports Hyperlinks

Figure 15.5

The Site Summary provides a summary of a site and links to run more detailed reports.

Site View

Recently Added Files

30 days

Name	Title	Created Date	Modified By	Modified	Type
SpawnTest.h...	ddd	7/7/2010 5:59 PM	Thinkpad	1 days	html
Default.aspx	Jimco Books - Featuring Ji...	7/7/2010 5:59 PM	Thinkpad	2 days	aspx
sitemap.aspx	Jimco Books Sitemap Hom...	7/7/2010 5:59 PM	Thinkpad	3 days	aspx
aboutus.aspx	Jimco Books About Us	7/7/2010 5:59 PM	Thinkpad	4 days	aspx
master.dwt	Untitled 1	7/7/2010 5:59 PM	Thinkpad	5 days	dwt
contactus.aspx	Jimco Books Contact Form	7/7/2010 5:59 PM	Thinkpad	6 days	aspx
main.css	main.css	7/7/2010 5:59 PM	Thinkpad	7 days	css
email.aspx	Enter your name	7/7/2010 5:59 PM	Thinkpad	10 days	aspx
Default.aspx.cs	Default.aspx.cs	7/7/2010 5:59 PM	Thinkpad	15 days	cs
affiliate.xml	affiliate.xml	7/7/2010 5:59 PM	Thinkpad	20 days	xml
contactus.as...	contactus.aspx.cs	7/7/2010 5:59 PM	Thinkpad	30 days	cs
Web.config	Web.config	7/7/2010 5:59 PM	Thinkpad	60 days	config
seuiew.css	articles/seuiew.css	7/7/2010 5:59 PM	Thinkpad		css
Default.aspx.cs	articles/061111/Default.a...	7/7/2010 5:59 PM	Thinkpad		cs
Default.aspx	Jimco Books From FrontP...	7/7/2010 5:59 PM	Thinkpad		aspx
fig4.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg
fig5.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg
fig1.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg
fig6.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg
fig2.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg
fig7.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg
fig8.jpg	articles/061111/images/fi...	7/7/2010 5:59 PM	Thinkpad		jpg

Report Settings

Folders Publishing Reports Hyperlinks

Figure 15.6

The Recently Added Files report is configurable using the Report Settings drop-down.

- **Recently Changed Files**—Displays all files that have recently changed. Just as with the Recently Added Files report, you can select a specific number of days to display files that have changed within that number of days.
- **Older Files**—Displays all files older than 72 days by default. You can adjust the number of days using the Report Setting drop-down.
- **Dynamic Web Templates**—Shows all Dynamic Web Templates and their file connections.
 - *For more information on Dynamic Web Templates, see Chapter 19, “Using Dynamic Web Templates.”*
- **Master Pages**—Displays all files and any master pages used by each file.
 - *For more information on Master Pages, see Chapter 27, “Using ASP.NET Master Pages and User Controls.”*
- **Style Sheet Links**—Displays all pages and any style sheets linked to them.
 - *For more information on using style sheet links, see Chapter 18, “Managing CSS Styles.”*
- **Unlinked Files**—Shows all files that are not linked to by links that can be followed from the home page. The Links To column shows the number of pages that directly link to a file, and the Links From column shows the number of pages to which the listed file links.
- **Slow Pages**—Displays all pages that take longer than 30 seconds to load by default. The time threshold can be adjusted using the Report Setting drop-down.
- **Hyperlinks**—Displays the Verify Hyperlink report, as shown in Figure 15.7.



caution

Expression Web can get confused with links in scripts and style sheets. Always investigate any files that show up in the Unlinked Files report to ensure that Expression Web isn't showing incorrect data.

Using these comprehensive reports, you can maintain tight control over the files in your site.

Configuring Reports

The reports in Expression Web are generated using some preset defaults. For example, the download speed of a page assumes a 56k connection by default, and the Recent Files report lists files that are fewer than 30 days old by default. You can configure these defaults by selecting Tools, Application Options to access the Application Options dialog shown in Figure 15.8.

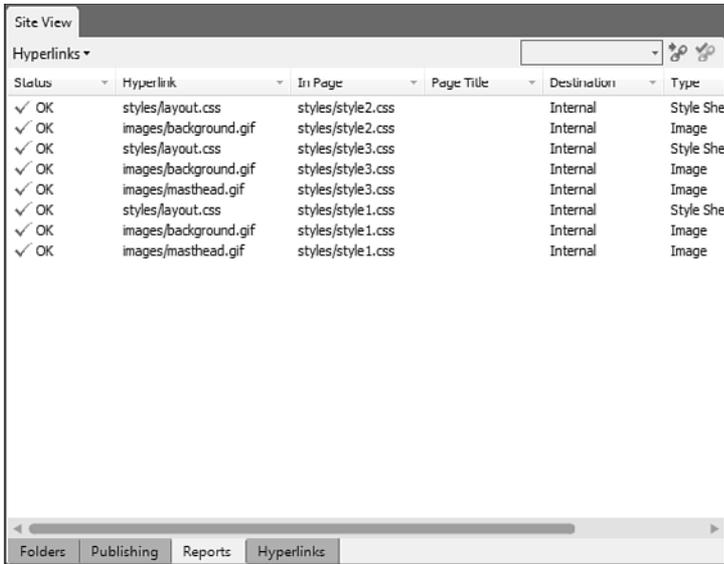


Figure 15.7
Hyperlinks can be verified or edited using the Hyperlinks report.

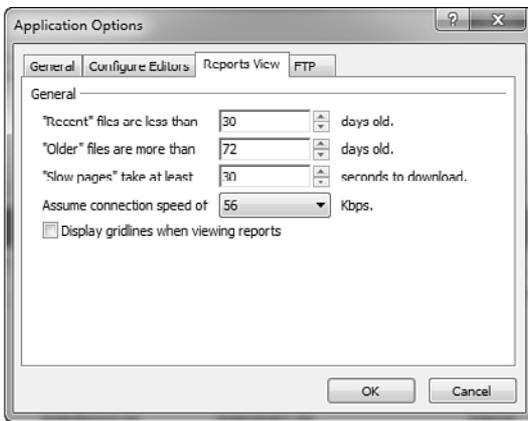
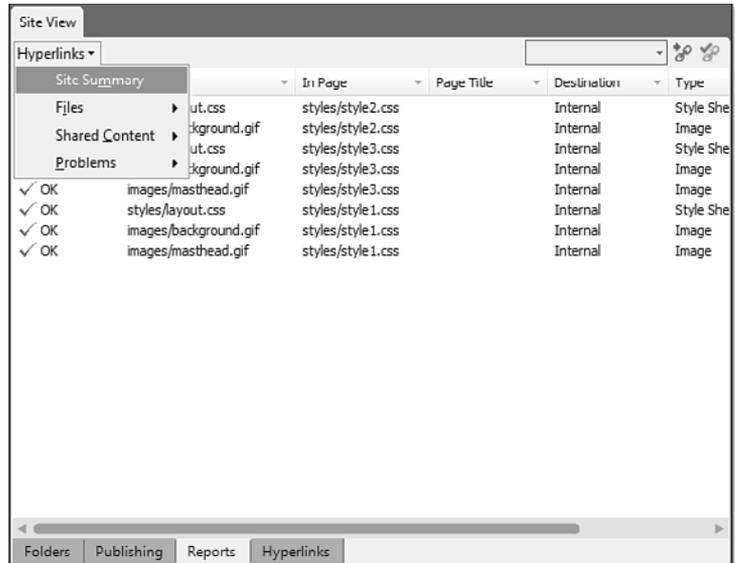


Figure 15.8
The Application Options dialog contains options for configuring the default presets for Expression Web reports.

After viewing a specific report, you can return to the Site Summary view by clicking the name of the current report in the upper-left corner of the main window and selecting Site Summary, as shown in Figure 15.9.

Figure 15.9

The drop-down button in the upper-left of the main screen provides quick access to any report.



Saving Reports

To share an Expression Web report with another designer, you can easily save any report as an HTML file. Simply display the report and then select File, Save to save it as an HTML file.

Expression Web defaults to saving the report in the current site, but you can specify any location.



Page Encoding Doesn't Affect Pages

If you have set the page encoding to something other than the default, but it doesn't seem to affect the content-type META tag in your pages, note that the page encoding setting only affects new pages that are created in Expression Web. It does not affect existing pages.

Using SEO Reports to Increase Traffic

Search engines drive a lot of traffic to today's websites. However, anyone who's ever attempted to locate something on the Internet using a search engine knows that it's often hard to filter good content from all the clutter. As a web designer, you can greatly increase the chances that you'll be discovered by paying attention to search engine optimization, or SEO.

Becoming an expert at SEO isn't easy. Each search engine uses a slightly different approach, and it can be difficult to learn all the subtle differences between search engines. Fortunately, Expression Web makes things easier using the SEO Checker, a new feature in Expression Web 4.

You can access the SEO Checker by selecting Tools, SEO Reports. You can choose to check all pages, open pages, selected pages, or the current page. You can also choose whether you are interested in errors or warnings as shown in Figure 15.10.

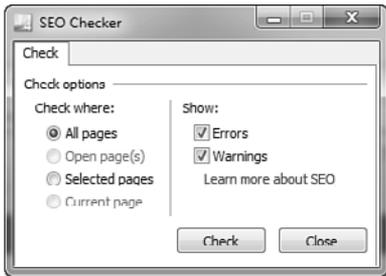


Figure 15.10

The SEO Checker makes it easier to optimize your site for search engines.

To check your site for SEO, click the Check button. Expression Web displays the results in the Search Engine Optimization panel as shown in Figure 15.11.

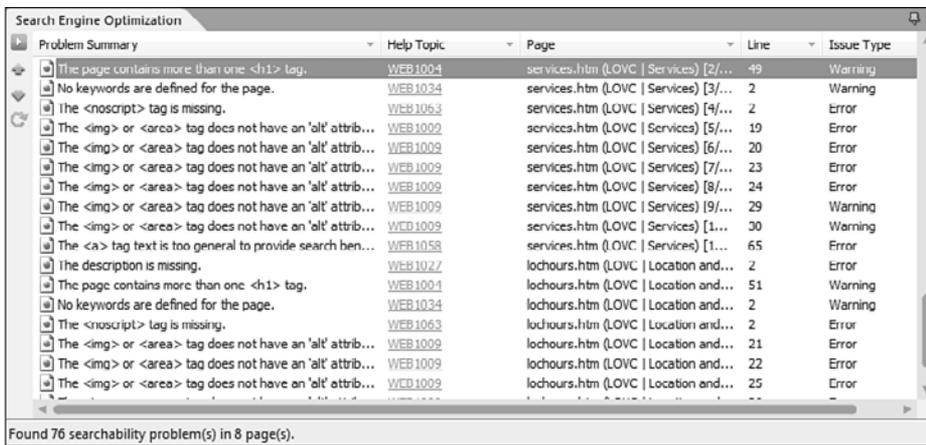


Figure 15.11

The Search Engine Optimization panel provides details of all SEO problems Expression Web finds.

Along with details of each problem found, Expression Web displays a help topic link that shows details of how you can correct the problem. Unfortunately, this information is weighted toward optimization for Microsoft's Bing search engine. For example, Expression Web displays a warning if a page is missing the keywords META tag, and Expression Web documentation says that search engines use this tag to identify important keywords in a



tip

You can get a concise description of a particular problem by right-clicking on it and selecting Problem Details from the menu.

site. However, not all search engines use the keywords META tag, and Google (the most popular search engine by a wide margin) ignores it completely.

To correct a problem identified by the SEO Checker, double-click on the entry in the Search Engine Optimization panel. When you do, Expression Web opens the page in Code View with the problem highlighted so that you can review it and correct it if you want. As a general rule, you should address all of the issues pointed out by the SEO Checker so that you can maximize your site's exposure in search engine results.

By checking your pages for SEO problems and correcting problems that Expression Web identifies, you can ensure that your site is optimized for search engines. However, I recommend that you not stop with Expression Web's report. Google has a comprehensive 22-page document on optimizing your site for its search engine that you can view at http://static.googleusercontent.com/external_content/untrusted_dlcp/www.google.com/en/us/webmasters/docs/search-engine-optimization-starter-guide.pdf. Microsoft has also published a guide for its Bing search engine, and you can access it at <http://download.microsoft.com/download/0/D/9/0D94EECB-C767-445E-B708-9C829275995F/Bing--NewFeaturesForWebmasters.pdf>.

This page intentionally left blank

USING PERSONAL WEB PACKAGES

What Are Web Packages?

Sites can be created with many different structures and formats and for many different purposes. Some formats are applicable to only one particular company or purpose, but many sites consist of one or more reusable parts. For example, a site might have a page that displays data from a database in tabular format, another page that provides a feedback form, another page that lists frequently asked questions, and another page that gives contact information. Such a site could be used by any number of companies or organizations with minimal reconfiguration.

Wouldn't it be nice to have some way of packaging such a site so it could easily be reused or be used by many companies or organizations? Web Packages were designed to do just that!

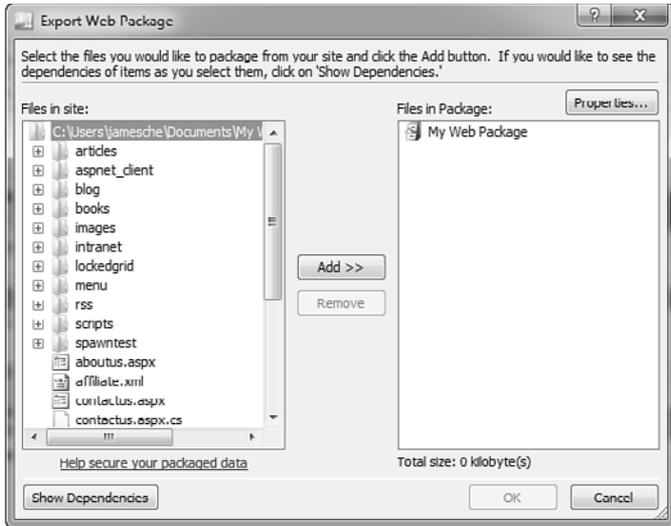
A Web Package enables you to package an entire site or parts of a site into a single file. That file can then be imported into a new or existing site. You could, for example, create a web-based forum comprised of several pages, a database, some graphics, some templates, and some style sheets. All the files that make up the forum could then be exported to a Web Package. That Web Package could then be used to easily add a web-based forum to any site.

Creating a Web Package

To create a Web Package, follow these steps:

1. Open the site that contains the files you want to use in your Web Package.

2. Select Site, Export to Web Package to display the Export Web Package dialog shown in Figure 16.1.

**Figure 16.1**

The Export Web Package dialog lets you easily create a Web Package to use in other sites.

The Export Web Package dialog consists of two panes. At the left is a listing of all files in the current site. At the right is a listing of all files that will be packaged into the new Web Package.

By default, the new Web Package is called My Web Package. If you'd prefer, you can change the title of the Web Package in addition to other properties by clicking the Properties button shown previously in Figure 16.1. When you do, the Web Package Properties dialog is displayed. Figure 16.2 shows the Web Package Properties dialog after entering information about a Web Package.

To add files to your Web Package, select them in the left pane in the Export Web Package dialog, shown previously in Figure 16.1, and click Add. Alternatively, you can double-click a single file to add that file to your Web Package.

When a file is added, Expression Web also adds any dependencies for that file. If you want to examine the dependencies for a particular file, select the file and click the Show Dependencies button. Expression Web displays all the dependencies for the file, as shown in Figure 16.3.

**tip**

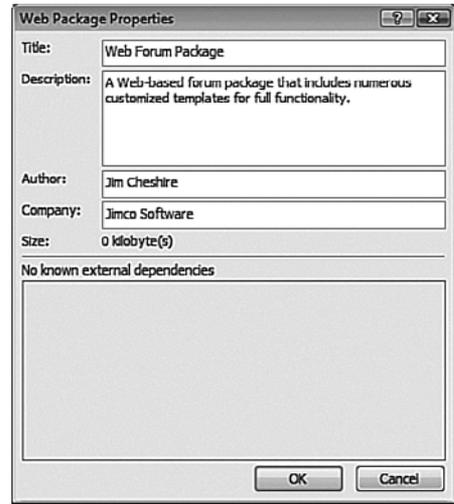
Users who import your Web Package will use the information you enter into the Properties dialog for more information on your Web Package. It's a good idea to always include this information.

**caution**

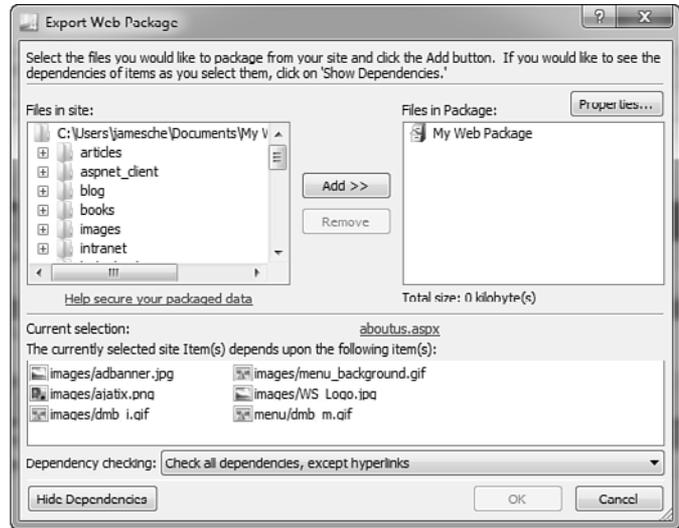
Expression Web uses the dependencies displayed to determine which files should be included automatically with the selected files. Therefore, it's important to carefully choose the correct dependency checking option before adding files to a Web Package.

Figure 16.2

You can further personalize a Web Package by configuring its properties using the Web Package Properties dialog.

**Figure 16.3**

After clicking the Show Dependencies button, you can easily locate files on which the selected files depend. Once you click the button, it changes to a Hide Dependencies button.



Expression Web uses the setting in the Dependency Checking drop-down to determine how it should locate dependencies. The following options are available:

- **Check All Dependencies, Except Hyperlinks**—Locates all files on which the currently selected files depend except those files linked to by the selected files

- **Check All Dependencies**—Locates all files on which the currently selected files depend, including files to which the currently selected files link
- **Do Not Check Dependencies**—Performs no dependency checking



caution

After a Web Package has been saved, it can no longer be edited. To alter the files included in the package, you must export a new Web Package with the new set of files.



File Dependencies Incomplete

Sometimes the dependencies displayed for a file don't include all the files that should be included. If you're concerned that Expression Web will not automatically add files that should be included in your Web Package, don't be alarmed. Expression Web uses metadata to keep track of file dependencies. Occasionally, that metadata can get out of date. To update it and correct file dependencies, open the site and rebuild the metadata.

If you are using a FrontPage Server Extensions site, select Site, Recalculate Hyperlinks to rebuild metadata. If you are using a disk-based site, select Site, Site Settings and clear the Maintain the Site Using Hidden Metadata Files check box and click OK. You then need to select Site, Site Settings again and check the Maintain the Site Using Hidden Metadata Files check box to rebuild the metadata.



caution

If your disk-based site contains subsites, rebuilding the metadata converts the subsites to a regular folder. After you rebuild metadata, you need to right-click on folders that should be a subsite and select Convert to Subsite to convert the folder back to a subsite.

After you've selected all the files to be included in your Web Package, click OK. Expression Web displays the File Save dialog, where you can select where to save your package.

Importing a Web Package

To import a Web Package, open a site or create a new one. Select Site, Import, Import from Web Package to display the File Open dialog. Browse to the Web Package file and click Open to import it. When you click Open, the Import Web Package dialog is displayed, as shown in Figure 16.4.

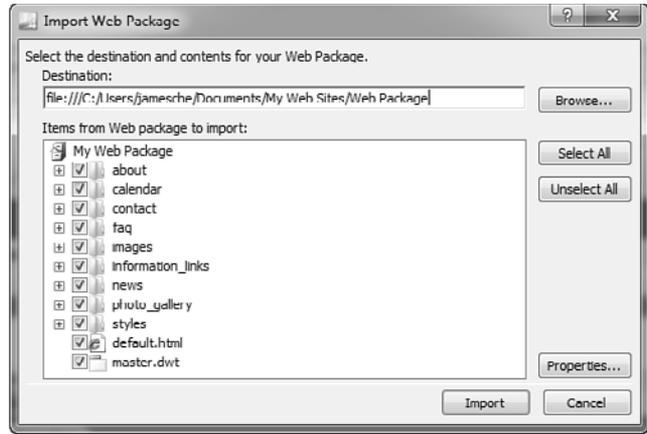


note

A sample Web Package appears in the Examples\Ch16\Files folder on this book's website at informat.com/register. You can import this sample into one of your sites to examine how a Web Package works.

Figure 16.4

The Import Web Package dialog displays all the files inside a Web Package. You can either import some or all of the files.



A Web Package can be imported into any existing site. By default, Expression Web creates a new folder in the site for the files in the Web Package. The folder name matches the name of the Web Package by default, but you can use a different folder name by specifying it in the Destination text box.

If you want more information on a Web Package prior to importing it, click the Properties button shown previously in Figure 16.4. Expression Web displays a dialog showing the author, company name, and a description of the Web Package to aid in determining whether you want to import the Web Package.

**tip**

If you want to import a Web Package into a new site that contains only the files in the Web Package, create a new Empty Web Site and import the Web Package into it.

**tip**

Keep in mind that Expression Web does not conduct any dependency checking when importing a Web Package, so if you choose to not import all files, carefully test the imported files and import any additional files if necessary.

To import a Web Package, select the files you want to import and click Import. When you do, Expression Web might display a security warning, as shown in Figure 16.5. If you don't know or trust the source of the Web Package, you can select Don't Run to cancel importing the Web Package.

If you click the Properties button, you can configure the properties of your Web Package. This is useful in cases where you might be distributing the Web Package to someone else because it allows them to see specific information about the Web Package while they are importing it.

**caution**

When you import the sample Web Package on this book's website at informat.com/register, you see the warning shown in Figure 16.5. In this case, it's perfectly safe to import the Web Package.



Figure 16.5 Web Packages that are not digitally signed display a warning when you try to import them. Import a Web Package only if you know and trust the source.

Capitalizing on Web Packages

Earlier in this chapter, I gave an example of creating a Web Package of a web-based forum. Creating a web-based forum is not a simple task and requires quite a bit of programming knowledge. However, the example should not be dismissed because it demonstrates the enormous opportunity that Web Packages provide to an innovative designer.

I get a lot of email from people looking for solutions to web design problems. The problems I see are almost never related to HTML code or how to design a particular page element. Instead, most people ask about adding large-scale functionality to a site, such as a feedback form that submits into a database and sends email.

If you're the kind of person who likes to write web applications or create innovative designs, Web Packages represent an enormous opportunity. You could easily open an e-commerce storefront offering Web Packages that provide snap-in capability to a site. Such an endeavor would be highly lucrative.

As you develop your sites, keep in mind which parts might be useful to others. Package those parts into Web Packages, and use the power of the Internet to market them to other designers. If you're going to pursue this route, however, you should get a digital certificate and sign your Web Packages so they don't generate the security dialog shown previously in Figure 16.5.

To get a digital certificate, you'll need to work with a certificate authority such as Verisign, Inc., and obtain a Class 2 digital certificate. Purchasing one of these certificates is fairly expensive, but it's a necessity if you want to enter into the realm of selling Web Packages.

Verisign, Inc., has a lot of information (including comprehensive guides) on its site. You can read more by visiting www.verisign.com/products-services/security-services/code-signing/digital-ids-code-signing/index.html.

CREATING STYLE SHEETS

An Introduction to CSS

In the most basic sense, HTML provides a description of a page. It tells a browser things such as “this text should be green,” “there should be a graphic here,” or “this text should link to a particular location.” In other words, HTML is primitive. It’s so primitive, as a matter of fact, that even the earliest web designers realized its limitations quickly. They yearned for a way to define the layout of a page.

In the early days of HTML, the models for disseminating information in written form were newspapers, magazines, and scientific journals. If a web designer wanted to try to present information in the same manner in a page, he was just out of luck. HTML did not provide the means to precisely control layout. As browsers developed, web designers found themselves even more frustrated as the few features that did allow for some layout techniques were removed. There had to be a better way.

In 1994, CSS was unveiled to the web design community. I can remember the advent of CSS, and I don’t remember it being a big deal. However, as time has progressed and as CSS has matured, it has gained in importance. In today’s world of web design, you’re severely impacted (and not in a positive way) if you aren’t well-versed in how to use CSS effectively.

Microsoft has traditionally fallen behind in support for CSS. FrontPage, Microsoft’s web development platform prior to Expression Web, suffered from poor CSS support. FrontPage’s weakness was highlighted to a greater extent due to the fact that Macromedia Dreamweaver (now Adobe Dreamweaver)—FrontPage’s primary competitor—had excellent support for CSS. Something had to change if Microsoft was going to stay in the web design game.

That something was Expression Web. The support for CSS in Expression Web was vastly improved over any of Microsoft's previous web development tools, and that tradition continues in the latest release of Expression Web. In the next couple of chapters, we explore the CSS capabilities of Expression Web. When we're finished, you'll have all the skills needed to confidently manage all your CSS needs in Expression Web.

The Purpose of CSS

CSS is well-suited to applying formatting to page elements. For example, you can change font color, size, and style using CSS. You also can change the position of text, tables, images, and so on. However, CSS isn't alone in its capability to format page elements. HTML has tags and attributes that provide similar functionality. The HTML `` tag, for example, has a `color` attribute, a `size` attribute, a `face` attribute, and the like, and all of these can be used to apply formatting to text. Given that fact, why even worry about CSS?

To gain a full understanding of why you would want to use CSS instead of traditional HTML formatting techniques, let's look at an example. The code in Listing 17.1 shows a simple page using only HTML to apply formatting.

Listing 17.1 A Simple Page

```
1 <html>
2 <head>
3   <title>Welcome</title>
4 </head>
5 <body>
6 <p>
7   <font face="Arial" size="4" color="black">
8     <b>Welcome to our site!</b>
9   </font>
10 </p>
11 <p>
12   <font face="Arial" size="2" color="navy">
13     We have plenty to offer. Browse around and
14     <a href="mailto:info@ourwebsite.com">let us know</a>
15     what you think.
16   </font>
17 </p>
18 <p>
19   <font face="Arial" size="4" color="black">
20     <b>Company News</b>
21   </font>
22 </p>
23 <p>
24   <font face="Arial" size="2" color="navy">
```



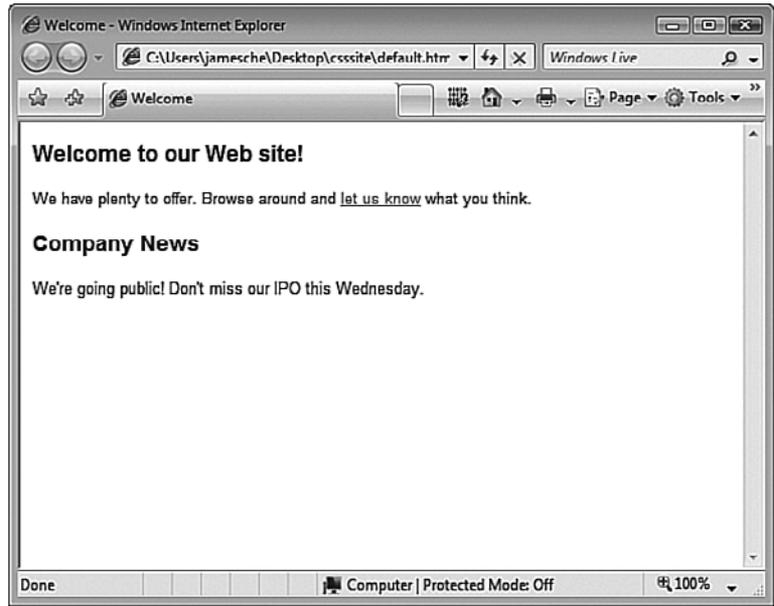
Formatting page elements using HTML is quickly being replaced by CSS. In fact, the most recent HTML recommendation from the W3C suggests that most HTML formatting techniques be avoided in favor of CSS.

```
25         We're going public! Don't miss our IPO this Wednesday.  
26     </font>  
27 </p>  
28 </body>  
29 </html>
```

Figure 17.1 shows the page from Listing 17.1 in a browser window.

Figure 17.1

A simple page formatted using only HTML tags and attributes.



As you can see, getting the formatting you want using simple HTML is easy. Now let's imagine I've used this same HTML formatting technique throughout an entire site. What happens if I need to change the font from Arial to Verdana? Worse yet, what happens if I need to change the font to Verdana and I also need to change the color and size of the headings?

The only way to take care of such tasks across a large site would be to use search and replace to change the code. But if I use search and replace, I have to allow my web design tool to make widespread changes across my site in bulk. What if my site includes some articles on CSS and I have included the word *Arial* in those articles? Unless I account for that in my search and replace, I could end up messing up the actual content of my site.



caution

Making site-wide changes using any type of search and replace tool is risky. Computers are notoriously bad at recognizing the context in which words are being used, and if you rely on a computer program to determine what content to change in your site, you are opening yourself up to errors that can go undetected until a site visitor points them out.

CSS solves this problem perfectly because it allows you to separate your content from your formatting. By applying formatting with CSS, you can simply make changes to a CSS file and it will apply to all relevant formatting across the entire site.

Listing 17.2 shows the page from Listing 17.1 with formatting applied using CSS instead of HTML code.

Listing 17.2 Page Formatting with CSS

```
1 <html>
2 <head>
3     <title>Welcome</title>
4     <link rel="stylesheet" type="text/css" href="styles/styles.css">
5 </head>
6 <body>
7 <p class="pageHeading">
8     Welcome to our site!
9 </p>
10 <p class="stdContent">
11     We have plenty to offer. Browse around and
12     <a href="mailto:info@ourwebsite.com">let us know</a>
13     what you think.
14 </p>
15 <p class="pageHeading">
16     Company News
17 </p>
18 <p class="stdContent">
19     We're going public! Don't miss our IPO this Wednesday.
20 </p>
21 </body>
22 </html>
```

This page looks much cleaner. It also has seven fewer lines of code than the page in Listing 17.1. That means less code for users to download, and that translates directly into pages that download faster.

In Listing 17.2, I used CSS classes to apply formatting. Line 4 links a CSS file to the page. Inside that CSS file, I created two CSS classes, one called `pageHeading` and another called `stdContent`. Those CSS classes are applied to page elements using the HTML `class` attribute.

As we progress through this chapter, I'll demonstrate how easy it is to use the features provided in Expression Web to format pages using CSS; first, though, let's briefly discuss how CSS can be applied to a page.



tip

The small example here actually decreased the size of this page by one-third. On actual pages, using CSS for formatting can reduce the size of pages dramatically. Site visitors will have to download the CSS file the page links to, but the CSS file gets cached on the user's end, so it has to be downloaded only once.



note

CSS classes are only one way of applying CSS to a page. We discuss all the methods of applying styles in the next couple of chapters.

How CSS Is Applied to Pages

The three ways to apply CSS to your pages are as follows:

- **External style sheets**—External style sheets are files with a `.css` file extension in which styles are defined for one or more pages.
- **Embedded style sheets**—Embedded style sheets use the same syntax as external style sheets, but the CSS code exists within the HTML instead of within a separate file.
- **Inline styles**—Inline styles are applied using the `style` attribute of an HTML tag.

It is uncommon to use only one of these techniques. Most web designers use a combination of two or more techniques to define the styles for a particular site. However, there are benefits and drawbacks to each technique, and understanding those benefits and drawbacks is critical to using CSS correctly.

External Style Sheets

When most people think of CSS, they think of external style sheets. An *external style sheet* is a special file with a `.css` file extension. Inside this file, the styles for one or more pages are defined. The page in Listing 17.2 is formatted using an external style sheet.

Styles in an external style sheet can be defined using existing HTML tags, by creating custom style definitions known as CSS classes, or by using a combination of the two.

External style sheets are connected to pages using the HTML `<link>` tag. The following code creates a link to a style sheet called `styles.css` located in a `styles` directory:

```
<link rel="stylesheet" type="text/css" href="styles/styles.css" />
```

In the preceding code snippet, I linked to a style sheet located in a folder named `styles`. In fact, a style sheet can be stored in any folder you choose, and web designers typically create a specific folder structure for organizing style sheets.

Multiple external style sheets can be attached to a page. If multiple style sheets are attached, styles are processed in the order in which they appear on the page. This process is known as *casca-*
ding.

Embedded Style Sheets

Embedded style sheets look just like external style sheets, but instead of existing inside a separate file that is linked to a page, an embedded style sheet is actually part of the HTML code of the page.



note

You'll learn more about cascading in the next chapter.



note

Embedded style sheets apply only to the page that contains them. Only external style sheets can be applied to multiple pages.

There are several reasons for using an embedded style sheet. Because an embedded style sheet overrides any of the styles configured in an external style sheet, embedded style sheets are often used to create page-specific styles while still using global styles that exist in an external style sheet.

It's also common to use embedded style sheets in scenarios where there is no semblance of a site structure. For example, if you distribute software and you include an HTML file in the root of your software's CD, you might use an embedded style sheet to define the CSS styles for the HTML file. You might also use an embedded style sheet if you send HTML email for a mailing list. I have several mailing lists to which I send mailings on a regular basis, and I use embedded style sheets to define the layout of those mailings.

Inline Styles

Inline styles are applied directly to an HTML tag via the `style` attribute. The following code shows an inline style applied to a `<p>` tag:

```
<p style="font-family:Arial; color: gray">Some Text</p>
```

➔ *Expression Web's Find and Replace feature is an excellent tool for locating inline styles in your site. For more information on locating HTML attributes using the Find and Replace tool, see "Using HTML Rules in Find and Replace," p. 179.*

The use of inline styles is perfectly fine if it's a deliberate decision. However, if you use inline styles indiscriminately, it can cause many problems because inline styles override a style defined anywhere else. In other words, if you have a page that is full of inline styles and you decide to move to external or embedded style sheets after those inline styles have been added, you'll need to remove the inline styles from the page. If you don't, styles that are defined in your external and embedded style sheets will be overridden by the inline styles.

Expression Web typically uses embedded style sheets instead of inline styles, and while embedded styles still override external styles, they are easier to locate and modify or remove.

We cover the details of inline styles in the next chapter.



caution

Expression Web sometimes applies formatting using inline styles. When making formatting changes to dialogs in Expression Web, it's always a good idea to check the code to ensure you haven't overridden a style or otherwise changed formatting in a way you didn't intend.

Formatting Content with CSS

In case you haven't picked up on it yet, I'm a big believer in learning by doing. In keeping with that philosophy, we're going to learn how to create and use CSS in Expression Web by creating a simple site and formatting it using CSS. Here's how:

1. Open an existing site or create a new one-page site in Expression Web.
2. Open the page that Expression Web creates in the site.
3. Enter the text **Welcome to Our Site**.

4. Select Heading 1 from the Style drop-down.
5. Press Enter.
6. Enter a few sentences of text. Anything you choose is fine.
7. Press Enter to start a new paragraph.
8. Enter the text **Links**.
9. Select Heading 2 from the Style drop-down.
10. Enter some hyperlinks, one per line, to some sites of your choice.
11. Save the page.

➡ *For more information on creating a site, see Chapter 2, “Creating, Opening, and Importing Sites.”*

➡ *For more information on formatting text, see Chapter 3, “Creating Pages and Basic Page Editing.”*

When you select Heading 1 from the Style drop-down, Expression Web wraps the paragraph with the <h1> tag. Selecting Heading 2 wraps the paragraph with the <h2> tag. By default, the browser determines what heading styles look like, but you can easily customize them using CSS.

Select File, New, CSS to create a new, blank style sheet. Add the code in Listing 17.3 to the empty style sheet.

**tip**

To generate some content for sample pages such as the one you just created, use the Lipsum Generator located at www.lipsum.com/.

Listing 17.3 The Style Sheet

```
1  h1, h2
2  {
3      font-family: Arial;
4      font-size: 14px;
5      color: Black;
6  }
7
8  p
9  {
10     font-family: Arial;
11     font-size: 9px;
12     color: Gray;
13 }
```

Save the style sheet into the current site as `styles.css`.

CSS files have a fairly simple syntax. In this CSS file are two blocks of CSS code. The first block consists of lines 1–6. The second block consists of lines 8–13. Each block begins with an HTML selector. A selector is really just an HTML tag you are formatting with CSS.

This code snippet shows the syntax used in a CSS file:

```
selector,[selector]...
{
    <CSS formatting code for the selector(s)>
}
```

At least one selector must be specified. However, you can specify more than one if you want by separating them with commas.

Following the selector(s) is an opening curly brace followed by CSS formatting code. A closing curly brace closes off the code for the specified selector(s).

CSS code itself is in the following format:

```
css_property: value;
```

In Listing 17.3, the first block specifies the `h1` and `h2` selectors. Notice that both of these selectors appear on the same line and are separated by a comma. In this case, I am formatting both the `<h1>` and `<h2>` HTML tags with the CSS code in lines 3–5. In line 8, I have specified the `p` HTML selector, and that selector is formatted using the CSS code in lines 10–12.

Now that we've created a CSS style sheet, we need to link it to our page so that the styles defined in the style sheet apply to the page. Switch back to the page you created earlier and do the following:

1. Select Format, CSS Styles, Apply Styles to activate the Apply Styles panel, as shown in Figure 17.2.



note

CSS has numerous properties; we won't cover all of them in this book. For a detailed look at CSS, read *CSS: The Definitive Guide* from O'Reilly Publishing.

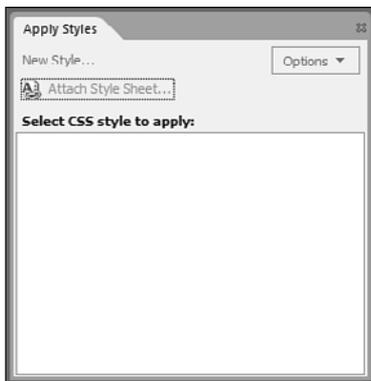


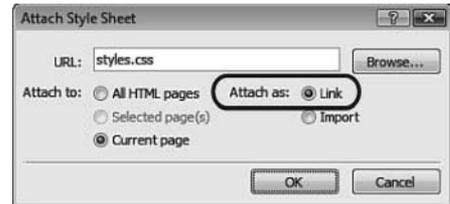
Figure 17.2

The Apply Styles panel is a powerful tool for applying CSS styles.

2. Click the Attach Style Sheet link in the Apply Styles panel.
3. Click the Browse button in the Attach Style Sheet dialog, as shown in Figure 17.3.

Figure 17.3

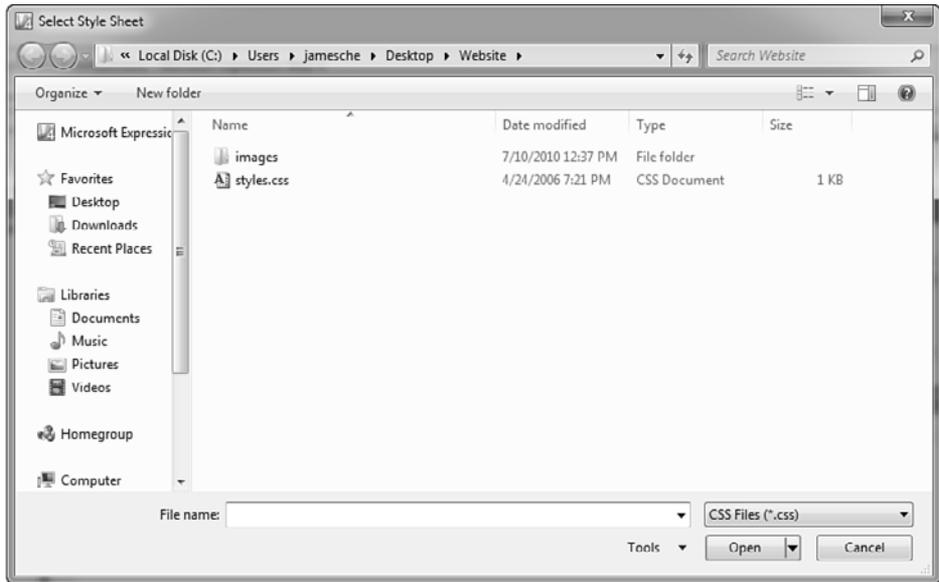
The Attach Style Sheet dialog allows you to link an external style sheet to one or all pages. You also can import a style sheet into a page as an embedded style sheet.



4. In the Select Style Sheet dialog, navigate to the `styles.css` file as shown in Figure 17.4.

Figure 17.4

The Select Style Sheet dialog provides a view of files on your computer as well as on file shares and web servers.



5. Select the `styles.css` file and click Open.
6. Make sure that the Link radio button is selected in the Attach Style Sheet dialog, as shown in Figure 17.3.
7. Click OK.



Attach Style Sheet Link Is Disabled

To attach a style sheet using the Apply Styles panel, you must be in Design View. If you have Code View activated, you won't be able to click the link. Switch to Design View, and you should be able to attach the style sheet without any problems.

After you click OK in the Attach Style Sheet dialog, you should see the formatting of the page change immediately. Figure 17.5 shows the original page without CSS formatting, and Figure 17.6 shows the page after applying the CSS style sheet.



Figure 17.5
Our simple page without CSS formatting applied.

After applying the style sheet, you might notice that the paragraph text on the page is probably a bit too small. Let's edit the CSS file to increase the size of the text slightly:

1. Open `styles.css` inside Expression Web. Leave `default.htm` open as well so you can quickly see the effect of your changes to the style sheet.
2. Change the CSS code for the `p` selector from this:

```
font-size: 9px  
to this:  
font-size: 11px
```

Figure 17.6

Linking the CSS style sheet to the page in Figure 17.5 has changed the formatting. No other changes were made to this page.



3. Switch to `default.htm` and review the change.

4. Save the changes to `styles.css`.



Prompted to Save Embedded File

If your style sheet contains unsaved changes and you save a page that is linked to the style sheet, Expression Web prompts you to save the style sheet as well using the Save Embedded File dialog. Simply click OK in this dialog to save both pages.

You can avoid this dialog by always explicitly saving your style sheet after making changes to it.

Notice that the changes made to the style sheet immediately changed the text size in `default.htm`, but those changes in formatting didn't actually cause a change to the `default.htm` file itself. In other words, you don't have to save `default.htm` after making a change to the style sheet. In fact, this is a good illustration of the power of CSS. If you had 1,000 pages linked to `styles.css`,

a change to `styles.css` would immediately affect all those pages without directly making any change to the pages themselves. You should now be starting to really grasp the importance of using CSS to develop sites efficiently.

Positioning Content with CSS

So far, we've looked at changing the formatting of text elements using CSS. However, CSS also has powerful positioning capabilities.

You've probably heard of *absolute positioning*. Absolute positioning is the technique of positioning elements on a page using an x,y coordinate system. Using this technique, the upper-left pixel on the page is at location 0,0. As you progress across the page horizontally, the x-coordinates increase; as you progress down the page vertically, the y-coordinates increase. Using this method, you can precisely plot a point on a page.

Absolute positioning is accomplished using CSS exclusively. In fact, one of the drawbacks to using HTML is that it has no concept of placement on a page other than basic alignment. For example, you can choose to align a paragraph on the right edge, left edge, or center of a page, but if you want more precise alignment, you need to use tables or insert a resizable page element adjacent to the object you're attempting to align. This approach is clumsy at best and a source of extreme frustration for web developers.

One of the reasons absolute positioning is a great solution is that it uses the pixel as a system of measurement. Regardless of screen resolution, computer type, or browser flavor, a pixel is always going to be a universal measurement. If you position a graphic 200 pixels from the left edge of a page, that graphic will be exactly 200 pixels in distance from the edge on every computer that displays your page.

Let's position a graphic on `default.htm` using CSS positioning.

1. Create a new folder in the current site. Name the folder `images`.
2. Import a graphic file of your choice into the `images` folder. You can use the sample graphic in the `Examples\Ch17\Files` folder at informit.com/register.
3. Place the cursor at the left of the first line on `default.htm` and press `Enter` to add a new line at the top of the page.
4. Add the image you imported in step 2.
5. Click the image to select it. Click the down arrow on the `` Quick Tag Selector and select `Positioning, position: absolute`, as shown in Figure 17.7.



tip

Absolute positioning might not be the Holy Grail that it first appears to be. There are some weird behaviors that you might notice if you use absolute positioning often. A good resource to keep track of weird behavior regarding CSS positioning is the *Position Is Everything* site located at www.positioniseverything.net.



caution

Absolute positioning should only be used when you want a page element to be positioned at a precise position regardless of the flow of other content. In some cases, positioning items absolutely can cause the layout of your page to change drastically as the browser resizes.

Figure 17.7

The Quick Tag Selector is a great way to change the positioning of an image quickly and easily.



- For more information on adding graphics to a page, see Chapter 9, “Using Graphics and Multimedia.”
- For more information on using the Quick Tag Tools, see Chapter 8, “Using the Quick Tag Tools.”
- For more information on positioning objects, see Chapter 3, “Creating Pages and Basic Page Editing.”

Even though you made no modifications to the `styles.css` file, the image you just moved actually uses inline CSS for positioning. A quick view of the code for the image is all that’s needed to confirm that point. Here is the `` tag after being repositioned:

```

```

- For more information on inline styles, see Chapter 18, “Managing CSS Styles.”

**Figure 17.8**

When an image has been set to be absolutely positioned, it actually appears on top of other things on the page.

Your exact tag will differ slightly depending on where you positioned the image, but you'll still see the same basic syntax. The `style` attribute added to the `` tag now contains the following settings:

- `position: absolute`—Specifies that the image should be positioned absolutely on the page.
- `top: 259px`—Positions the image 259 pixels from the top of the page. Your value will likely be different.
- `left: 232px`—Positions the image 232 pixels from the left edge of the page. Your value will likely be different.
- `z-index: 1`—Sets the z-position for the image. Z-position 0 is the surface of the page. Higher z-positions appear above content with lower z-positions.

➡ *For more information on z-position, see Chapter 23, “Using Layers.”*

If you remember my previous comments on using inline styles, you might begin to question why Expression Web chooses an inline style for the positioning of this image. In many cases, the position of absolutely positioned page elements is page-specific. Therefore, positioning those elements using inline styles is the most efficient method.

In some cases objects can be absolutely positioned on all pages. For example, you might have a menu that appears on all your pages. If you were going to position that menu absolutely, you'd

likely want to control its position using an external style sheet so you could easily reposition it on all pages by simply changing a single CSS file. A CSS class is the perfect solution for such a task.

CSS Classes

So far, we've looked at using HTML selectors (`h1`, `h2`, and `p`) inside a style sheet. These work fine when you want to control the formatting or placement of all instances of those particular tags, but they don't work so well for positioning tasks, such as the earlier menu example.

Take our image, for example. If we want to change the method of positioning and use an external style sheet instead of an inline style, we could add the `img` selector to `styles.css` and specify positioning for that selector. However, if we do that, we are going to alter the positioning of *all* images on all the pages that link to `styles.css`. That's obviously not what we want to do. Wouldn't it be nice if we could define our own explicit way of referring to our logo image inside the CSS file? We could then define some CSS code that would apply only to the logo image and not other images. That's precisely what CSS classes were designed to do.

Basic Application of a CSS Class

The syntax for a CSS class is almost identical to what you've already seen. The difference is that CSS classes are preceded by a dot. The following CSS code defines a new CSS class called `logo`:

```
.logo
{
    CSS Code
}
```

To apply the `logo` CSS class to an element on our page, the HTML class attribute is used. The following `` tag uses the CSS properties assigned to the `logo` CSS class:

```

```

Let's change `default.htm` so that the logo on it is positioned using a CSS style.

1. Add the following code to `styles.css`:

```
.logo
{
    position: absolute;
    top: 259px;
    left: 232px;
    z-index: 1;
}
```

2. Save `styles.css`.



caution

CSS is case-sensitive. Therefore, a class called "logo" is not the same as a class called "Logo".



tip

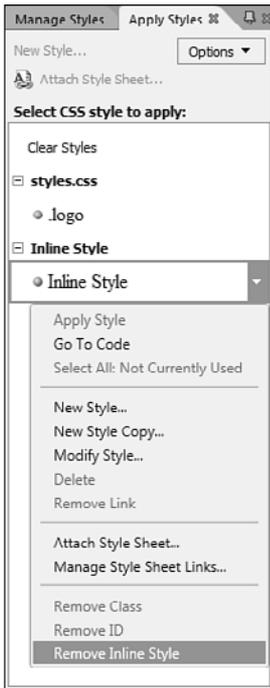
When the CSS class is defined in the HTML tag, there is no dot in front of the name. The dot appears only in the style sheet.



note

I'm using absolute positioning in this example because it isn't impacted by other page elements. However, it's a good idea to brush up on all your options for CSS positioning. You can read more on CSS positioning at http://www.w3schools.com/css/css_positioning.asp.

3. Switch to `default.htm`.
4. Select the image.
5. In the Apply Styles panel, click the down arrow next to **Inline Style** and select **Remove Inline Style**, as shown in Figure 17.9. This removes the inline style that was applied when we dragged the image into position earlier.

**Figure 17.9**

Removing an inline style is as simple as clicking a menu item thanks to the Apply Styles panel. You'll learn about many more capabilities of this tool in the next chapter.

6. Make sure the image is selected, and then select the `.logo` class from the Apply Styles panel, as shown in Figure 17.10.

As soon as you click the `.logo` class in the Apply Styles panel, the image should jump to the position specified in the style sheet.

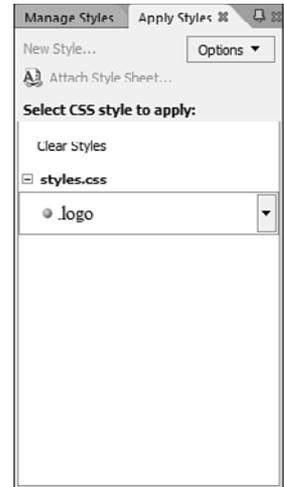


CSS Class Not Visible

If you've created a CSS class in your style sheet but it's not visible when you try to apply that class using the Apply Styles panel, make sure the style sheet in which the class was created is linked to the page you are editing. If the style sheet is not linked to the page, the classes within the style sheet will not be available.

Figure 17.10

There are many ways to apply CSS classes. The Apply Styles panel is probably the easiest way.

**tip**

When we created the `logo` CSS class, an HTML selector was not specified. Therefore, the `logo` CSS class applies to any HTML tag with a `class` attribute value of `logo`. Technically, the CSS definition for the `logo` class as we defined it is `*.logo`, meaning that it applies to all tags.

If we had wanted to limit the `logo` class so that it only applied to the `img` selector, we could have defined it as `img.logo`.

➡ For more details on using the Apply Styles panel, see Chapter 18, “Managing CSS Styles.”

Applying Multiple CSS Classes

The power of CSS classes is only partially illustrated by our simple example. You can actually define several CSS classes in a style sheet and apply all of them to an HTML selector by separating each class name with a space. The following HTML code applies the `logo` and `headImg` classes to an image:

```

```

The order of CSS classes is important. Classes are applied in the order in which they appear, so in the preceding example, settings defined in the `headImg` class override the settings defined in the `logo` class.

This technique can be taken a step further by specifying multiple classes in the CSS file itself. Consider the following CSS code:

```
div.newsHead.monthHead
{
    font-face: Arial;
}
```

In this case, the defined style is only applied to HTML `<div>` tags with a `class` attribute containing both the `newsHead` and the `monthHead` CSS classes. Therefore, the CSS code would apply to the HTML tag:

```
<div class="newsHead blogContent monthHead">
```

but would not apply to the HTML tag:

```
<div class="newsHead blogContent">
```

Pseudo-Classes

In addition to the capabilities available using selectors and classes, CSS also includes a number of pseudo-elements and pseudo-classes that allow you to apply complex formatting easily.

Link and dynamic pseudo-classes are the most common pseudo-classes by far. If you've used CSS in your pages before, you've probably used these pseudo-classes without realizing it.

Link pseudo-classes allow you to define a different appearance for HTML hyperlinks based on whether a link has been visited. The `:link` pseudo-class formats an unvisited link, and the `:visited` pseudo-class formats a visited link.

The following CSS code formats links in blue and visited links in red:

```
a:link { color: blue; }
a:visited { color: red; }
```

Dynamic pseudo-classes take link pseudo-elements one step further by allowing you to format hyperlinks based on the current state of a link. The `:hover` pseudo-class formats a hyperlink over which the mouse is hovering; the `:focus` pseudo-class formats a hyperlink that has the focus; and the `:active` pseudo-class formats a hyperlink that is currently being activated.

The following CSS code builds on the previous code by specifying that links have an overline and underline when hovered over, a bolded font when they have the focus, and are italicized when the links are active:

```
a:hover { text-decoration: underline overline; }
a:focus { font-weight: bold; }
a:active { font-style: italic; }
```



note

The syntax that is most common includes the `a` selector, but the CSS specifications do not require it.



:hover Pseudo-Class Doesn't Work

If you've used the `:hover` pseudo-class to format your hyperlinks, and Expression Web doesn't indicate that there's any problem with the CSS code but the effect doesn't work, you have probably added the `:hover` pseudo-class in the wrong order inside your CSS code. The `:hover` pseudo-class should appear after the `:link` and `:visited` pseudo-classes. Otherwise, the formatting specified in the `:hover` pseudo-element will be overridden.

Pseudo-Elements

Pseudo-elements allow web designers to apply formatting that would otherwise be impossible to obtain. The current CSS specification defines two pseudo-elements: the `:first-line` pseudo-element and the `:first-letter` pseudo-element.

The `:first-line` pseudo-element allows you to style the first line of a paragraph. The following code makes all characters in the first line of a paragraph uppercase:

```
p:first-line { text-transform: uppercase; }
```

This pseudo-element is incredibly powerful because applying this formatting using any other method would be impossible in most cases. The content that appears on the first line of a paragraph is often dictated by the width of the browser window. As the browser window is resized, the word-wrap feature of the browser causes the specific words that appear on the first line to change. The `:first-line` pseudo-element deals with that perfectly and automatically styles only the characters that appear on the first line.

The `:first-letter` pseudo-element allows for styling the first character in a paragraph. This pseudo-element is often used for drop caps or other effects that are often seen in printed media. The following CSS formats the first character in a paragraph to large, blue type:

```
p:first-letter { font-family: Times New Roman; font-size: 14px; color: blue; }
```

It might sound like rhetoric, but the power of CSS cannot be overstated in today's world of web design. As new standards are developed, CSS continues to surge in importance. This chapter gave you a small taste of what is possible using CSS. As you progress through the next chapter, you'll discover just how easy it is to take advantage of CSS using Expression Web.



caution

Keep in mind that pseudo-elements might not render in all browsers. Test carefully in multiple browsers when using pseudo-elements.



note

Pseudo-elements are not rendered in the Design View of Expression Web. To see the formatting of pseudo-elements, you need to preview your page in a browser. Some pseudo-element styles (such as those applied to paragraphs) are also not displayed in the Apply Styles panel.



note

For additional information on in-depth CSS topics, visit the CSS 2.1 specification on the W3C site at www.w3.org/TR/CSS21/cover.html.

Centering a DIV with CSS

CSS is fairly easy to learn, but accomplishing some tasks isn't intuitive. Over the past couple of years, I've gotten quite a few emails from readers struggling to center a DIV using CSS. Doing so isn't at all intuitive, but it's easy if you know how.

To center a block element such as a DIV using CSS, set the margin-left and margin-right attributes to auto. By setting the margins to auto, you are telling the browser to handle the margin settings automatically, resulting in the DIV being centered horizontally on the page.

The most efficient way of centering a DIV using this method is to create a CSS class. The following CSS code is an example of such a class.

```
.centered
{
    margin-left: auto;
    margin-right: auto;
    width: 500px;
}
```

If you apply this CSS class to a DIV, it sets the width of the DIV to 500 pixels and centers it horizontally on the page.

In the next chapter, you learn how to use the powerful CSS tools in Expression Web to take even more control over CSS.

MANAGING CSS STYLES

Expression Web's CSS Tools

In the previous chapter, we covered the basics of CSS. In this chapter, we put that knowledge to use and cover the CSS toolset in Expression Web in-depth. As you'll soon see, Expression Web's CSS toolset is powerful and easy to use.

Expression Web has several tools for managing and applying CSS styles. We'll cover them all in detail, but first, let's look at a brief overview of each one.

Apply Styles Panel

The Apply Styles panel is a convenient way to apply styles to elements in a page (see Figure 18.1). However, it's also a quick way to remove styles, add new styles, and modify existing styles. Styles are applied using the Apply Styles panel simply by clicking a style entry in the panel.

The Apply Styles panel uses color-coding for different CSS elements and even lets you select all instances of a particular CSS element on a page.

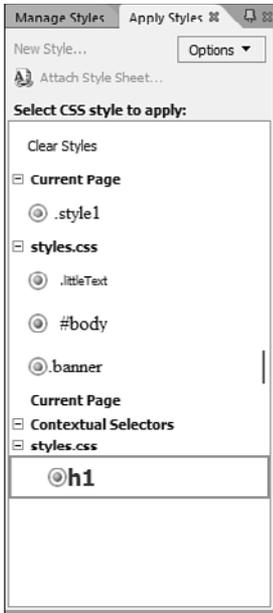


tip

By default, the Apply Styles panel is located in the lower-right of the Expression Web window.

Manage Styles Panel

The Manage Styles panel is similar in functionality to the Apply Styles panel (see Figure 18.2).

**Figure 18.1**

The Apply Styles panel makes applying styles, editing styles, and removing styles easy.

**Figure 18.2**

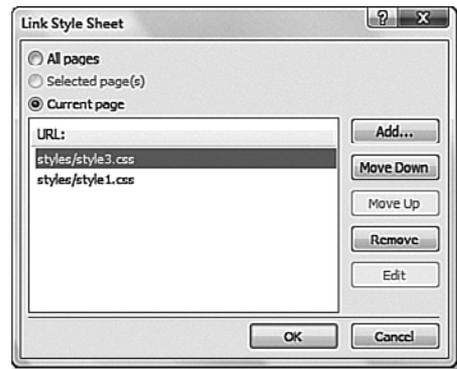
The Manage Styles panel is a quick way to locate CSS code for a particular style or to select a style for manipulation via the CSS Properties panel.

The primary difference between this panel and the Apply Styles panel is that simply clicking a style in the Manage Styles panel doesn't apply the style to the selected page element. You can apply a style by right-clicking the style and selecting Apply Style from the menu, but the Manage Styles panel is more often used to select a style so you can edit the CSS code or change CSS properties using the CSS Properties panel.

Link Style Sheet Dialog

The Link Style Sheet dialog is more than just a method of adding style sheet links to one or more pages (see Figure 18.3). It also allows you to arrange the order in which style sheets are applied to a page.

Figure 18.3
Linking style sheets is accomplished using the Link Style Sheet dialog. You can adjust the order of linked style sheets as well.



The Link Style Sheet dialog also provides you with an Edit button you can use to quickly open a selected style sheet for editing.

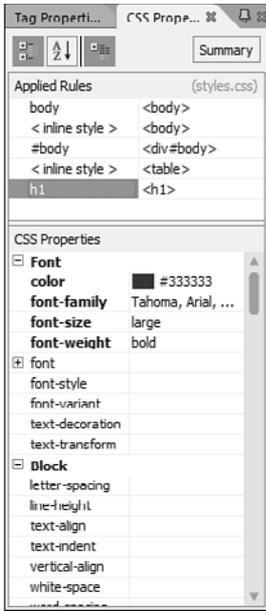
CSS Properties Panel

The CSS Properties panel provides a grid view of CSS styles applied to a page or page element (see Figure 18.4). The grid is context-sensitive, which means the CSS properties it displays are based on what is currently selected on the active page.

The CSS Properties panel can display CSS properties in categories or in alphabetical order. It displays all rules that apply to the active selection and even shows you where each style is defined.

note

To access the Link Style Sheet dialog, select Format, CSS Styles, Manage Style Sheet Links.

**Figure 18.4**

The CSS Properties panel appears to be a simple tool at first, but it's actually extremely powerful.

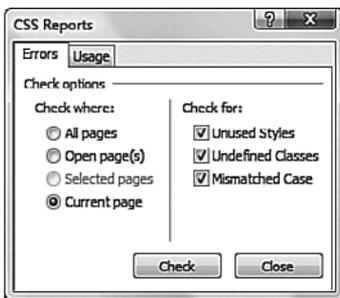
CSS Reports

CSS reporting in Expression Web is implemented using two separate dialogs: the CSS Reports dialog shown in Figure 18.5 and the CSS Reports panel shown in Figure 18.6.



note

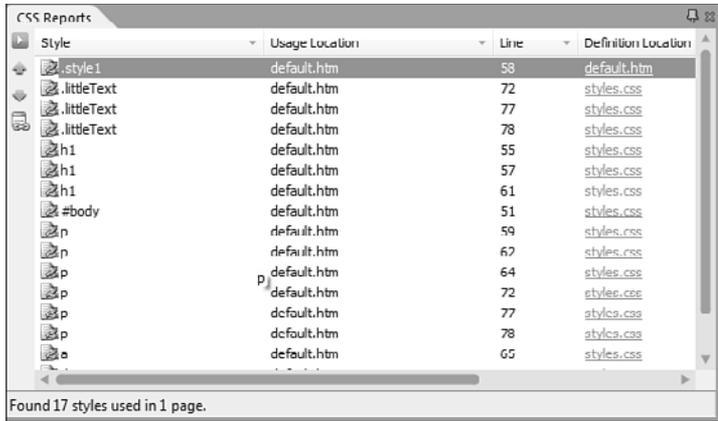
You access the CSS Reports dialog by selecting Tools, CSS Reports.

**Figure 18.5**

Expression Web provides CSS reports for style usage as well as style errors. CSS report properties are configured in the CSS Reports dialog.

Figure 18.6

Results of CSS reports are displayed in the CSS Reports panel. From here, you can easily access CSS code in your pages and CSS files.



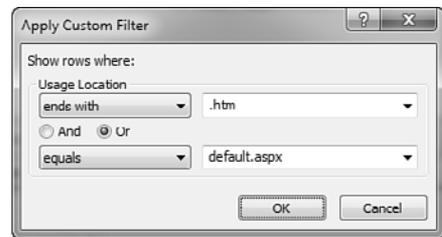
The CSS reporting feature can be filtered using the Apply Custom Filter dialog, as shown in Figure 18.7. It's a powerful way to examine CSS usage, unused CSS styles that need to be cleaned up, and any CSS errors in your site.

note

We cover CSS reports in more detail later in this chapter.

Figure 18.7

You can filter the results of CSS reports using the Apply Custom Filter dialog. Many filter options are available.



Style Builder

The Style Builder lets you easily build complex CSS styles quickly without knowing CSS (see Figure 18.8). Styles can be saved into a page or a style sheet.

Styles are broken down into multiple categories, and the options available in the Style Builder change according to the category chosen.

tip

The Style Builder is a feature that refers to the New Style dialog and the Modify Style dialog.

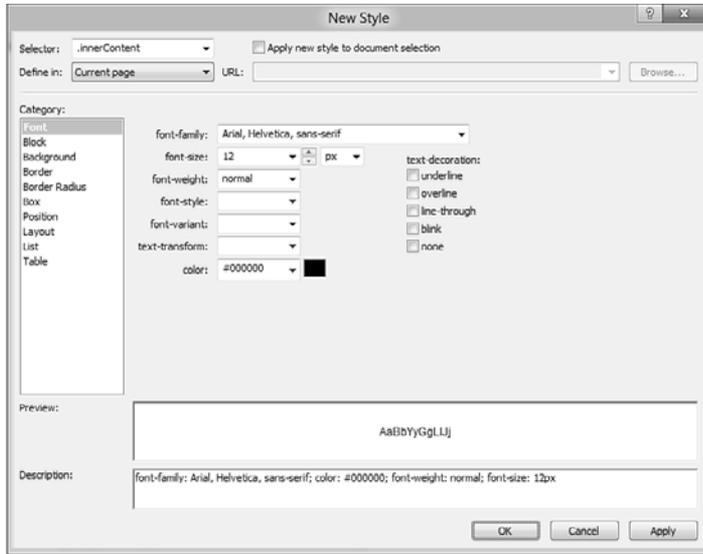


Figure 18.8
The Style Builder is a powerful way to quickly create and modify CSS styles.

Working with Styles

Perhaps the easiest way to learn how to use all the previously mentioned tools to manage styles is to use one of the site templates provided by Expression Web. These templates contain numerous predefined styles so you can use all the CSS features available in Expression Web.

Create a new site in Expression Web using the Organization 3 site template. When Expression Web has finished building the site, open the master .dwt file to explore some of the styles.

➔ *For more information on Dynamic Web Templates, see Chapter 19, “Using Dynamic Web Templates.”*

➔ *For more information on creating sites, see Chapter 2, “Creating, Opening, and Importing Sites.”*

After you've opened master .dwt, click some of the page elements. As you do, notice that the CSS Properties, Apply Style, and Manage Styles panels change based on your selection. The selected style is highlighted in the top of the CSS Properties panel and in the Manage Styles panel. The lower part of the Apply Styles panel displays the selected style along with a preview.



tip

The master .dwt file is an Expression Web Dynamic Web Template and is used as a template for other pages in the site.



note

If you previously closed the master .dwt file, open it again so you can use it as you work through the rest of this chapter.



No Styles Appear in Panel

To see styles in the panels, you must have styles applied to the page that is currently active in Expression Web. You should link a style sheet, import a style sheet, or define some styles directly in the page first. You then can manage those styles using the CSS panels in Expression Web.

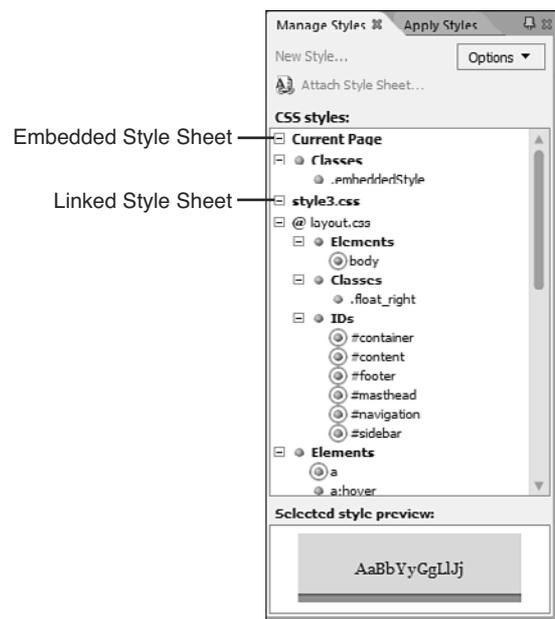
Let's take an in-depth look at each of the tools we briefly covered earlier.

Using the Manage Styles Panel

The Manage Styles panel shows a list of styles from the current page and from linked style sheets. As shown in Figure 18.9, each style sheet linked to the active page will have an expandable section inside the Manage Styles panel, as will any embedded style sheets.

Figure 18.9

The Manage Styles panel sections off each style sheet. In this case, there is one linked style sheet and one embedded style sheet.



You might notice that `@layout.css` appears under the `style3.css` style sheet in the Manage Styles panel. The `layout.css` style sheet is an imported style sheet inside `style3.css`. By using an imported style sheet, common styles can be created once in a single style sheet and then imported into other style sheets using the `@import` directive.



The Manage Styles panel displays imported style sheets with an `@` character in front of them. Clicking the plus sign next to an imported style sheet displays all the styles defined in the imported CSS file.

The following code appears at the top of `style3.css` and imports all the styles defined in `layout.css` into the `style3.css` style sheet:

```
@import url("layout.css");
```

CSS Inheritance and Cascading Order

As you've already seen, CSS rules can come from many places. You can have a CSS file linked to a page; you can have CSS styles embedded within a page; and you can have CSS styles applied inline with the HTML code itself. CSS styles can also be applied in other ways. For example, your browser internally applies a style to every page that is displayed without you even knowing it. That's why all pages that don't explicitly specify a background color appear with a white background in Internet Explorer.

The CSS landscape of a page can get complex, and that's why rules define how CSS gets applied. There are two basic concepts to CSS rules as they are applied: inheritance and cascading order.

Inheritance specifies that an HTML element takes on the style of the parent element. For example, consider the following HTML code:

```
<h3 style="font-size: 12pt;">This is a Heading with <em>emphasis</em></h3>
```

Note that the `<h3>` element has a style attribute that sets the font size to 12 pt. Because the `` element doesn't explicitly set a font size, it inherits the font size of its parent element and appears in a 12-point font.

The cascade order, on the other hand, determines the order in which CSS rules are applied. In the simplest terms, the cascade order dictates that an embedded style overrides a style defined in a linked style sheet and an inline style overrides both an embedded style and a linked style. Many web designers consider that description of the cascade order to be complete, but in fact, there are other rules to the cascade order that can cause subtle changes in how your page is rendered.

For a complete description, it's best to refer to the W3C's CSS specification at www.w3.org/TR/REC-CSS2/cascade.html.

Each style in the Manage Styles panel is represented by a colored dot. The four colors to represent styles are as follows:

- **Red**—Represents a CSS ID
- **Green**—Represents a CSS class
- **Blue**—Represents a selector
- **Yellow**—Represents an inline style



tip

The colored dots for styles in use are circled in gray. Unused styles do not display a circled colored dot.



For more information on CSS classes, style sheet links, and other terminology related to CSS, see Chapter 17, "Creating Style Sheets."

CSS IDs

In the previous chapter, we covered CSS classes. This chapter introduces the CSS ID. A CSS ID is almost exactly like a CSS class except an ID refers to a single instance of a particular selector. A CSS class can be used by any number of selectors.

A CSS ID is defined in a style sheet using the # prefix. For example, the following CSS code defines the sidebar CSS ID from the style3.css style sheet:

```
#sidebar {
  float: right;
  width: 380px;
  padding-top: 5px;
  padding-bottom: 5px;
  text-align: left;
}
```

To specify that a particular page element should take the formatting of this ID, you use the id attribute:

```
<div id="sidebar">
```

To control the appearance of HTML elements within a block element defined as a particular ID, simply specify the selector after the ID name. For example, the following CSS code formats all <h1> elements within the sidebar div defined previously:

```
#sidebar h1 {
  font-size: 18px;
}
```

CSS IDs are a powerful formatting technique because they provide a convenient way of formatting specific portions of your pages.

The Options button in the Manage Styles panel displays the Options menu, allowing you to choose which styles appear in the list and how they are displayed.

The following items on the Options menu control how styles are categorized:

- **Categorize by Order**—When this item is selected, styles are categorized in the order in which they appear in the style sheet. Note that each style sheet is still listed separately.
- **Categorize by Element**—When this item is selected, styles for HTML elements are arranged in alphabetical order. (CSS IDs and classes are still arranged in the order in which they appear in the style sheet.) Inherited styles for a particular HTML element appear nested under the element, as shown in Figure 18.10.

**Figure 18.10**

When Categorize by Element is selected, each HTML element is listed with inherited styles nested underneath.

- **Categorize by Type**—When this item is selected, each type of style (elements, classes, and IDs) is listed in a nested structure, as shown in Figure 18.11.

The following items on the Options menu control the styles that are displayed in the Manage Styles panel:

- **Show All Styles**—When this item is selected, styles from all style sheets are displayed in the list. The colored dots for used styles are circled in gray.
- **Show Styles Used in Current Page**—When this item is selected, only styles used in the current page are visible.
- **Show Styles Used on Selection**—When this item is selected, only styles used on the current selection are visible. If nothing is selected in the active document, the Manage Styles panel will be empty.

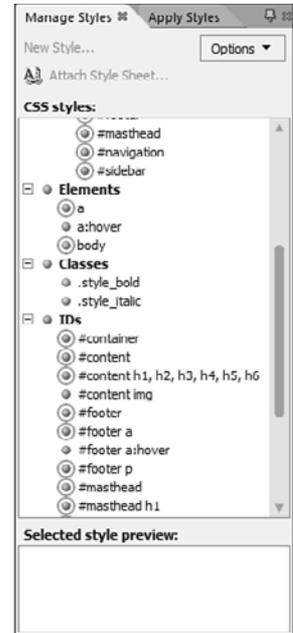


Selected Styles Don't Display Preview

If you've selected a page element you know has a style applied to it, but don't see the preview of the selected style in the panel, make sure you have selected the page element in Design View. If you select a page element in Code View, the style preview will not work.

Figure 18.11

When Categorize by Type is selected, each type of style acts as a top-level member of a nested hierarchy.



The Show Styles Used on Selection menu item is a powerful way of identifying where certain styles originate. If you select the `div` with an ID of `content` on the `master.dwt` page, the Manage Styles panel appears, as shown in Figure 18.12. This allows you to tell at a glance that the `content` CSS ID is defined in both the `layout.css` and `style3.css` CSS files.

Figure 18.12

Using the Manage Styles panel, you can find out exactly where certain styles originate. In this case, the `content` CSS ID is defined in both `layout.css` and `style3.css`.



This capability is even more powerful when a particular element is being influenced by CSS styles that are actually applied to higher-level elements or when an element is taking on CSS styles from multiple places. Suppose you have a paragraph that unexpectedly appears in a blue font and you need to track down the CSS style that is causing that to happen.

Without the Manage Styles panel, locating a source of formatting could be fairly difficult because of inheritance and the cascading rules for CSS. Using the Manage Styles panel, you can simply click inside the paragraph and immediately locate the CSS code that's causing the blue text.

The final two items on the Options menu are Separate Grouped Selectors and Display Selected Style Preview.

By default, Expression Web groups selectors affected by a particular style. For example, in Figure 18.11 (shown previously), the `#content` CSS ID is followed by several HTML heading selectors. The CSS code that defines this `#content` ID is as follows:

```
#content h1,h2,h3,h4,h5,h6 {
    color: #a0522d;
}
```

The Separate Grouped Selectors option breaks the selectors out of the group so they appear as separate items, as shown in Figure 18.13.

The Display Selected Style Preview menu item toggles the Selected Style Preview portion of the Manage Styles panel.

Using the Apply Styles Panel

The Apply Styles panel is similar in functionality and design to the Manage Styles panel. However, as its name indicates, it is intended primarily for applying styles to page elements.

Similar to the Manage Styles panel, the Apply Styles panel uses color-coded dots to indicate different types of styles. It also breaks out style sheets and embedded styles into collapsible sections. You can easily apply any style to a page simply by clicking the name of the style in the Apply Styles panel.

As you click an area of a page, the styles applied to the area are highlighted in the Apply Styles panel. If the area you've clicked is an HTML selector styled in a style sheet, the selector and any associated styles appear at the bottom of the Apply Styles panel in the Contextual Selectors section, as shown in Figure 18.14.



tip

The Selected Style Preview portion of the Manage Styles panel shows a preview of only the formatting affected by the selected style. What you see in the actual page might differ due to formatting caused by the specific HTML element or by direct formatting, and so on.



note

The Manage Styles panel can be categorized by Order (the default), Element, or Type. In Figure 18.12, I've categorized it by Type.



tip

You can right-click a style in the Manage Styles panel and select Go to Code to immediately open the CSS file where a style is defined. Double-clicking the style also takes you directly to the style.

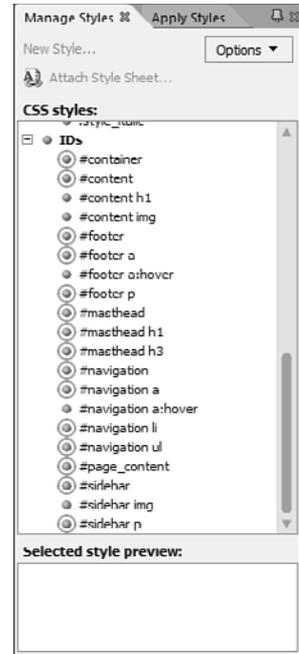


note

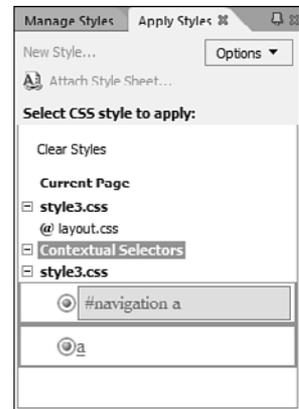
Right-clicking a style in the Manage Styles panel provides additional capabilities such as modifying styles or creating new styles. These capabilities are shared by the Apply Styles panel; we cover them in detail later in this chapter.

Figure 18.13

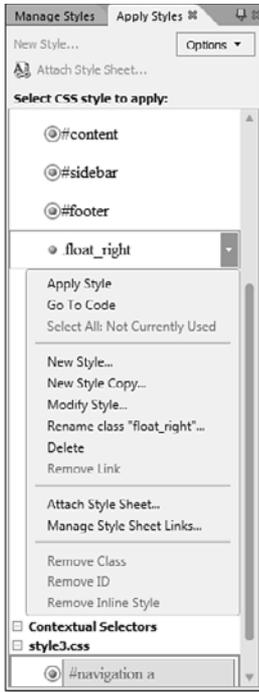
When the Separate Grouped Selectors menu option is checked, HTML selectors that are grouped with a particular CSS element are shown as separate items.

**Figure 18.14**

The Contextual Selectors section of the Apply Styles panel displays styles that are defined using HTML selectors.



Hovering over a style in the Apply Styles panel reveals an arrow button at the right edge of the style name. Clicking that button displays a menu, which provides many ways of interacting with the style (see Figure 18.15).

**Figure 18.15**

Working with styles using the Apply Styles panel is similar to using the Manage Styles panel. The menu shown here is available in both panels.

The following menu items are available when you click the arrow button next to a style:

- **Apply Style**—Applies the selected style to the current selection.
- **Go To Code**—If the selected style is in an external style sheet, Expression Web opens the style sheet and scrolls to the CSS code for the style. If the selected style is an embedded or inline style, Expression Web switches to Split View and displays the corresponding CSS code in the page.
- **Select All x Instances**—Selects all instances of the selected style in the current page. The x is replaced with the number of actual instances of the selected style. If the selected style is unused, the menu item displays Select All: Not Currently Used and is disabled.
- **New Style**—Opens the Style Builder so a new style can be created. The new style is not based on the selected style.
- **New Style Copy**—Opens the Style Builder so a new style can be created. The new style is based on the selected style.
- **Modify Style**—Opens the Style Builder so the selected style can be modified.

**tip**

You can also access the menu by right-clicking a style or style sheet in the Apply Styles panel.

- **Delete**—Deletes the currently selected style. If multiple styles apply to the selected page element, this menu item is disabled. When deleting a style, Expression Web prompts you for confirmation before the style is deleted.
- **Remove Link**—This menu item is enabled only when you right-click the name of a linked style sheet. When clicked, it removes the link to the selected style sheet.
- **Attach Style Sheet**—Opens the Attach Style Sheet dialog so a style sheet can be attached.
- **Manage Style Sheet Links**—Opens the Link Style Sheet dialog so existing style sheet links can be configured. The Link Style Sheet dialog also allows you to attach new style sheets.
- **Remove Class**—Removes the application of the selected CSS class from the selected page element. Removing the class does not alter the CSS code; it only removes the class assignment from the currently selected page element. This menu item is available only when a CSS class is applied to the currently selected page element and when the currently selected style in the Apply Styles panel is a CSS class.
- **Remove ID**—Removes the application of the selected CSS ID from the selected page element. Removing the ID does not alter the CSS code. This menu item is available only when the selected page element is styled using a CSS ID and when the currently selected style in the Apply Styles panel is a CSS ID.
- **Remove Inline Style**—Removes inline styles applied to the currently selected page element. This menu item is available only when an inline style is applied to the currently selected page element and when the currently selected style in the Apply Styles panel is an inline style.



caution

A CSS ID should be applied only once per page. If you attempt to apply a CSS ID to a second element in a page, Expression Web displays a warning.



Cannot Remove CSS Elements

Before you can remove a CSS class, ID, or inline style, you must select the page element from which you want the style removed. Removing a style doesn't really remove the style from the style sheet where it exists; it simply removes the application of that style from any selected page elements.

First, select one or more instances of the style on the page; you will then be able to remove the style from those instances.

Using the CSS Properties Panel

The Apply Styles and Manage Styles panels focus on working with one or two styles. The CSS Properties panel, on the other hand, is more of an all-in-one tool that makes it easy to manage all the styles applied to your page.

The CSS Properties panel is divided into two sections, as shown in Figure 18.16. The top section is called the Applied Rules section, and it shows all the CSS rules that apply to the selected page element. Rules are displayed in the order in which they are applied to the page.

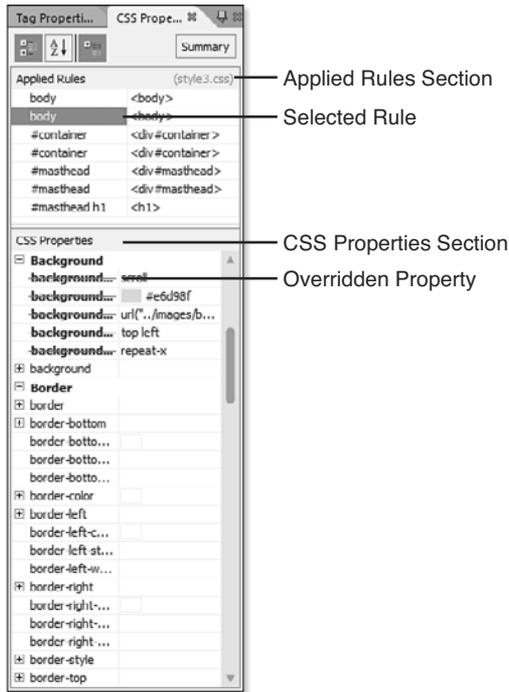


Figure 18.16

The CSS Properties panel is among Expression Web's most powerful CSS tools.

The bottom part of the CSS Properties panel displays the CSS style properties for the selected rule. Properties defined for the selected rule appear in bold text, and overridden properties appear with a red line through them.

When a rule is selected in the CSS Properties panel, Expression Web displays where the rule is defined in the Applied Rules header of the CSS Properties panel. If the rule is defined in a linked style sheet, the name of the style sheet is displayed as a link to open the style sheet when clicked. (Note that in Figure 18.16, `style3.css` appears as a link that can be clicked to open the CSS file where the rule is defined.) If the link is defined in an embedded style sheet, Expression Web displays (Current Page) as the location where the rule is defined.

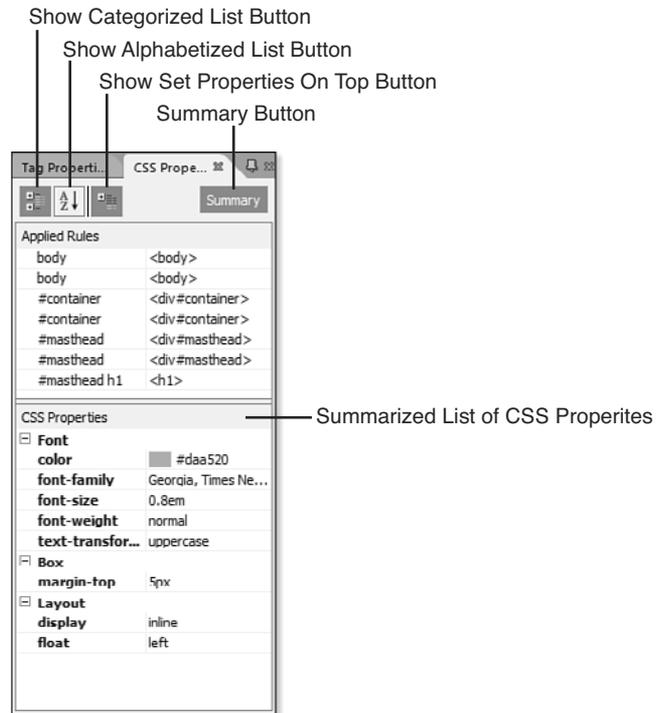
As shown in Figure 18.17, a series of four buttons appears at the top of the CSS Properties panel:

note

Inline styles displayed in the Applied Rules section of the CSS Properties panel do not display anything in the title bar to indicate where the rule is defined because they are always defined on the element itself.

Figure 18.17

The series of buttons at the top of the CSS Properties panel makes adjusting the display easy.



- **Show Categorized List**—When this button is selected, the CSS properties are displayed in collapsible categories.
- **Show Alphabetized List**—When this button is selected, the CSS properties are displayed in alphabetical order.
- **Show Set Properties On Top**—When this button is selected, CSS properties explicitly set for the selected rule are always displayed at the top of the list.
- **Summary**—This button causes the CSS Properties panel to display a summarized list of all properties applied to the selected page element regardless of which CSS rule is selected in the panel.

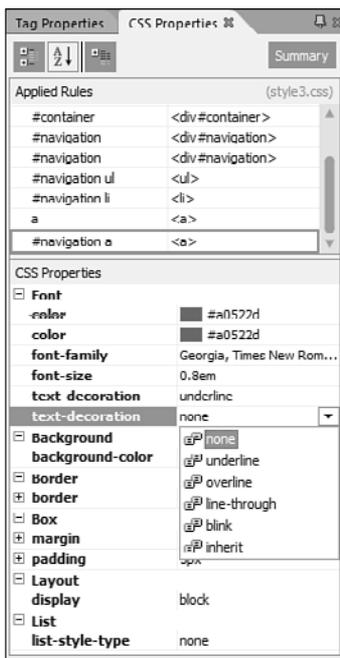
The Summary button is one of the most powerful CSS tools available in Expression Web because it makes it easy to find out exactly what is affecting the formatting of a selected page element. In Figure 18.18, you can see that the selected element has two properties for `text-decoration`, but one of them has a line through it, indicating it has been overridden. By hovering over the overridden property, you can easily determine which other property is overriding it. As you define your CSS styles, you'll find yourself using this feature often because it lets you easily track down CSS properties that don't seem to be taking effect when they ought to.

Modifying CSS properties using the CSS Properties panel couldn't be easier. You don't have to know CSS code and syntax to modify properties because most of the properties listed expand to display the valid values in a drop-down list. To modify a property, simply select the new value from the list, as shown in Figure 18.18.

As you select properties in the panel, the rule to which that property applies is automatically highlighted. This enables you to ensure that you are changing the correct property.

**tip**

The styles that can be edited in the Style Builder are the same styles that are available in the CSS Properties panel. The Style Builder is just a different interface into the same CSS properties.

**Figure 18.18**

Changing CSS properties is accomplished using the drop-down menu of valid property assignments.

When you modify a style that exists in a CSS file, Expression Web automatically opens and edits the CSS file. You then are prompted to save those changes to the CSS file when the page it is linked to is saved.

Using the Style Builder

The Style Builder is a common name for the New Style and Modify Style dialog boxes. When creating a new style, the New Style dialog is displayed; when modifying an existing style, the Modify Style dialog is displayed. Both dialogs are the same, but the options available are different based on whether you are creating a new style.

**caution**

Keep in mind that a modification to a style sheet linked to the current page also modifies that same style in other pages that link to the same style sheet.

In Figure 18.19, a new style is being created. Note that the Define in drop-down is set to Existing Style Sheet and the `style3.css` style sheet is selected. You can also choose to add the new style to a new style sheet or to the current page. When adding styles to the current page, they are added to an embedded style sheet.

Figure 18.19
The Style Builder is a comprehensive dialog for creating new styles or modifying existing styles without having to write CSS code.



The Selector drop-down contains a list of all the HTML selectors, but you also can enter a CSS class name or ID into the text box portion of the drop-down to create a new CSS class or ID.

To configure a particular category of CSS property, select the category from the Category list, as shown in Figure 18.19; then configure the style as desired. Categories for which style properties have been configured appear in bold text. After you've finished configuring a style, click OK to add it or apply your modifications. If the Apply New Style to Document Selection check box is checked when creating a new style, the style will automatically be applied to the selected element in the page. Otherwise, the style will simply be added to the style sheet as specified.



tip

The Define In drop-down, Apply New Style to Document Selection check box, and URL text box are disabled when modifying an existing style.

Using the Attach Style Sheet and Link Style Sheet Dialogs

Styles sheets can easily be added to the current page or to all pages using the Attach Style Sheet and Link Style Sheet dialogs. To access the Attach Style Sheet dialog, select Format, CSS Styles, Attach Style Sheet. To access the Link Style Sheet dialog, select Format, CSS Styles, Manage Style Sheet Links.

The Attach Style Sheet dialog is used to either attach a style sheet link to one or more pages or to import styles from a style sheet into one or more pages (shown in Figure 18.20).

**tip**

When creating new CSS classes and IDs, be sure to append a . to CSS class names and a # to CSS IDs.

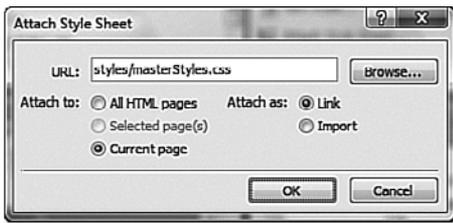


Figure 18.20

CSS style sheets are attached or imported using the Attach Style Sheet dialog. You can add style sheets to the current page, pages selected in the Folder List, or to all pages.

Imported styles are added to a page using the CSS `@import` directive. For example, if a style sheet located at `styles/style1.css` is imported via the Attach Style Sheet dialog, the following CSS code is added to the page:

```
<style type="text/css">
  @import url('styles/style1.css');
</style>
```

The Link Style Sheet dialog is specifically designed to be used with linked external style sheets (see Figure 18.21). Not only can you add and remove style sheet links using this dialog, but you can also arrange the order of style sheets.

The CSS tools in Expression Web comprise a powerful CSS feature set that has not been seen in any previous Microsoft web development platform. Not only can you easily create styles and modify existing styles, but you also can investigate the source of formatting on a page using the CSS Properties panel. By using these tools together, you can take control of CSS in your site.

**tip**

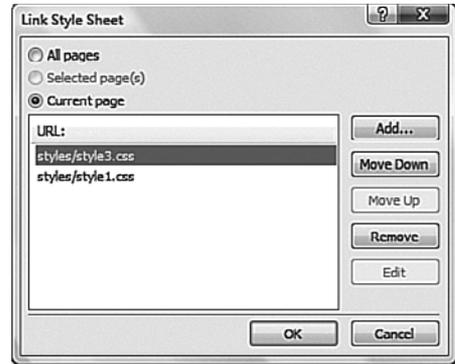
Imported style sheets are useful when you want to compartmentalize a specific set of CSS styles that can be applied easily to a page or series of pages. In fact, many of the Expression Web templates use imported CSS style sheets to apply common formatting.

**tip**

Linked style sheets are applied in the order in which they appear in the Link Style Sheet dialog, so a duplicate style in a style sheet lower in the list overrides any previous instances of that style.

Figure 18.21

The Link Style Sheet dialog provides a clean interface for linking style sheets and arranging their order.



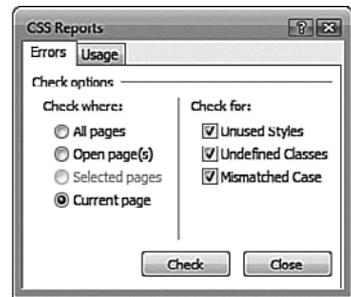
CSS Reports

Using CSS makes your site much easier to maintain, but if you don't keep your CSS code maintained as well, it can become unwieldy. Fortunately, Expression Web's CSS reporting feature can help you identify CSS problems quickly.

To access CSS reports, select Tools, CSS Reports to display the CSS Reports dialog, as shown in Figure 18.22. You have the option of checking all pages, all open pages, pages that are currently selected, or only the current page.

Figure 18.22

The CSS reporting feature is a powerful tool for finding problems in your CSS code.



Checking for CSS Errors

The Errors tab, shown previously in Figure 18.22, allows you to easily locate the following types of errors in your CSS code:

- **Unused Styles**—Styles that exist in a style sheet but are not used
- **Undefined Classes**—Styles that are used but not defined in a style sheet
- **Mismatched Case**—A mismatch in case between a CSS class or ID in a page and the case used in a style sheet

After choosing the errors you want to check, click the Check button to run the report. Results are displayed in the CSS Reports panel, as shown in Figure 18.23.

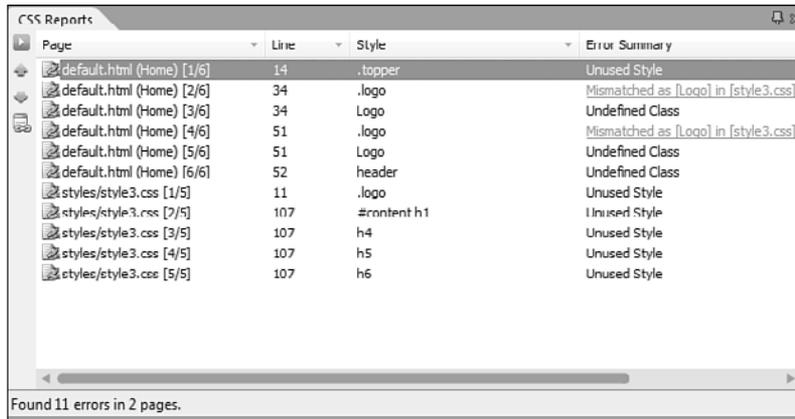


Figure 18.23
The CSS reports panel displays CSS report results, so you can easily locate and correct errors.

To locate an error displayed in the CSS Reports panel, double-click the error or right-click the error and select Go to Page, as shown in Figure 18.24. Expression Web opens the page containing the error, and the offending code is highlighted so you can easily correct it without having to dig through the code.

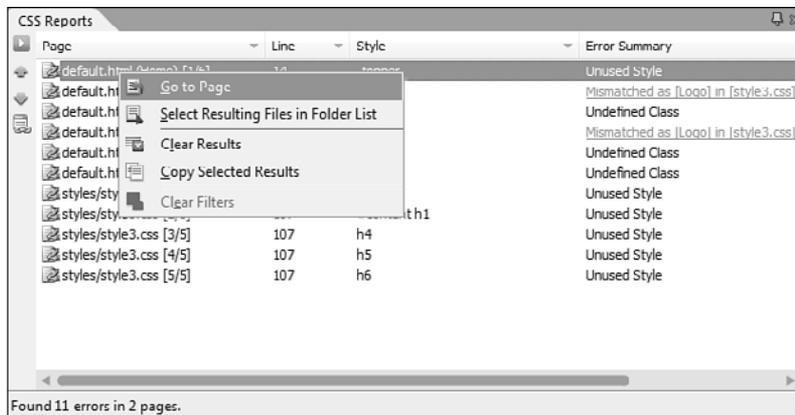
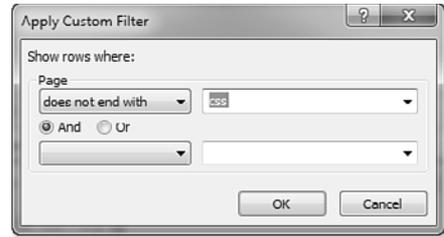


Figure 18.24
Easily locate a CSS error by double-clicking the error or using the right-click menu.

As with other panels, the CSS Reports panel can be sorted by clicking a column header. It can also be filtered by clicking the small black arrow at the right edge of a column and selecting Custom to display the Apply Custom Filter dialog, as shown in Figure 18.25.

Figure 18.25

The CSS Reports panel can be filtered using the Apply Custom Filter dialog. In this case, CSS files are filtered out of the report.



Checking CSS Usage

The Usage tab in the CSS Reports dialog allows you to run a report that shows how CSS is used in a page (see Figure 18.26). The following options are available:

- **Class Selectors**—When this item is checked, CSS reports shows which CSS class selectors are in use.
- **ID Selectors**—When this item is checked, CSS reports shows which CSS ID selectors are in use.
- **Element Selectors**—When this item is checked, CSS reports shows all HTML element selectors.



tip

The Element Selectors option generates a lot of output. Use it only if you absolutely need a report of all your HTML selectors.

Figure 18.26

The Usage tab enables you to run reports that detail how CSS is being used in your site.



Arranging CSS Styles

As you've seen, you have plenty of choices to make when it comes to using CSS in your site. One of the choices with the greatest impact is whether to define a style in an external or an embedded style sheet. Invariably you will end up needing to move some of your CSS classes or IDs from an embedded style sheet to an external style sheet, and vice versa. You might also want to move classes or IDs from one external style sheet to another.

Moving CSS classes and IDs is fast and easy using the Manage Styles panel in Expression Web. In Figure 18.27, an embedded CSS class called `.redLink` exists in the current page. The `styles.css` external style sheet is also linked to the page. To move the `.redLink` CSS class from the embedded style sheet in the current page to the `styles.css` external style sheet, simply drag the `.redLink` class to the `styles.css` header, as shown in Figure 18.27.

You can also drag and drop HTML elements. When an HTML element is dropped onto a style sheet, Expression Web creates a CSS entry for the selector.

Using this method, you can quickly and easily manage your CSS content with minimal effort.



Figure 18.27

CSS IDs and classes can be moved from a page to an external style sheet and vice versa by dragging and dropping the style in the Manage Styles panel.

USING DYNAMIC WEB TEMPLATES

An Introduction to Dynamic Web Templates

There are millions of sites on the Internet, and one thing they have in common is that they are each made up of pages that contain a lot of duplicated content. A typical site is composed of pages that consist of a common framework. That common framework plays an integral role in the success of a site because it defines the site's look and feel.

Another trait of a successful site is freshness. Keeping a site updated is more than just adding new content. Every once in a while, you need to change things around and give the site a fresh look. This is especially important as new web technologies begin to emerge. If you want to be on the cutting edge, you've got to keep your site updated.

Updating your site is easy if you have only a few pages. However, if your site has been up for any length of time, you've likely accumulated a lot of content on many pages. Updating a site with many pages can be a big hassle, and not just because it can be time consuming. Code that looks great on one page might appear slightly different on another page. This is especially true if you are using tables in your layout. The best way to prevent layout headaches is to use the exact same layout code for each page, but your site can then become a nightmare to maintain if you have to manually reproduce the layout on each page.

This kind of problem is exactly what Dynamic Web Templates were designed to address. A Dynamic Web Template allows you to create a master page (a Dynamic Web Template) that defines your layout. When you create the Dynamic Web Template, you create areas in the page

(called *editable regions*) that contain unique content on pages that use the Dynamic Web Template. When a page is attached to the Dynamic Web Template, you are allowed to add or edit content only inside an editable region. All other areas of the page are locked, and content for those areas is provided by the Dynamic Web Template.

➔ *Dynamic Web Templates should not be confused with ASP.NET Master Pages. If you want an ASP.NET template solution, see Chapter 27, “Using ASP.NET Master Pages and User Controls.”*

Creating a Dynamic Web Template

To create a Dynamic Web Template, select File, New, Page. In the New dialog, select Dynamic Web Template, as shown in Figure 19.1, and click OK.

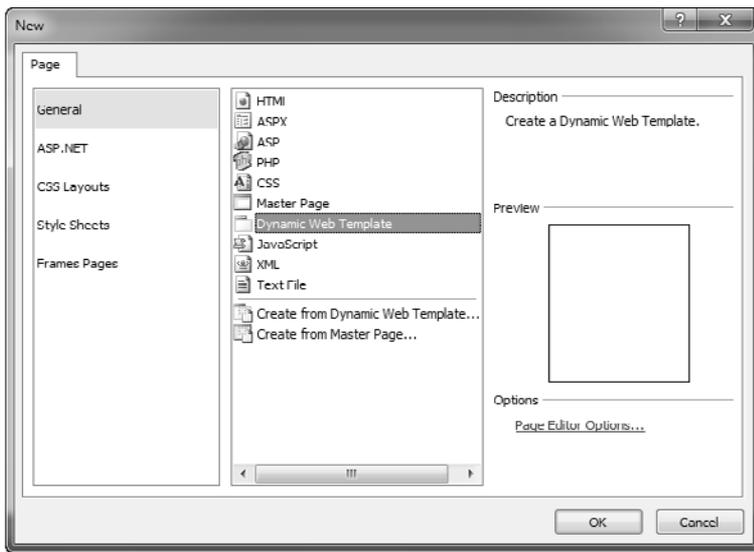


Figure 19.1
Create a Dynamic Web Template using the New dialog.

When you create a page based on a Dynamic Web Template, you will be able to add new content only to editable regions defined in the Dynamic Web Template. All other portions of the page are protected from editing. Because of that, you'll need to define a layout for the Dynamic Web Template before you add any new editable regions.



tip

A new Dynamic Web Template contains an invisible editable region called doctitle that surrounds the HTML <title> tag. This editable region allows you to change the title on pages attached to the Dynamic Web Template. Without that editable region, you wouldn't be able to change the title of any of your pages.

Creating a Page Layout

The Dynamic Web Template contains the “chrome” for your site. Menus, logos, and other common page elements are added to the Dynamic Web Template. Next, editable regions are added as placeholders for content that will be unique from page to page.

To define a page layout:

1. Create a new one-page site and create a new Dynamic Web Template. Save the Dynamic Web Template as `master.dwt`.
2. Import the `styles.css` file from the `Examples\Ch19\Files\Website` folder on the website that accompanies this book into your site and attach it to your new Dynamic Web Template.
3. Select the existing editable region and delete it. (You can use the `<DWT:editable>` Quick Tag Selector to select it easily.)
4. Add a new table.
5. Set the number of rows to 4 and the number of columns to 1.
6. Set the cellpadding and cellspacing to 0.
7. In the top row, add a logo. Use the `logo.jpg` file in the `Examples\Ch19\Files\Website\Images` folder on `informit.com/register`.
8. Select the second table row and add a class attribute with a value of `horizRule`.
9. In a real site, the second row would contain a menu. For this example, enter some text in place of the menu (see Figure 19.2).
10. In the third row, add a new table with one row and one column.
11. Set the width of the new table to 95%.

➡ *For more information on using tables, see Chapter 5, “Using Tables.”*

➡ *For more information on formatting pages with CSS, see Chapter 17, “Creating Style Sheets.”*

You now have a basic layout for a site. Your Dynamic Web Template should look similar to Figure 19.2.

Adding Editable Regions

To add page-specific content, you need to define some editable regions for the Dynamic Web Template. Let’s add two editable regions:

1. Place the insertion point inside the third table row.
2. Select Format, Dynamic Web Template, Manage Editable Regions.

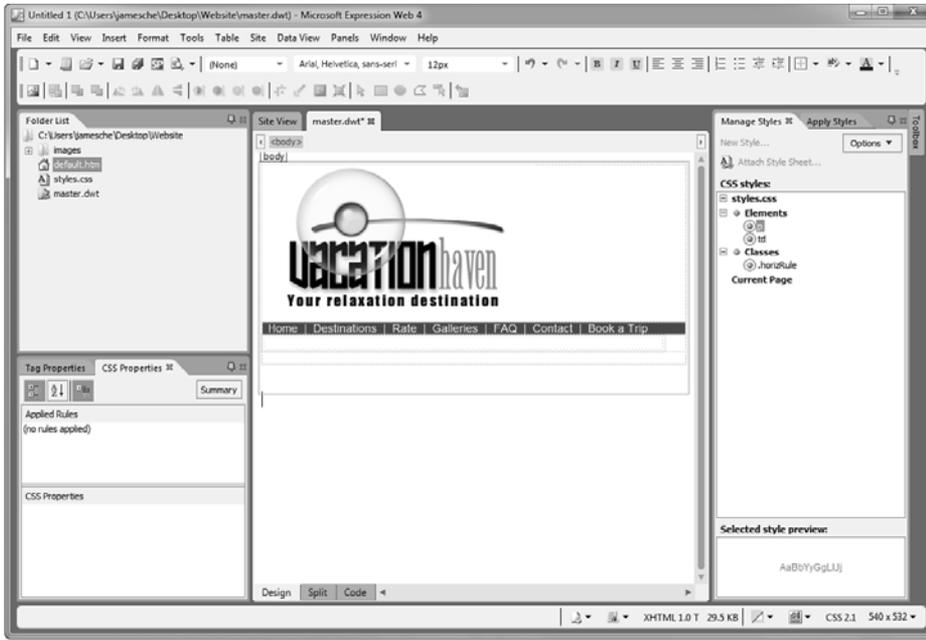


Figure 19.2 A dynamic Web Template defines the common features of a site such as a logo and menu system.

3. In the Editable Regions dialog, enter **maincontent** for the region name and click the Add button, as shown in Figure 19.3.
4. Click the Close button.
5. Right-click inside the fourth table row and select Manage Editable Regions from the context menu. (This is an alternative method of accessing the Editable Regions dialog.)
6. In the Editable Regions dialog, enter **bannerad** for the region name and click the Add button.
7. Click the Close button.
8. Save the Dynamic Web Template.

You now have two visible editable regions in the Dynamic Web Template. (The doctitle editable region is still on the page, but it's not a visible editable region.) The **maincontent** editable region will be used for the main content and the **bannerad** editable region will be used for a banner ad that will appear at the bottom of the page. Your Dynamic Web Template should now look similar to Figure 19.4.



tip

There's a bug in Expression Web 4 SP2 that causes editable regions to appear with a grayed-out background, making them indistinguishable from non-editable regions. This problem is corrected in SP2a. You can get SP2a from support.microsoft.com/kb/2635101.

Figure 19.3
Editable regions are added and managed using the Editable Regions dialog.

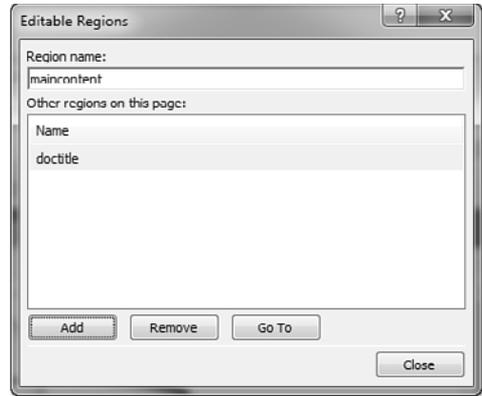


Figure 19.4
The Dynamic Web Template contains two editable regions, one for main content and one for an ad banner.



The maincontent Editable Region

The bannerad Editable Region

Attaching a Dynamic Web Template

To use a Dynamic Web Template, you must attach it to a page. The page then takes on the appearance of the Dynamic Web Template. Custom content for that page can be entered into editable regions. All other content is protected and cannot be edited.

Attaching to an Existing Page

Let's attach `master.dwt` to the `default.htm` page:

1. Open `default.htm`.
2. If you're not in Design View, switch to it so you can easily see the effect of attaching the Dynamic Web Template.
3. Select Format, Dynamic Web Template, Attach Dynamic Web Template.
4. From the Attach Dynamic Web Template dialog, select the `master.dwt` Dynamic Web Template, as shown in Figure 19.5.

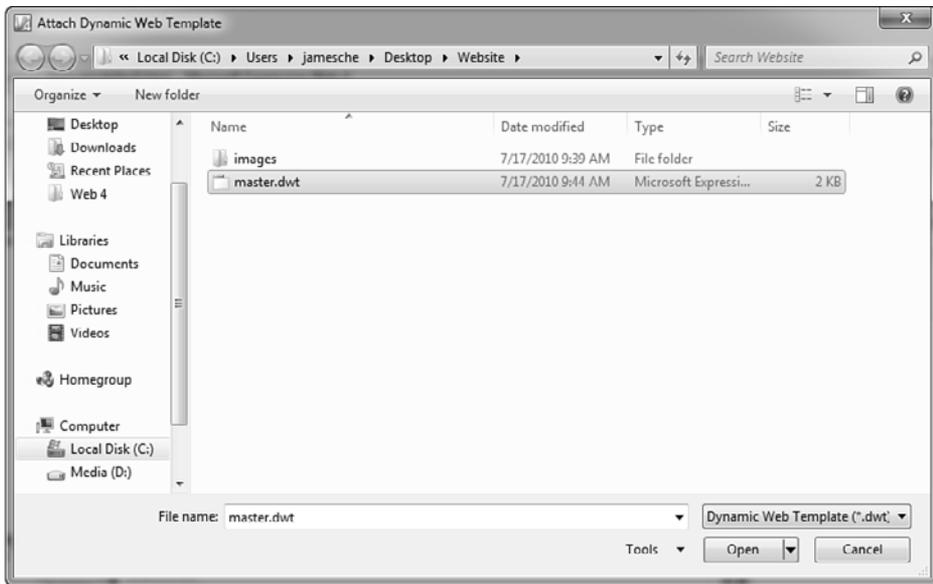


Figure 19.5
The Attach Dynamic Web Template dialog displays all the Dynamic Web Templates in your site.

5. Click Open to attach the Dynamic Web Template. Expression Web warns you that this action will replace content, as shown in Figure 19.6.
6. Click Yes to attach the Dynamic Web Template.

Figure 19.6
When you first attach a Dynamic Web Template to an existing page, Expression Web warns you that some content will be replaced.



7. Expression Web displays a dialog informing you that the file has been updated. Click Close to return to the `default.htm` page.

The `default.htm` page should now look like the Dynamic Web Template. However, notice that all content outside the editable regions is locked and cannot be edited.



tip

You can provide default content for editable regions by adding content inside the editable regions in the Dynamic Web Template. When a page is attached to the Dynamic Web Template, the editable regions will contain the content from the Dynamic Web Template. That content can then be changed in individual pages.



tip

You can right-click a Dynamic Web Template in the Folder List and select New from Dynamic Web Template from the menu that appears to create a new page based on a Dynamic Web Template.

Providing Default Content for Editable Regions

At first glance, it can seem counterintuitive to provide default content in an editable region. However, sometimes doing so can make maintaining a site easier.

Consider a scenario in which your pages contain a menu that provides site navigation for your pages. If you include that menu outside an editable region, any changes to the menu will be applied to all pages in the site. However, by including the menu inside an editable region, you can use the default menu automatically when pages are attached to your Dynamic Web Template, but you also have the option of making a change to the menu on individual pages if necessary by overriding the Dynamic Web Template's menu on those pages.

Attaching to a New Page

You can attach a Dynamic Web Template to a page when the page is created. Do the following:

1. Select File, New, Page.
2. From the New dialog, select Create from Dynamic Web Template, as shown in Figure 19.7.
3. Click OK.
4. In the Attach Dynamic Web Template dialog, select the Dynamic Web Template you want to use.
5. Click Open to create the page.

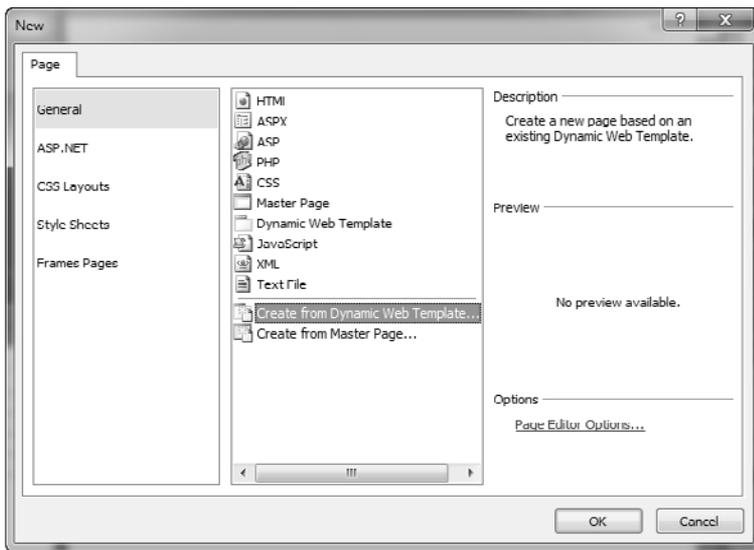


Figure 19.7

The New dialog allows you to select a Dynamic Web Template when a page is created.

Updating a Site with Dynamic Web Templates

One of the greatest advantages of using Dynamic Web Templates is that they make applying site-wide changes easy by simply modifying the Dynamic Web Template and then performing an update on all attached pages.

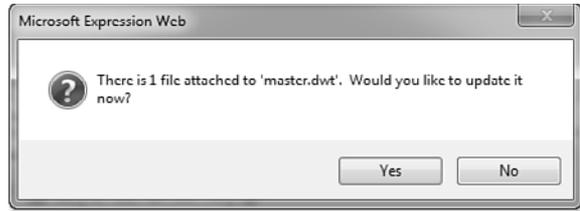
Modifying a Dynamic Web Template

When you save a modified Dynamic Web Template, Expression Web notifies you if there are any attached pages and asks whether you want to update them, as shown in Figure 19.8. If you click Yes, Expression Web automatically updates all pages attached to the Dynamic Web Template.

note

When you update a page attached to a Dynamic Web Template, none of the content in that page's editable regions is modified. Only the content provided by the Dynamic Web Template is updated.

Figure 19.8
You are prompted to update attached pages when you save a modification to a Dynamic Web Template.



Dynamic Web Template Menu Items Disabled

If you have a site open but can't click some of the Dynamic Web Template menu items, it's likely that the site you are using is hosted on a Web server running the FrontPage Server Extensions. You cannot use all Dynamic Web Template features if the site open in Expression Web is using the FrontPage Server Extensions.

The best way to work around this problem is to design your site locally as a disk-based site and then publish the site to the web server.

➔ *For more information on disk-based sites and the FrontPage Server Extensions, see Chapter 2, "Creating, Opening, and Importing Sites."*

If you choose not to update attached pages when the Dynamic Web Template is saved, you can update them later by selecting Format, Dynamic Web Template, Update All Pages. When the update has completed, a dialog is displayed, showing a report of the update, as shown in Figure 19.9.

You can select one or more pages and update only those pages by selecting Format, Dynamic Web Template, Update Selected Pages. Expression Web then updates only the pages you have selected.

Alternatively, you can select or open a Dynamic Web Template and select Format, Dynamic Web Template, Update Attached Pages to update all pages attached to the selected or opened Dynamic Web Template.



caution

Updating all pages attached to a Dynamic Web Template cannot be undone. Before you choose to update all pages, make sure that you want to perform the update because Expression Web automatically saves any pages it updates without confirmation.

Modifying an Attached Page in Code View

Content in a page attached to a Dynamic Web Template can be edited in Design View only if that content is within an editable region. You can, however, edit other content in the attached page by switching to Code View and editing the code itself.

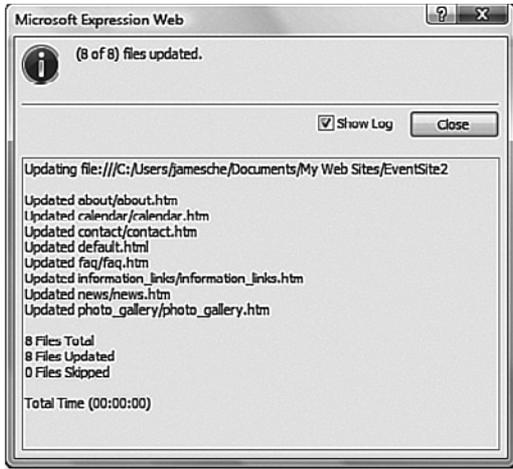


Figure 19.9
Expression Web can display a report of updates to attached pages.

When you view a page attached to a Dynamic Web Template in Code View, Expression Web highlights any code provided by the Dynamic Web Template in orange. When you make any changes to this code, switching views or saving the page causes Expression Web to notify you that you have made changes to a noneditable region of the page, as shown in Figure 19.10. At that point, you have two choices:

- **Always Restore Non-editable Content While Editing This Page**—When this option is selected, Expression Web restores the original content from the Dynamic Web Template. Any future code edits outside an editable region are restored without notification. The purpose of this option is to protect you from inadvertent code changes to content provided by the Dynamic Web Template.
- **Keep All Changes**—Keeps your code changes, allowing you to modify code outside an editable region.

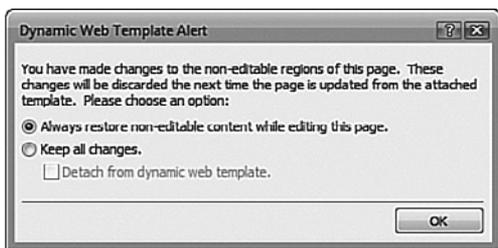


Figure 19.10
Expression Web will not allow you to modify protected content from a Dynamic Web Template unless you explicitly tell it to let you.

**tip**

When you select the option to always restore noneditable content, Expression Web restores Dynamic Web Template code without notification. However, this setting affects only the current editing session. If you open the page later and make changes, the setting resets and Expression Web warns you if changes are made to protected code.

If you choose to keep your changes, you can also check the Detach from Dynamic Web Template check box to detach the page from the Dynamic Web Template. If you choose this option, the link between the page and Dynamic Web Template will be severed. You might want to select this option if you don't want the page to be affected by future changes to the Dynamic Web Template.

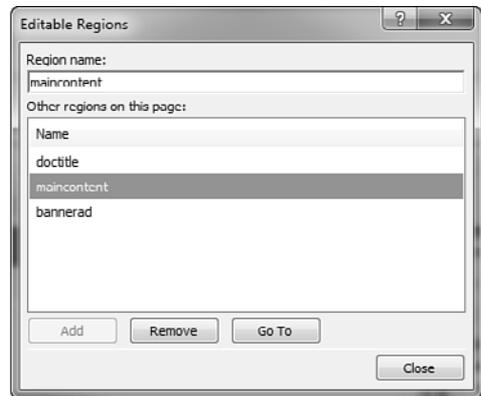
**caution**

Before you detach from a Dynamic Web Template, be advised that if you decide to reattach the page to the Dynamic Web Template later, Expression Web will likely duplicate content on the page. Content on the page that originally was part of the Dynamic Web Template will be moved into an editable region, and the content from the Dynamic Web Template will be re-added to the page.

Managing Editable Regions

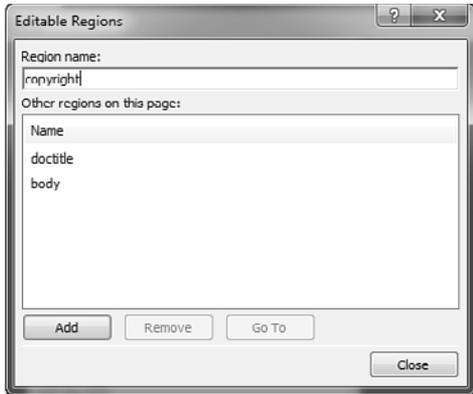
Editable regions can be added, renamed, or removed using the Editable Regions dialog shown in Figure 19.11.

Figure 19.11
Editable regions are easily edited using the Editable Regions dialog.



Adding a New Editable Region

To add a new editable region, place the insertion point where you want the new editable region and select Format, Dynamic Web Template, Manage Editable Regions to display the Editable Regions dialog. Enter a name for the new editable region and click Add to add it to the Dynamic Web Template, as shown in Figure 19.12.

**Figure 19.12**

Adding a new editable region is simple using the Editable Regions dialog. In this case, I'm adding a new editable region for a copyright statement.

When you save the Dynamic Web Template after adding a new editable region, Expression Web asks whether you want to update all the attached pages. When you do, the new editable region is added to all attached pages.

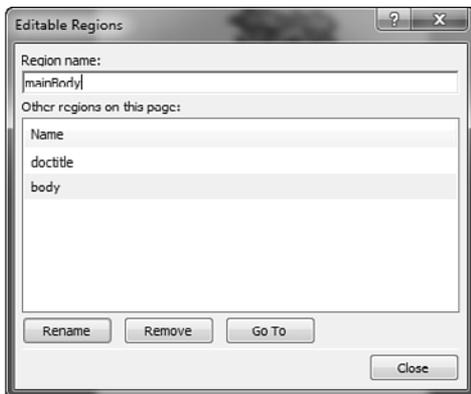
Renaming an Existing Editable Region

Existing editable regions can be renamed easily by clicking inside the editable region and then selecting Format, Dynamic Web Template, Manage Editable Regions to display the Editable Regions dialog. Enter the new name for the editable region, and then click Rename, as shown in Figure 19.13.



tip

Contrary to the way you would think it would work, to rename an editable region, you must first click inside the editable region and then select Manage Editable Regions. Otherwise, Expression Web tries to add a new editable region using the new name.

**Figure 19.13**

Rename an editable region by selecting the region, entering the new name, and clicking Rename.



No Rename Button in Editable Regions Dialog

If you've selected an editable region and then selected Manage Editable Regions to rename it, but a rename button doesn't appear when the Editable Regions dialog is displayed, it's not a bug. To rename an editable region, click anywhere inside the editable region first. If you select the entire editable region first, Expression Web does not allow you to rename it.

You can rename another editable region by selecting it from the list of other editable regions, entering a new name, and clicking Rename.

Resolving Mismatched Editable Regions

When you rename an editable region and update attached pages, Expression Web displays the Match Editable Regions dialog so that editable regions can be matched (see Figure 19.14). In most cases, Expression Web correctly maps editable regions to the corresponding content, but if it doesn't, you'll need to modify the mapping. In Figure 19.15, the copyright editable region has been renamed `copyrightStatement`. Notice that Expression Web is assuming that the copyright editable region that currently exists on my pages should be mapped to the `contentBlock` editable region. In this case, the mapping needs to be modified.

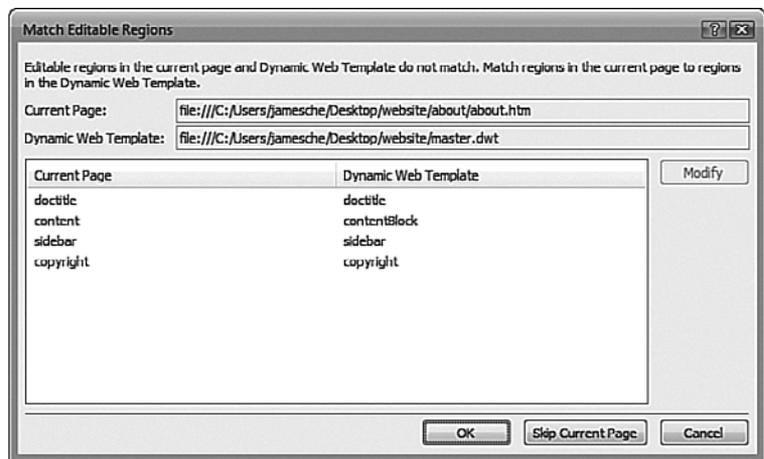


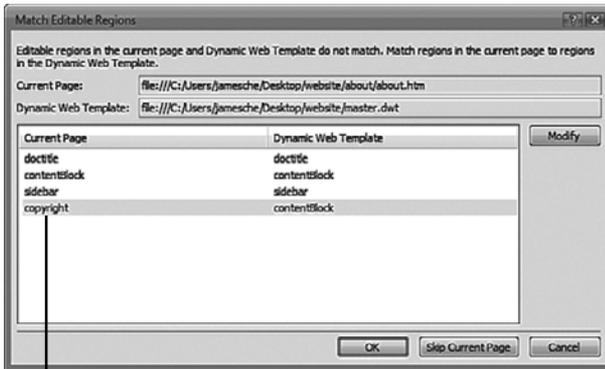
tip

If you detach a page from a Dynamic Web Template and then change your mind, you can easily reattach the Dynamic Web Template. Simply press `Ctrl+Z` or select Edit, Undo Detach from Dynamic Web Template.

Figure 19.14

The Match Editable Regions dialog lets you easily map editable regions on existing pages to newly renamed editable regions.





Incorrectly mapped editable region

Figure 19.15

In this case, the recommended mapping is wrong. The existing copyright editable region needs to be mapped to the newly renamed copyrightStatement editable region.

To remap an editable region, select the editable region from the Match Editable Regions dialog and click the Modify button. Select the correct editable region from the New Region drop-down, as shown in Figure 19.16, and click OK to remap the editable region.



Figure 19.16

Select the correct editable region from the Choose Editable Region for Content dialog to remap it.

If you want to remap the editable region on the current page later, click the Skip Current Page button on the Match Editable Regions dialog.

Detaching a Dynamic Web Template

Sometimes having a page attached to a Dynamic Web Template is too restrictive. In such cases, you can detach the page from the Dynamic Web Template.

You detach a page from a Dynamic Web Template by opening the page and then selecting Format, Dynamic Web Template, Detach from Dynamic Web Template. When you detach a page from a Dynamic Web Template, the connection between the page and the Dynamic Web Template is severed, but no content is removed from the page. Any changes made to the Dynamic Web Template from that point on do not affect the page.

Under the Hood

Dynamic Web Templates can make managing your site easier. Expression Web makes creating editable regions for flexible page layouts easy, but some web designers prefer to make changes in Code View for maximum control. Even if you spend the majority of your time in Design View, it's still helpful to know how Dynamic Web Templates work under the hood.

Perhaps the easiest way to understand how Dynamic Web Templates work is to look at Code View after creating a new Dynamic Web Template. The code in Listing 19.1 is from a newly created Dynamic Web Template.

Listing 19.1 Dynamic Web Template Code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html dir="ltr" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <!-- #BeginEditable "doctype" -->
  <title>Untitled 1</title>
  <!-- #EndEditable -->
</head>
<body>
  <!-- #BeginEditable "body" -->
  <div>
  </div>
  <!-- #EndEditable -->
</body>
</html>
```

By default, every Dynamic Web Template contains two editable regions called `doctype` and `body`, respectively. The `doctype` editable region surrounds the `<title>` tag and the `body` editable region is inside the `<body>` tag. Expression Web determines the location of editable regions using the codes `#BeginEditable` and `#EndEditable`.

In Listing 19.1, the `doctype` editable region code is as follows:

```
<!-- #BeginEditable "doctype" -->
  <title>Untitled 1</title>
<!-- #EndEditable -->
```

Any code between the `#BeginEditable` and `#EndEditable` codes is inside the editable region and can be modified. In this case, the code inside the `doctype` editable region is the HTML `<title>` tag.



tip

Because the Dynamic Web Template codes that Expression Web uses are not recognizable by web browsers, they are enclosed in HTML comments so web browsers won't generate errors.

You can use this knowledge of how Dynamic Web Templates work to tweak the position of editable regions in code. By moving the `#BeginEditable` and `#EndEditable` codes so that they enclose content you want to be editable, you can precisely control the page elements that appear inside an editable region.

For example, consider the following code:

```
<!-- #BeginEditable "mainTable" -->
<div>
  <table style="width: 100%">
    <tr>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
    </tr>

    <tr>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
    </tr>
  </table>
</div>
<!-- #EndEditable -->
```

This code defines an editable region called `mainTable`, and the entire table is included within an editable region. Suppose you want to make the bottom row of the table an editable region but leave the top row protected. There's no way to make that change in Design View, but you can do it easily by making a small code modification. The following shows the new code:

```
<div>
  <table style="width: 100%">
    <tr>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
    </tr>
    <!-- #BeginEditable "mainTable" -->
    <tr>
      <td>&nbsp;</td>
      <td>&nbsp;</td>
    </tr>
    <!-- #EndEditable -->
  </table>
</div>
```

Notice that the `#BeginEditable` and `#EndEditable` codes are now enclosing only the second table row. Only content within that second row is inside the editable region. The rest of the table is protected content.

USING INTERACTIVE BUTTONS

Overview of Interactive Buttons

Interactive buttons are a convenient way to add graphical navigation links that provide interactivity and are easily editable as your site changes over time (see Figure 20.1).

Interactive buttons use a series of images and some JavaScript to swap out images as you interact with a button by hovering over it or clicking it. Because interactive buttons use standardized Dynamic HTML code, they work on all modern browsers, assuming JavaScript is enabled.



Figure 20.1 Interactive buttons aren't appropriate for site-wide navigation, but they are a nice solution if you're looking for attractive navigation buttons.

Inserting and Configuring Interactive Buttons

One of the first things you're likely to notice about interactive buttons is that there are a lot of choices in button style. As shown in Figure 20.2, you have a large list from which to choose, including 31 categories and well over 100 styles.

To insert an interactive button, select Insert, Interactive Button. The Interactive Button dialog consists of three tabs: the Button tab, the Font tab, and the Image tab.

The Button Tab

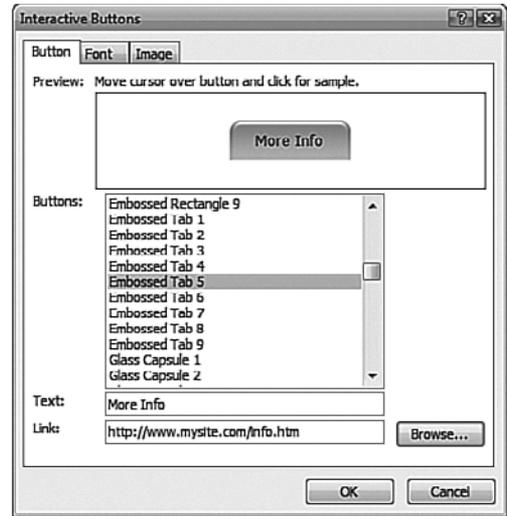
The Button tab, shown in Figure 20.2, allows you to select a button style and configure basic properties of the button. You can specify the text that appears on the button as well as the URL to which the button should link. If you enter a lot of text for the button, the text can exceed the width of the button. In such cases, you can either change the font size using the Font tab or modify the size of the button itself using the Image tab.

note

Because of the many security threats that exist on the Internet, some people disable scripting in the browser in an attempt to add a degree of protection from rogue scripts. Although doing so isn't advised and provides minimal protection at best, you must still consider such possibilities when designing your site.

Interactive buttons will not swap images if scripting is disabled, but buttons will still appear and the links associated with buttons will still work.

Figure 20.2
Expression Web gives you many styles of interactive buttons from which to choose.



To get a good idea of how your button will appear on the page, Expression Web provides a Preview window. Hover over a button in the Preview window to see the interaction. By clicking a button, you can see how the button will change its appearance when clicked.

The Font Tab

The Font tab, shown in Figure 20.3, provides configuration settings for the font that appears on an interactive button. In addition to the font, you can choose font style (default, italic, bold, or both bold and italic) and font size.

Font colors can also be configured on the Font tab, as shown in Figure 20.4. You can configure the colors for three button states:

- **Original state**—The state of the button when the mouse is not hovering on it and it is not being clicked
- **Hovered state**—The state of the button when the mouse pointer is hovered over it
- **Pressed state**—The state of the button when the button is clicked



tip

The text you enter for an interactive button is actually embedded into the images that will make up the button when you save your page. If you want to edit the text, you need to do it using the Interactive Buttons dialog in Expression Web.



tip

The Preview window shows how a button changes as you interact with it with your mouse. However, any links you have configured will not be active in the Preview window.



tip

If the font size you need is not listed, simply enter the size in the Size text box.

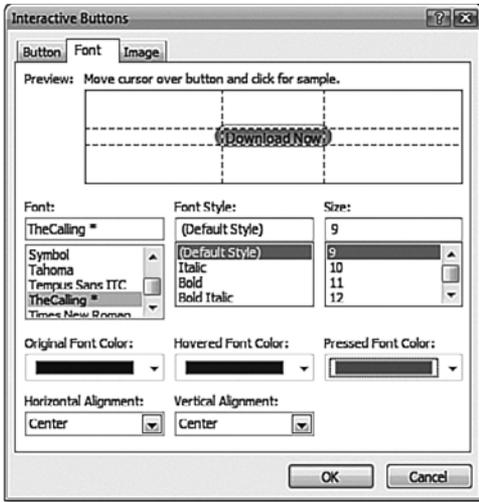


Figure 20.3

The Font tab provides all the tools you need to configure the font for an interactive button.

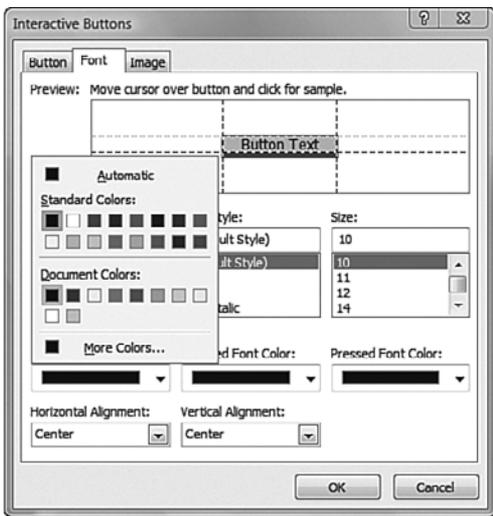


Figure 20.4

The font color can be configured for the different states of an interactive button.

To configure the alignment of a button's text, select the desired alignment (left, center, or right) from the alignment drop-downs. You can configure both the horizontal and vertical alignment of the text.

**tip**

If your page already has a color scheme defined, the Interactive Buttons dialog will pick up your color scheme in the Document Colors section shown in Figure 20.4. This lets you easily keep within your color scheme.

The Image Tab

The Image tab, shown in Figure 20.5, provides settings for controlling the size and format of the images that make up an interactive button.

To control the width and height of an image, enter a pixel value in the Width and Height text boxes. Arrow buttons are provided for precise size adjustments. When the Maintain Proportions check box is checked, a change in height will cause a proportionate change in width, and vice versa. Removing the check from the Maintain Proportions check box allows you to make uniquely shaped buttons, as shown in Figure 20.6.

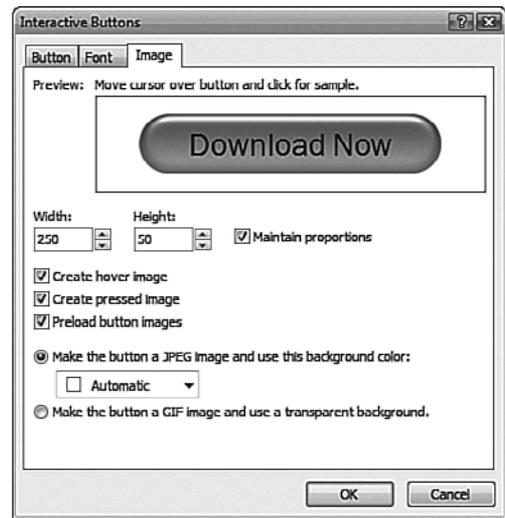


caution

The Preview window on the Font tab provides alignment lines to aid in font formatting. To ensure that your text appears correctly, make sure it doesn't extend outside the font alignment lines.

Figure 20.5

The Image tab is where you configure the images that make up an interactive button.

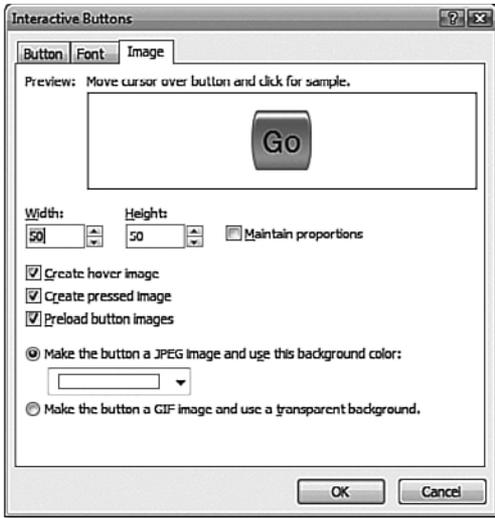


By default, Expression Web creates images to be displayed when an interactive button is hovered over and when it is clicked. The Create Hover Image check box allows you to choose whether Expression Web will create the hover image. If the check box is unchecked, the interactive button will not change when it is hovered over. Similarly, the Create Pressed Image check box controls whether an image will be created for the pressed state of an interactive button.



Interactive Button Images Broken

If you see a broken image in your browser when browsing to a page with interactive buttons, chances are you removed one or more of the image files the interactive button requires. Fortunately, fixing this problem is simple. Open the page in Expression Web and double-click the interactive button. Then simply click OK without making any changes. When you save the page, Expression Web resaves the images for the interactive button and your problem will be fixed.

**Figure 20.6**

The Embossed Capsule button style can be made more unique by specifying an equal height and width.

When the Preload Button Images check box is checked, Expression Web adds a script to the page to preload the images that make up the interactive button. By preloading images, users of your site will not have to wait for the images to load when they hover over or click a button. When images are not preloaded, users will experience a delay before the new image is displayed. Therefore, it is recommended that you leave this check box checked.

The Image tab contains options for the types of images Expression Web uses to build your interactive button. You can choose between JPEG and GIF images.

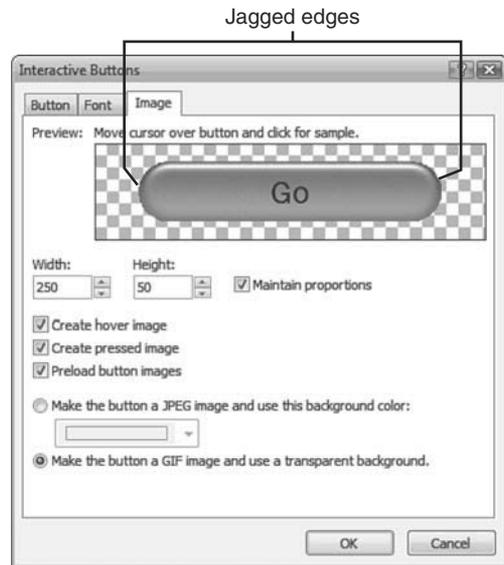
➔ *For information on choosing between different image formats, see Chapter 9, “Using Graphics and Multimedia.”*

When the Make the Button a JPEG Image and Use This Background Color radio button is selected, Expression Web saves your images in the JPEG format and uses the background color you specify. Selecting the Make the Button a GIF Image and Use a Transparent Background radio button causes Expression Web to save your images in the GIF format and use a transparent background. However, as shown in Figure 20.7, the GIF option can cause jagged edges on buttons with rounded corners, so use it carefully.

**tip**

The background color on a JPEG image is visible only if you select a button type that does not have right-angled edges.

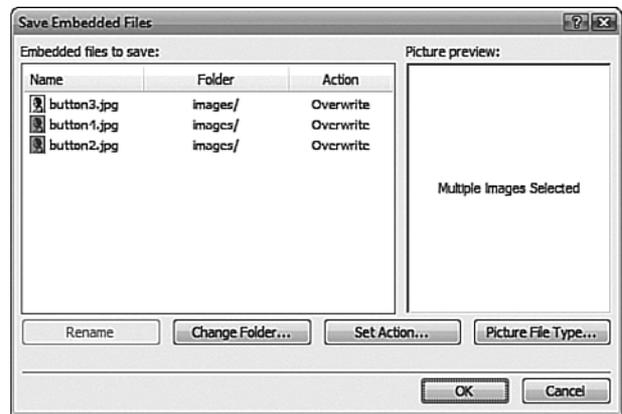
Figure 20.7
The GIF option can create some jagged edges, so be careful when selecting it.



Saving an Interactive Button

Because Expression Web builds interactive buttons using image files, when you save a page on which an interactive button has been inserted, you are prompted to save the images, as shown in Figure 20.8.

Figure 20.8
Expression Web saves images that make up an interactive button when you save the page.



➔ For more information on the Save Embedded Files dialog, see Chapter 9, “Using Graphics and Multimedia.”

It might seem as though there is a connection between image files and an interactive button, but there isn't. Therefore, if you remove an interactive button from a page, you want to manually delete the image files from your site so they don't take up unnecessary disk space.

Editing an Interactive Button

Editing an interactive button is easily accomplished by double-clicking the interactive button in Design View. The Interactive Button dialog is displayed and you can edit all the properties of the button.

When you edit an interactive button, Expression Web always prompts you to resave the images that make up the button even if the change you have made does not affect the images. In fact, if you open the Interactive Button dialog and click OK without making any changes, Expression Web still prompts you to resave the images. In all cases, allowing Expression Web to resave and overwrite the existing images will cause no problems.



tip

To move the images created by an interactive button after you've added the button, simply drag and drop the image files into the desired folder from within the folder list. Expression Web automatically updates the links so that the interactive button still works correctly.



tip

You can also edit an interactive button by right-clicking the button and selecting Button Properties from the menu that appears.



caution

If you choose a different image format when editing an interactive button, Expression Web will not delete the images you previously saved in the original format.



Multiple Image Copies Saved for an Interactive Button

By default, Expression Web saves interactive button images in the root folder of your site. If you change that and save them in a subfolder, the interactive button still works fine. However, if you then take some action that resets Expression Web's folder setting (such as flip-flopping the image format), Expression Web loses track of the images saved in a different folder and resaves new images in the root folder of the site.

You can also cause multiple copies to be created if you change from JPEG to GIF or GIF to JPEG without flip-flopping. Expression Web doesn't delete the original image files when you switch image formats, so you'll want to do it manually.

If you're not sure which image files to remove, remove them all, and then double-click the interactive button. Next, click OK in the Interactive Button dialog and save the page. Expression Web automatically saves all the images for the interactive button and you'll be fine.

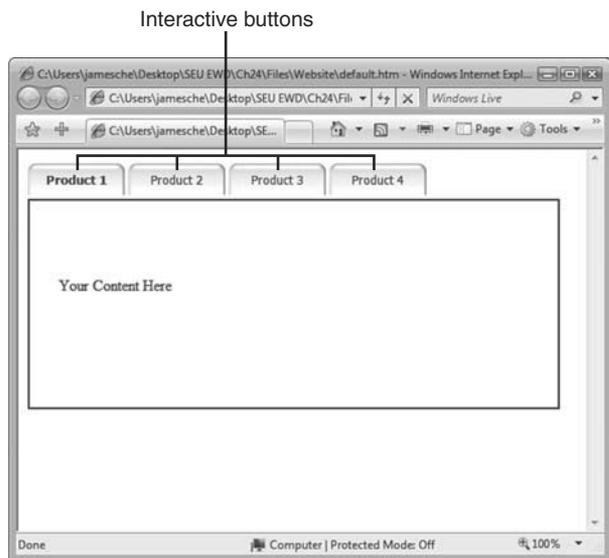
Practical Uses for Interactive Buttons

I began this chapter by saying that interactive buttons don't fit the bill as a site-wide navigation tool. I stand by that statement, but that doesn't mean there aren't some practical reasons for using interactive buttons.

Many of the examples in this chapter showed an interactive button being used as a Download Now button. Everyone who uses the Web downloads software from time to time, and there's nothing more frustrating than searching a page for a link to download a file you need. An interactive button is the perfect choice because it stands out and is easily created.

Another great use of interactive buttons is for the creation of tabbed navigation within a page, as shown in Figure 20.9. This type of interface is easily created using tables and interactive buttons.

Figure 20.9
Interactive buttons are a great choice for creating tabbed “dialogs” on pages.



➡ For more information on using Tables, see Chapter 5, “Using Tables.”

In short, anytime you need a few buttons on your site, interactive buttons are a great choice. Be creative with interactive buttons, but don't burden yourself by using them for site-wide navigation. If you do, you'll find yourself struggling to manage the images that are created and your navigation won't look as professional as it should. A much better choice for this task is a CSS menu system such as the ones you can create with the Ajatix Pure CSS Menu add-in for Expression Web. You can find Ajatix's add-in at <http://www.ajatix.com/pure-css-menu/expression-web-add-in.html>.

This page intentionally left blank

USING BEHAVIORS

Understanding and Working with Behaviors

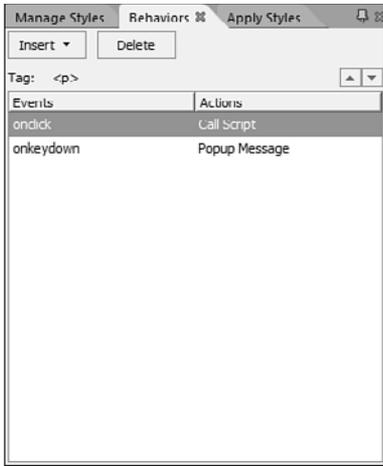
Behaviors make it easy for web designers to add interactive and dynamic features to a site. You can insert behaviors that range from simple, such as calling a client-side script, to complex, such as configuring a DHTML menu for your site.

Using the Behaviors Panel

Behaviors are inserted from the Behaviors panel, shown in Figure 21.1. The Behaviors panel is the perfect place for behaviors because it allows you to work with them without the interface getting in the way. That's important because behaviors cannot be inserted and forgotten. After you insert a behavior, there's plenty of tweaking you can do, and not having to dismiss a dialog between each change makes working with behaviors more efficient.

To access the Behaviors panel, select Panels, Behaviors. Click the Insert button and you'll see a collection of 13 behaviors you can insert into your page. Whether a specific behavior is available for insertion depends on what is selected on the page when you click the Insert button.

In some situations, you might need to insert more than one behavior for a single element. For example, when you are implementing rollover buttons, you need a Swap Image behavior and a Swap Image Restore behavior for the image. Behaviors are processed from top to bottom. When you insert more than one behavior on a single HTML element, depending on the type of behavior, you can change the order of precedence using the up and down arrow buttons in the Behaviors panel. Select a behavior and click the up arrow to move the behavior up in the list and the down arrow to move it down in the list.

**Figure 21.1**

The Behaviors panel lets you conveniently insert and work with behaviors.

How Behaviors Work

Behaviors are implemented using *events* and *actions*. An event is a message that is intercepted by the web browser when a certain action takes place. For example, when you click a hyperlink, an `onclick` event is sent to the browser. An action is what the browser does in response to the event. Behaviors allow you to add interactive scripts to your pages to take advantage of events.

The number and type of events available depend on the element you select before inserting a behavior. When you insert a behavior, Expression Web adds attributes to the closest HTML element to the left of the insertion point. (The Tag label in the Behaviors panel indicates the element to which the behavior will be applied.)

Adding Behaviors Within a Paragraph

Behaviors are easily applied to hyperlinks, images, and other page elements because these elements are already associated with an HTML tag to which the behavior can be applied. However, when you want to define a behavior for one or more words within a paragraph, it becomes a bit more complicated. If you simply select the words and apply a behavior, you will find that the behavior is applied to the entire paragraph and not just the selected words.

To apply a behavior to one or more words within a paragraph, you can enclose those elements within a `span` tag. The easiest way to do this is to select the word or words you want to use,

note

The events available in the Behaviors panel are documented on Microsoft's MSDN site. To access the documentation, browse to <http://msdn2.microsoft.com/en-us/library/m5533051.aspx>.

caution

After a behavior has been applied to an element, there is no visual indicator that it has been applied. Because of this, you can easily inadvertently apply a behavior to an HTML element only to find that another behavior has been applied to that element's parent. In these cases, the parent's behavior will take precedence.

switch to Split View, and enclose those words in a span tag, including a unique ID. For example, to use the words “click here” in a paragraph as the link for a behavior, change the code from this:

```
<p>For more information, click here.</p>
```

to this:

```
<p>For more information, <span id="mylink">click here</span>.</p>
```

Switch back to Design View. You can now click the span Quick Tag Selector to select the words “click here” before inserting your behavior. Your behavior will then be applied to those words. Without using this method, the behavior would affect the entire paragraph.

➔ *For more information on the Quick Tag Selector, see Chapter 8, “Using the Quick Tag Tools.”*

Expression Web Behaviors

Expression Web's behaviors offer a way to add interactivity and dynamic elements to your pages without having to know programming. Behaviors should work in all common browsers without problems because Microsoft uses standard DHTML and scripting techniques to ensure maximum compatibility.

The Call Script Behavior

The Call Script behavior runs a line of script when the event you specify is raised. You will likely want to write some script first and then call it using this behavior, but you don't have to take that approach. If you have only one line of script to run, you can enter that line in the Call Script dialog and Expression Web will run it when the designated event occurs.

Suppose you've written a script called `showInfo` that you want to run when an image is clicked on your page. To do this with the Call Script behavior, perform these steps:

1. Select the image.
2. If the Behaviors panel is not visible, select Panels, Behaviors.
3. Click the Insert button and select Call Script.
4. Type `showInfo()`; in the Call Script dialog, as shown in Figure 21.2.
5. Click OK.



tip

Behaviors are implemented in Expression Web using a combination of a JavaScript source file and an HTML file. If you have the coding knowledge necessary, you can edit these files to enhance a behavior to suit your specific needs. These files are located by default in the Microsoft Expression\ Web 4\en\Behaviors\ACTIONS folder inside the Program Files folder.



caution

Keep in mind that there is no way for Microsoft to test compatibility for every possible configuration, so be sure you test your pages before publishing your site.



note

The event defaults to `onclick`, but you can choose another event if necessary by selecting the event from the drop-down in the Behaviors panel.

➔ *For more information on creating and using scripts in your pages, see Chapter 22, “Client Scripting.”*

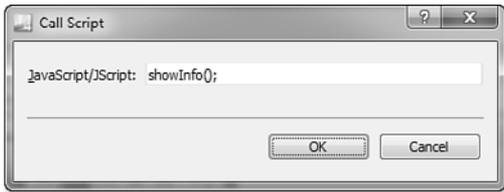


Figure 21.2
The Call Script behavior enables you to easily call a script when a specified event is triggered.

The Change Property Behavior

The Change Property behavior, shown in Figure 21.3, enables you to change any property on any HTML element on your page. (A property is a style applied to a particular page element.) The Change Property dialog allows you to configure some default properties such as font styles, borders, visibility, and so on, but you can also change other properties that aren't listed by clicking the Add button in the Change Property dialog.

➔ *For more information on using styles, see Chapter 18, “Managing CSS Styles.”*

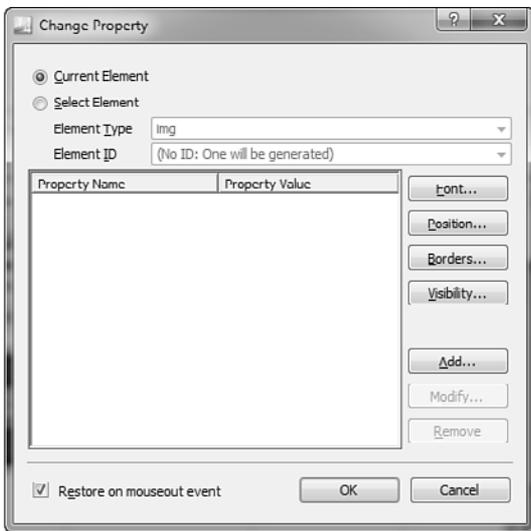


Figure 21.3
The Change Property behavior is useful for adding interactivity to your page.

Using the Change Property behavior, it is easy to add professional-quality interactivity to your site with minimal effort. For example, suppose you have a series of graphics in a page, and each graphic links to a page of your site. When a user of your site hovers the mouse over each graphic, you want a text description of the link to appear on the page, and when the mouse leaves the graphic, you

want the text description to disappear. The Change Property behavior allows you to easily implement this functionality without writing a single line of code.

To create an example of interactivity using the Change Property behavior, perform these steps:

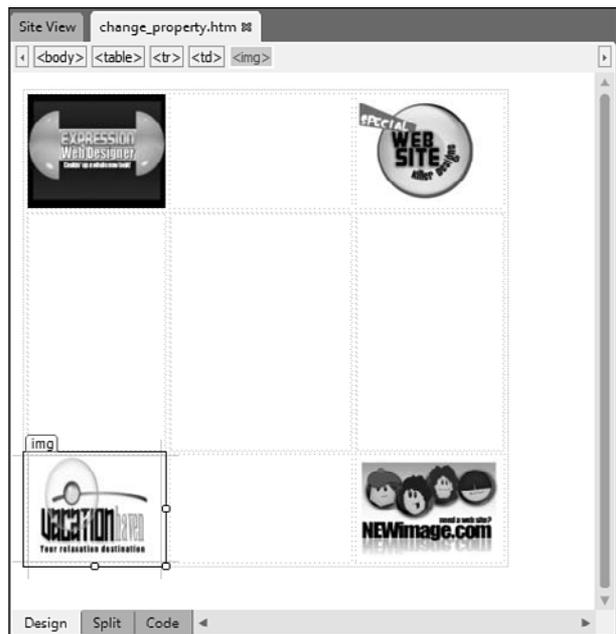
1. Insert a table 400 pixels wide that contains three columns and three rows.

➔ *For more information on inserting and configuring tables, see Chapter 5, “Using Tables.”*

2. Add some graphics of your choice to each corner cell in the table. Feel free to use clip art if you have no graphics handy.
3. Resize the center cell so it is 150 pixels wide and 200 pixels high. (See Figure 21.4 for an example of what the table should look like at this point.)

Figure 21.4

The table has four graphics and space in the middle for layers that will contain text.



4. Insert a layer. Position and size it so it fits within the center cell. Name the layer layer1.

➔ *For more information on using layers, see Chapter 23, “Using Layers.”*

5. Insert three more layers and position them directly on top of layer1. The easiest way to accomplish this is to copy layer1 and then paste the other layers. Name the other layers layer2, layer3, and layer4, respectively.

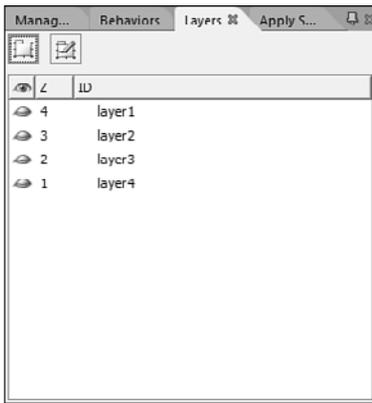
caution

Be careful not to nest the layers underneath each other. If a new layer appears nested under an existing layer, drag it inside the Layers panel so it is at the same level as the other layers.

6. Select layer1 from the Layers panel and enter some text of your choice.
7. Change the visibility of layer1 to hidden by clicking the eye icon so the eye is closed.
8. Select layer2 and enter some text of your choice.
9. Change the visibility of layer2 to hidden.
10. Repeat steps 8 and 9 for the other two layers. (See Figure 21.5 for the completed Layers panel.)

**tip**

The **Visibility** property defaults to **Inherit**, which means the element will be visible only if the element in which it is contained is visible.

**Figure 21.5**

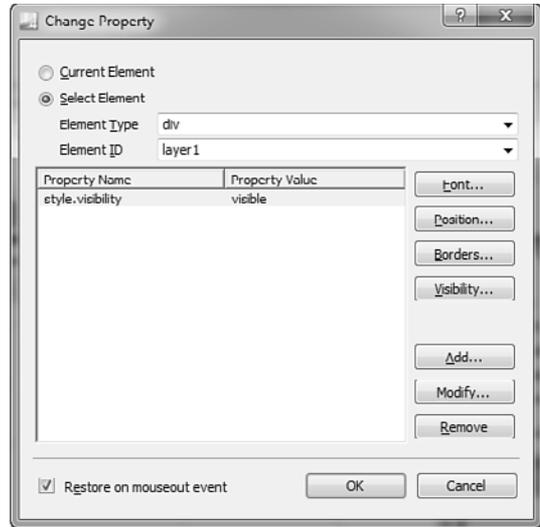
The four layers for the rollover text are all invisible, which is signified by the closed eye icon.

11. Select the upper-left image. From the Behaviors panel, select Insert, Change Property.
12. In the Change Property dialog, click the Select Element radio button.
13. Choose div from the Element Type drop-down, and then select layer1 from the Element ID drop-down.
14. Click the Visibility button and click the Visible radio button. Click OK.
15. Check the Restore on Mouseout Event check box.
16. Click OK.
17. In the Behaviors panel, change the onclick event to onmouseover.

Complete steps 11–17 for the other images and add a Change Property behavior for layer2, layer3, and layer4, just as we did for layer1. Save the page and preview it in your browser. You have just created a professional-quality effect with a few clicks of the mouse. Figure 21.6 shows the completed Change Property dialog box for one of the layers.

Figure 21.6

The completed Change Property dialog box shows the property change for layer1.



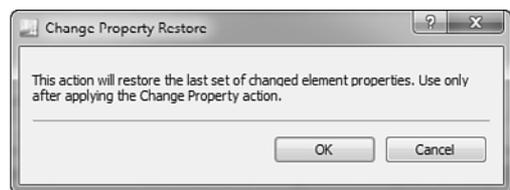
The Change Property Restore Behavior

The Change Property Restore behavior is meant to be used in combination with the Change Property behavior. It restores the most recent properties changed by the Change Property behavior to their previous values. (When you selected the Restore on Mouseout Event check box in step 17 of the previous section, “The Change Property Behavior,” Expression Web automatically inserted a Change Property Restore behavior to restore the property.)

No dialog exists for this behavior. As seen in Figure 21.7, when you insert the Change Property Restore behavior, Expression Web tells you what it will do and then adds it to the Behaviors panel. You can easily determine which properties are being restored by double-clicking the Change Property behavior listed directly under the newly inserted Change Property Restore behavior.

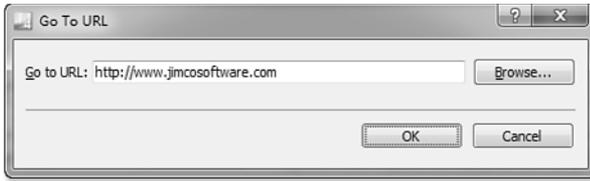
Figure 21.7

The Change Property Restore behavior requires no user input.



The Go To URL Behavior

The Go To URL behavior, shown in Figure 21.8, simply redirects users to a specified URL when the event you choose occurs.

**Figure 21.8**

The Go To URL behavior does just what its name implies.

The Jump Menu Behavior

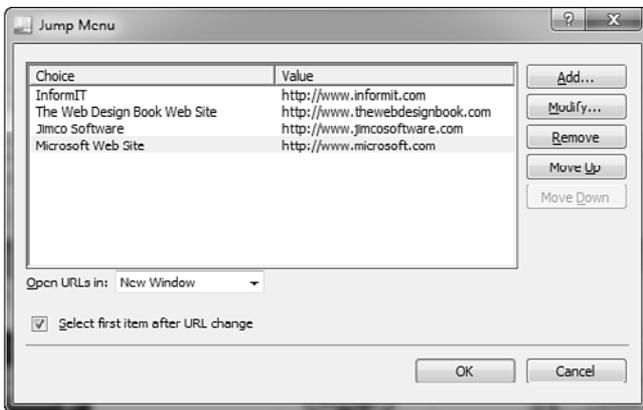
The Jump Menu behavior allows you to easily add a drop-down box to a page. When an item is selected in the drop-down, the user's browser goes to the URL specified for that item.

To create a jump menu, select Insert, Jump Menu from the Behaviors panel to activate the Jump Menu dialog (see Figure 21.9). You don't need to select anything first. Expression Web inserts a drop-down list for the jump menu at the insertion point's location.

Click the Add button. Enter the text that you want to appear in the drop-down list in the Choice field and the URL to which you want that item to link in the Value field. Click OK to add the item to the jump menu.

After you have added one or more items to your jump menu (more than one item is recommended), you can configure that item to open in either the default target for the page or in a new window.

By checking the Select First Item After URL Change box, you can force the jump menu to return to the first item in the list when clicking the Back button in your browser after jumping to a URL.

**Figure 21.9**

Items are added to a jump menu by specifying the text to appear and the URL to which the item should link.

The Jump Menu Go Behavior

The Jump Menu Go behavior is designed to be used with the Jump Menu behavior. When you don't want your Jump Menu behavior to jump as soon as a new item is selected in the drop-down list, you can use the Jump Menu Go behavior as a trigger for the menu.

To use the Jump Menu Go behavior, insert a Jump Menu behavior and then insert or select an element (such as an image that says “Go!”) to use for your trigger. Select Insert, Jump Menu Go from the Behaviors panel and select the jump menu from the drop-down, as shown in Figure 21.10.

Figure 21.10

Using the Jump Menu Go behavior enables you to create a trigger for your jump menu.



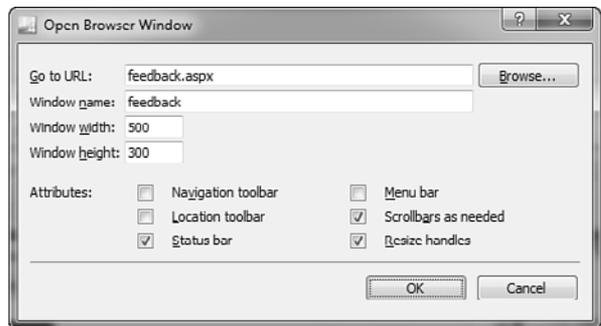
After you've done that, you must stop the jump menu from jumping as soon as you select an item. The Jump Menu event defaults to onchange, but if you leave the event set to onchange, you won't have the opportunity to use the Jump Menu Go component because the menu will jump as soon as you select an item. To change the jump menu so it doesn't jump when you select a different item, select the jump menu's drop-down list, and then select the Jump Menu behavior in the Behaviors panel. Click the Delete button in the panel to delete the onchange event from the panel. (Don't press the Delete key on your keyboard.) This removes the entire entry for the jump menu, but don't worry. After doing this, your Jump Menu Go behavior will be the trigger for the jump menu.

The Open Browser Window Behavior

The Open Browser Window behavior, shown in Figure 21.11, allows you to configure a hyperlink to open in a new browser window.

Figure 21.11

Easily create pop-up windows with specific configurations using the Open Browser Window behavior.



You can control the following attributes for the new window:

- **Window Name**—A unique name for the new browser window so you can refer to it with other behaviors or your own scripts

- **Window Width**—The width of the new window in pixels
- **Window Height**—The height of the new window in pixels
- **Navigation Toolbar**—The toolbar that contains the Forward button, Back button, Home button, and so on
- **Location Toolbar**—The toolbar that contains the address bar
- **Status Bar**—The bar at the bottom of the new window
- **Menu Bar**—The menu bar at the top of the new window
- **Scrollbars as Needed**—Turns on scrollbars if the content of the new window is larger than the window itself
- **Resize Handles**—Allows for resizing of the new window by dragging the edges

**tip**

Internet Explorer 6.0 and later versions do not allow for new windows to be opened without a status bar by default. Therefore, even if you leave the status bar check box unchecked, most users will still see a status bar displayed.

When adding the `Open Browser Window` behavior to text, it's most useful to use the `Change Property` and `Change Property Restore` behaviors to change the pointer to a hand when users hover over your link.

To implement this, add your `Open Browser Window` behavior first. Then add a `Change Property` behavior and add your own property by clicking the `Add` button. Use `style.cursor` for the Property Name and `hand` for the Property Value. Make sure you also check the `Restore on Mouseout` Event box so that Expression Web will add the `Change Property Restore` behavior.



Open Browser Window Pops Up Behind Existing Window

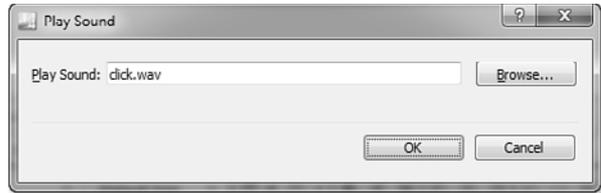
If you are having trouble with the browser window appearing behind the main window—you can see both, but the new browser window opens behind the main browser window—you can change the order of your behaviors by clicking the arrow buttons in the Behaviors panel. An `Open Browser Window` behavior should always appear at the bottom of the list because it's the last thing you want to do on a page. If you perform some other action on the page after the `Open Browser Window` behavior has created a new window, the main window will resume focus and the new browser window will be moved behind it.

The Play Sound Behavior

The `Play Sound` behavior, shown in Figure 21.12, enables you to play a sound when the selected event is triggered. The `Play Sound` behavior supports WAV files, MIDI files, RealAudio files, AIFF sound files, AU sound files, MPEG files (such as MP3), Windows Media files, and AAC files.

Figure 21.12

The **Play Sound** behavior enables you to play a sound when a particular event is triggered.

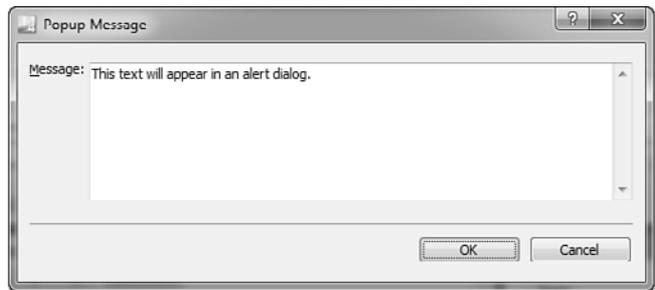


The Popup Message Behavior

The **Popup Message** behavior, shown in Figure 21.13, provides a dialog in which you can enter a message that should be displayed in an alert dialog in the browser. When the selected event is triggered, the text you enter appears in an alert dialog with an OK button.

Figure 21.13

The **Popup Message** behavior makes creating JavaScript pop-up messages simple.



The Preload Images Behavior

The **Preload Images** behavior, shown in Figure 21.14, allows you to preload images when a page initially loads. The most common use of preloading images is when swapping images on mouse roll-over buttons. In these situations, you want the image that displays when the user points to the button to load as soon as the page loads, even though it is not initially visible. If you don't preload such images, the user will experience a delay while the image loads the first time he or she points to the button.

To preload one or more images, insert the **Preload Images** behavior, click the **Browse** button, browse to your image, and then click **OK**. After you have selected your image, click the **Add** button to add it to the **Preload Images** list. You can add as many images as you want to the list.



caution

If you choose an event that doesn't require user interaction, your popup will likely be blocked by popup blockers.



caution

Be aware that preloading images doesn't make them free as far as bandwidth is concerned. You will still want to keep your image files as small as possible.

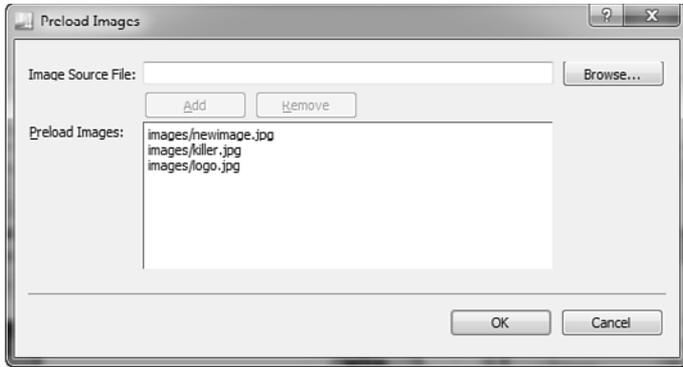


Figure 21.14
The Preload Images behavior makes preloading images a snap.

The Set Text Behavior

The Set Text behavior consists of four separate behaviors: Set Text of Frame, Set Text of Layer, Set Text of Status Bar, and Set Text of Text Field.

Set Text of Frame

The Set Text of Frame behavior sets the HTML of the frame you specify when the selected event is triggered. Suppose, for example, you have a frames page that consists of miscellaneous content on the left, main content on the right, and a small navigation frame on top of the main content frame. Using the Set Text of Frame behavior, you can easily implement a bread crumb effect in the navigation frame, as shown in Figure 21.15.

To insert the Set Text of Frame behavior, open a page containing a frame and select the HTML element containing the event with which you want to trigger the behavior. Select Insert, Set Text, Set Text of Frame from the Behaviors panel. Select the frame into which you want to enter HTML, enter the HTML code that you want to appear in the frame, and click OK.

If you set the background color for the frame page or inline frame and want it to be preserved, be sure you check the Preserve Background Color box before clicking OK, as shown in Figure 21.16. Otherwise, the background color reverts to the default for the browser you are using.

Set Text of Layer

The Set Text of Layer behavior enables you to specify the HTML to be rendered in the specified layer when the selected event is triggered. The bread crumb effect shown previously in Figure 21.15 can be implemented using the Set Text of Layer behavior by using a layer to hold your bread crumb navigation information instead of a frame.

note

When hovering over a hyperlink, most browsers display the URL for the hyperlink. For security reasons, you probably won't be able to use the Set Text of Status Bar behavior to show text in the status bar when hovering over a hyperlink.

➔ For more information on using layers in Expression Web, see Chapter 23, "Using Layers."

Figure 21.15
A bread crumb effect is simple to implement with the Set Text of Frame behavior.

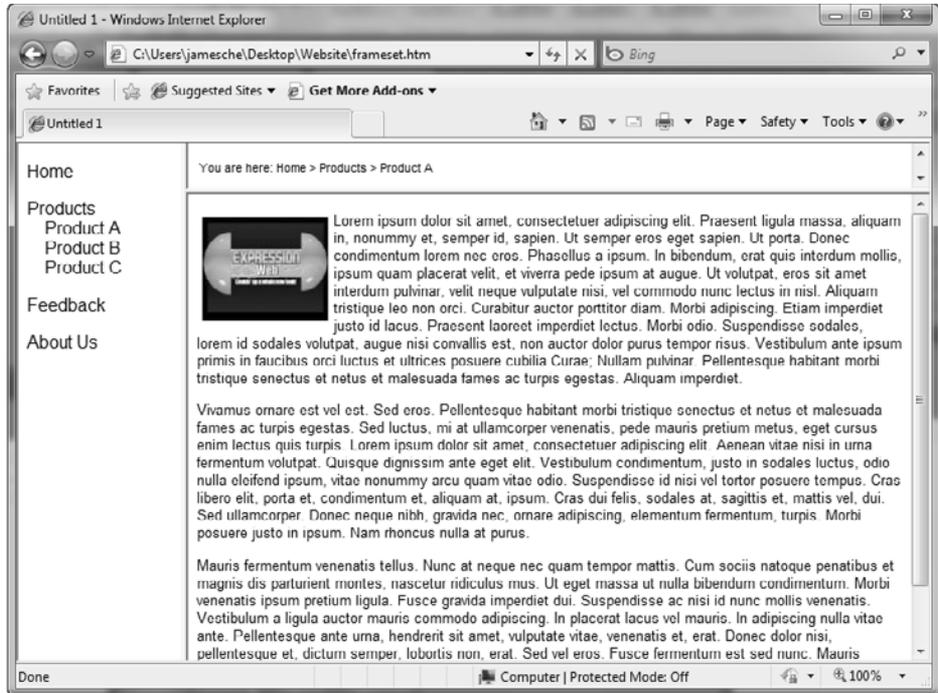
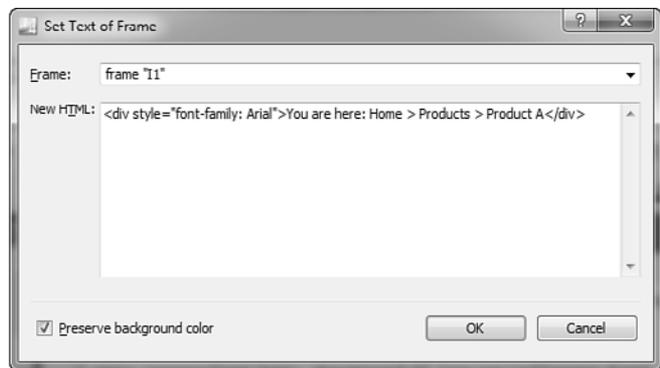
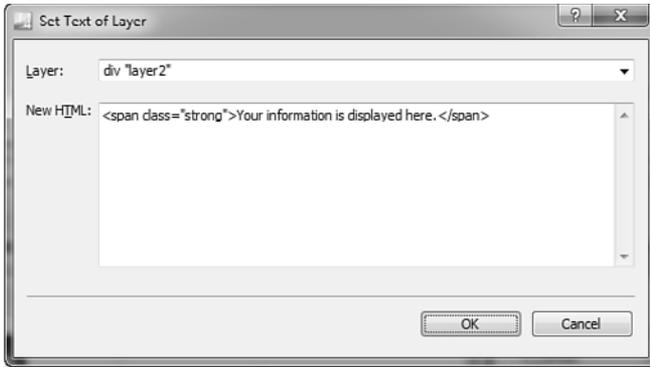


Figure 21.16
The Set Text of Frame behavior enables you to maintain the current background color after setting a frame's HTML.



To insert a Set Text of Layer behavior, shown in Figure 21.17, open a page containing a layer and select the HTML element containing the event with which you want to trigger the behavior. Select Insert, Set Text, Set Text of Layer, enter the HTML you want to have rendered in the layer, and click OK.

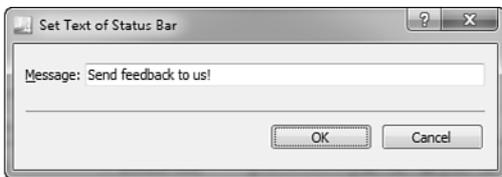
**Figure 21.17**

The Set Text of Layer behavior enables you to easily set the HTML to be used to render the content of a layer.

Set Text of Status Bar

The Set Text of Status Bar behavior allows you to easily change the text that appears in the Status Bar at the bottom of the browser window. This can be useful to display messages of interest to visitors of your site. For example, you can use this behavior to display a descriptive message when a user hovers over hyperlinks on your page.

To insert the Set Text of Status Bar behavior, select the HTML containing the event with which you want to trigger the behavior and then select Insert, Set Text, Set Text of Status Bar. Enter the message you want displayed in the Status Bar and then click OK, as shown in Figure 21.18.

**Figure 21.18**

The Set Text of Status Bar behavior lets you easily change the text displayed in the Status Bar—just enter your text and click OK.

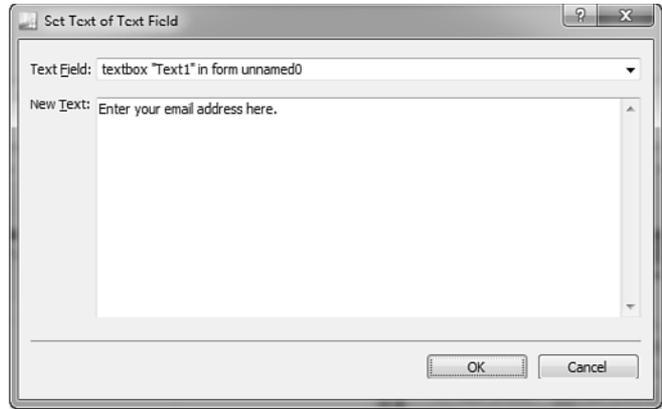
Set Text of Text Field

The Set Text of Text Field behavior, shown in Figure 21.19, sets the text that appears in a text box or text area field. This behavior can be used when you need to set the value for a form field based on a particular event.

To insert the Set Text of Text Field behavior, open a page that contains a text box or text area field and select Insert, Set Text, Set Text of Text Field. Enter the text you want to appear in the text field and click OK.

Figure 21.19

The **Set Text of Text Field** behavior sets the text of text boxes and text areas.



The Swap Image Behavior

The **Swap Image** behavior allows you to swap one image for another when a specified event occurs. The most common use of this behavior is to create rollover buttons. In fact, when you insert an interactive button, Expression Web uses the **Swap Image** behavior to implement the interactivity of the button.

➔ *For more information on interactive buttons, see Chapter 20, “Using Interactive Buttons.”*

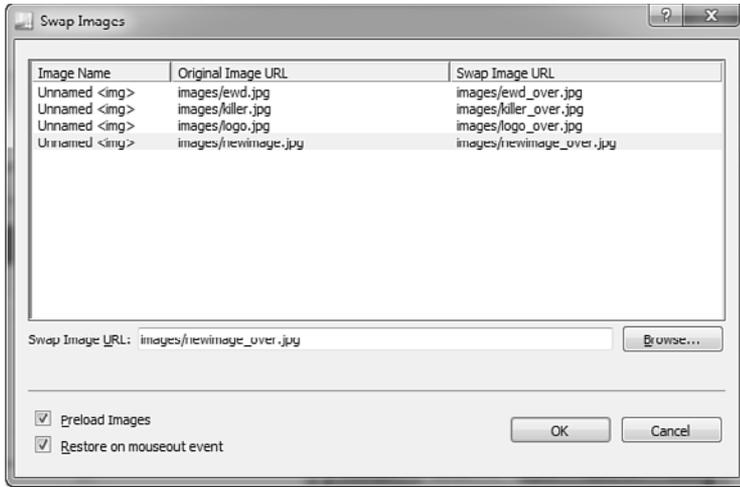
To use the **Swap Image** behavior, open a page containing one or more images, select the image you want to swap, and select **Insert, Swap Image** from the Behaviors panel. The **Swap Images** dialog contains a list of all images in the document (see Figure 21.20). The image you selected is highlighted, but you can configure a swap image for as many images as you want before clicking **OK**. Just keep in mind that you are configuring one event, so if you configure a swap image for `img1`, `img2`, and `img3` and attach it to the `onmouseover` event for `img1`, all three images will swap when you roll over `img1`.

It is best to use two images of the same size for your rollovers. The **Swap Image** behavior uses the image dimensions of the original image for the swap image. If the proportions are different or if Expression Web is forced to resize an image, you are going to get less than optimal results.

The **Swap Images** dialog also provides a check box labeled **Preload Images**. Checking this box automatically inserts a **Preload Images** behavior to preload all the swap images you configured. **Restore on Mouseout Event** automatically inserts a **Swap Image Restore** behavior, which we will talk about next.

The Swap Image Restore Behavior

The **Swap Image Restore** behavior works much like the **Change Property Restore** behavior. It restores the images that were swapped in the entry immediately below it in the Behaviors panel. It is used only after applying the **Swap Image** behavior.

**Figure 21.20**

The Swap Image behavior makes rollover button creation simple.

This behavior is automatically inserted if the Restore on Mouseout Event box is checked when configuring a swap image in the Swap Images dialog.

When to Use Behaviors

Generic scripts such as those produced by Expression Web behaviors are convenient and allow you to quickly add scripting capability to your pages. However, such convenience comes at a cost. Because the scripts used to implement behaviors have to work in all scenarios and with all pages, they are significantly more bloated than targeted, custom created scripts. Larger scripts mean larger pages, and hence, longer download times.

You are also locked into a specific implementation when using behaviors. In other words, by using prebuilt scripts, you lose a significant amount of flexibility and functionality. You can certainly modify the scripts used by behaviors, but you're much better off simply writing your own or paying a JavaScript developer to write some for you.

Because of these concerns, behaviors are best used for small tasks and not for things like a menu system. That said, small tasks such as those appropriate for behaviors, are also often easily solved with your own scripts, which can make the use of behaviors unnecessary. In the end, the decision is yours and should be based on how much time you want to allocate to filling a particular need.

**tip**

Another alternative is to search the Internet for script samples, but be sure to test such samples in all browsers.

CLIENT SCRIPTING

A History of Browser Scripting

Early sites consisted of many of the same page elements we use today: forms, images, hyperlinks, and static text. They also consisted of small applications called *applets* that ran inside the page and were written in a new programming language called Java. In 1995, Netscape Communications added Java support to its flagship product, Netscape Navigator. However, Netscape was painfully aware that many site developers were not Java developers, so it needed to find a way to allow non-Java developers to interact with Java applets on pages. It did so with the introduction of LiveScript, a technology that was renamed JavaScript by the time it made it into Netscape Navigator 2.0.

Web developers were quick to embrace JavaScript, but it was not used the way Netscape intended. It was mainly used to provide programmatic access to page content such as forms, images, and text—not to script Java applets. In fact, one of the common uses for JavaScript at the time is still a common use of script today—image swapping. Mouse rollovers were starting to appear all over the Internet.

At the same time, Microsoft released Internet Explorer 3.0, a major upgrade to its less-than-stellar web browser. With the release of Internet Explorer 3.0, Microsoft unveiled its own flavor of JavaScript, coined JScript. Microsoft also added its own scripting language (VBScript) and support for a new and emerging technology—Cascading Style Sheets (CSS). The inclusion of CSS support did not buy Microsoft much, however. Microsoft's JScript implementation did not include support for image swapping, and that made it useless in the eyes of web developers. Developers started writing scripts that checked browser versions, and if they detected Internet Explorer, they simply did not attempt to present any of the new dynamic content scripting provided.

Microsoft reacted with the release of Internet Explorer 3.02. Internet Explorer 3.02 added, among other things, support for image swapping. It also continued the divergence of scripting implementation among the major players in the web browser world. Web developers were still not able to write a script that would easily run on any browser. Instead, they had to write a version of their scripts for each browser and use the appropriate one depending on which browser was being used to access their page. The web developer community was aching for standards to be introduced to alleviate this problem. Sound familiar?

Netscape and Sun Microsystems, assisted by the European Computer Manufacturers Association (ECMA), standardized browser scripting with the release of ECMAScript in 1998. Ironically, Netscape was also in the process of releasing Netscape Navigator 4.0, a browser that would bring a completely proprietary document object model with it. Microsoft did the same with the release of Internet Explorer 4.0. These 4.0 series browsers introduced a robust new method of programming pages called *Dynamic HTML* (DHTML), a combination of HTML, CSS, and scripting. However, they also widened the gap in compatibility between the two browsers. ECMAScript was too late to stop the momentum.

➡ For more information on DHTML, see “The Document Object Model” section in this chapter, p. 363.

The majority of client-side scripting on the Internet today is used to script DHTML effects in pages. Although Internet Explorer still supports VBScript as a scripting language, the vast majority of web developers use JavaScript and not VBScript because it works in all major browsers. For that reason, this chapter sticks to a discussion of JavaScript.

Expression Web provides tools such as behaviors and interactive buttons that can generate JavaScript code for you. However, if you want to add more robust scripts to your page or modify the scripts Expression Web generates—or if the scripts Expression Web generates don't perform the task you need—knowing how to write JavaScript is a vital skill to have.

The purpose of this chapter is not to teach you how to be a JavaScript programmer. Instead, it is intended to give you a taste of what client-side scripting can be used for and the basics of how it is used. If you are interested in learning how to take maximum advantage of this powerful technology, you should pick up a book specifically with that purpose in mind.

- For a comprehensive discussion of JavaScript and how to use it to make your pages more interactive, read *Special Edition Using JavaScript* from Que Publishing.
- O'Reilly and Associates, Inc., has a site with great information concerning the history and evolution of JavaScript. You can access it at www.oreillynet.com/pub/a/javascript/2001/04/06/js_history.html.

 **note**

Around this same time, web developers began writing scripts to check for Netscape browsers by seeing whether the browser identified itself as “Mozilla,” an identifier Netscape Navigator used at the time. As browsers added functionality, they all began to identify themselves as “Mozilla Compatible” so scripts would work successfully. Even today, you will see all browsers identified as Mozilla if you review server logs for your site.

JavaScript Basics

JavaScript can be difficult for new programmers to learn. The language has several traits that lend to this difficulty, but perhaps the most frustrating for beginners is that JavaScript is case-sensitive. Almost all JavaScript programmers have gone through the pains of pulling their hair out because of a lowercase letter where there should have been an uppercase one, or vice versa. Debugging problems because of wrong case can be tough. Fortunately, Expression Web provides IntelliSense for JavaScript, making it much easier to avoid many of the problems new programmers encounter.

➔ *For more information on using IntelliSense, see Chapter 4, “Using Page Views.”*

➔ *For more information on debugging JavaScript code, see the “Debugging” section of this chapter, p. 377.*

Adding JavaScript to a Page

JavaScript can be added to a page either right within the page's HTML code or in an external script file with a .js file extension. To add JavaScript to a page, use the `<script>` tag. JavaScript code can consist of standalone code sections that are executed as soon as the browser encounters them and JavaScript functions that are executed only when they are explicitly called.

A *function* is a section of code that performs an action. To run code in a function, you simply use the function's name. The following code section defines a JavaScript function called `writeDateTime` in a page. The function is called when the page is loaded by specifying the function name in the `onload` event of the `<body>` tag:

```
<html>
<head>
  <script language="javascript" type="text/javascript">
    function writeDateTime() {
      document.write(Date());
      return true;
    }
  </script>
</head>

<body onload="writeDateTime();">
</body>
</html>
```

When this page is opened in a web browser, the current date and time are displayed, as shown in Figure 22.1. This happens because the `onload` attribute of the `<body>` tag calls the `writeDateTime` JavaScript function. We'll discuss the details of the function later in this chapter.

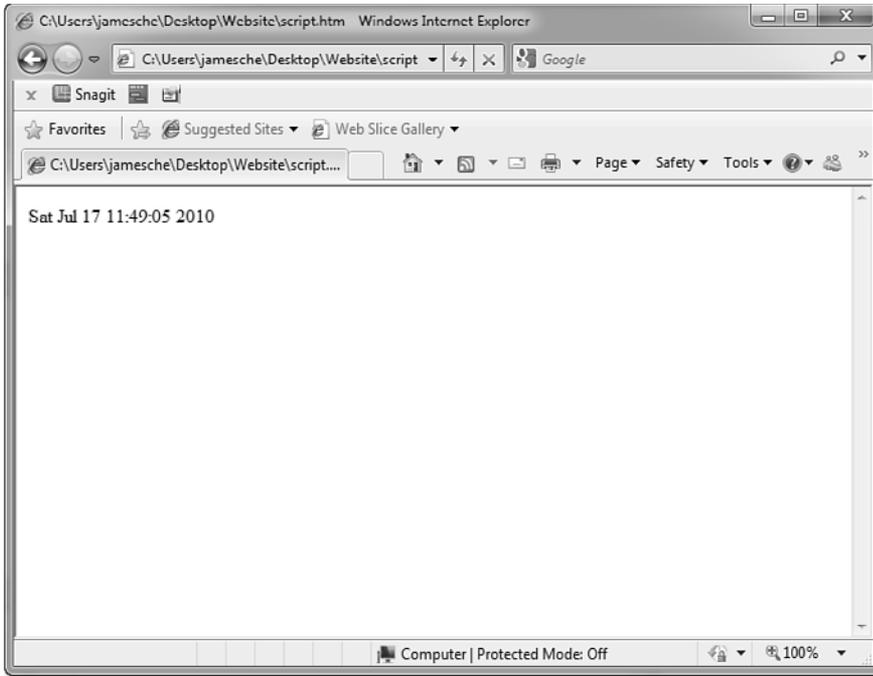


Figure 22.1
The output of the `writeDateTime()` JavaScript function.

Linking to an External Script File

JavaScript can also be included in an external file and linked to a page with the `<script>` tag. The following code calls the same JavaScript function as the previous example, but it uses an external script file:

```
<html>
<head>
  <script language="javascript" src="jscript.js" type="text/javascript">
  </script>
</head>

<body onload="writeDateTime();">
</body>
</html>
```

The `jscript.js` file contains the JavaScript function that is called in the `onload` attribute of the `<body>` tag. The code inside `jscript.js` is as follows:

```
function writeDateTime() {
  document.write(Date());
}
```

Note that the `jscript.js` file does not contain any HTML code, not even the `<script>` tag. It contains only JavaScript code.

Including your scripts in separate files has benefits. The primary benefit is that reusing the scripts in many files without duplicating the code in each file is simple. You also can easily change scripts if needed. If you include a script inside HTML files, a change in the script must be made in every HTML file that includes the script. If the script is inside a `.js` file, the script only needs to be changed once—in the `.js` file. All HTML files that link to that script will then run the updated script automatically.

Adding Inline JavaScript

As mentioned earlier, JavaScript can be entered as standalone code instead of being contained inside a function. When JavaScript is entered into a page as standalone code, the code is executed as soon as it is encountered. The following code produces the same output as the code you saw previously, but it does so with a standalone code segment instead of a JavaScript function:

```
<html>
<head>
</head>
<body>
  <script language="javascript">
<!--
    document.write(Date());
-->
</script>
</body>
</html>
```

Notice that this example does not contain a line declaring a JavaScript function. Instead, only the code that was previously inside the `writeDateTime` function is included. In the previous example, for the code to execute, you had to call the `writeDateTime` function from the `onload` attribute of the `<body>` tag. In this example, no function call is required because the JavaScript code is not within a function. As you can see, placing JavaScript code inside a function allows you to control when that code is run.



caution

Be aware that browsers store script files in the browser's cache. If a user's browser does not check for an updated script file when your page is requested and uses the script file in cache instead, unpredictable behavior can occur.

Unfortunately, there isn't a simple solution to this, but one thing you can do is include some text on your site (perhaps in an FAQ section) that explains how to clear the browser's cache.



tip

Because JavaScript inside a function does not run until the function is called, the web browser will not inform you of any errors in that code until you call the function. Therefore, when you are testing pages with JavaScript functions in them, be sure you call all your functions during testing. One nice way of doing this is to create a test page that calls all of your functions.

The Document Object Model

A lot of the interactivity in pages today is accomplished using DHTML. When you write DHTML code, you use scripting (typically JavaScript) to control the CSS attributes of HTML elements on a page. For example, many of the flyout menus so commonly seen on the Internet these days (see

Figure 22.2) are created by causing page elements to appear when a mouse is hovered over a menu item and disappear when the mouse pointer is moved off the menu item. These types of menus are often referred to as DHTML menus because they use DHTML for their functionality.

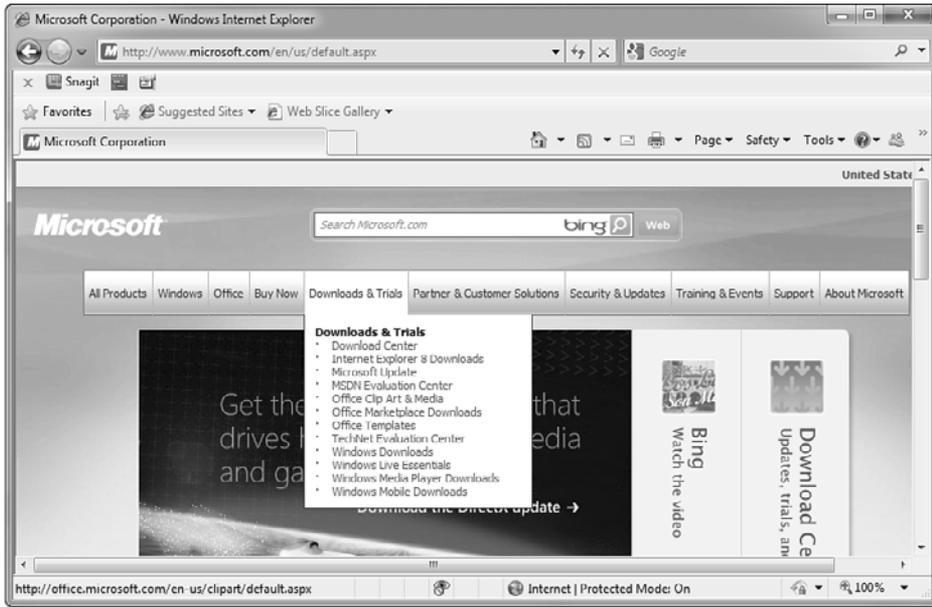


Figure 22.2 Flyout menus, such as this one on the Microsoft site, are created with JavaScript code.

To script the elements on a page, the JavaScript code must have some way to programmatically access those elements. It does this using the Document Object Model (DOM) for the browser. For the most part, the DOMs for the major browsers are similar, but that doesn't necessarily mean that code that works in one will work in another. Even so, with a bit of work, you can write JavaScript code that is compatible with all the major browsers available today.

At the top of the DOM hierarchy is the `window` object. Underneath the `window` object, you will find an extensive list of objects—far too many to cover in a single chapter of this book. However, you learn about some of the commonly used objects in the DOM.

note

The W3C has a standard for the DOM, but most web browsers do not strictly conform to it. Instead, most browsers implement their own DOM that includes specific functionality only for that browser.

The window Object

The `window` object refers to the browser window itself. It is used to manipulate the browser window and its elements. For example, to change what is displayed in the status bar (the lower portion of the browser window), you would use the `status` property of the `window` object.

The following code example changes the status bar message:

```
<html>
<head>
  <script language="javascript" type="text/javascript">
    function changeStatus(msg) {
      window.status = msg;
      return true;
    }
  </script>
</head>
<body onload="changeStatus('Welcome!');">
</body>
</html>
```

When this page is opened in the web browser, the status bar text will say *Welcome!*. Note that when the `changeStatus` function is called, the text *Welcome!* is passed to it in parentheses. When the function runs, the text passed to it is assigned to the `msg` variable, which is then displayed in the status bar.

Using this method of passing a value to a function makes it convenient to reuse the function for other purposes. For example, to display a different message in the status bar of the browser at a different point in the page, you would simply call the `changeStatus` function again and pass the text you want to display to it. This function can be made even more robust by saving it as an external `.js` file and simply linking that file to each page that needs to use the `changeStatus` function.

The document Object

One of the objects under the `window` object in the DOM hierarchy is the document object. The document object is one of the most frequently used objects by JavaScript programmers because it provides access to all the elements on a page.

In Internet Explorer, the `all` collection of the document object contains a reference to every element on a page and is often used by Internet Explorer developers to reference a particular item. For example, consider this `<div>` tag:

```
<div id="border">page border goes here.</div>
```

To get a reference to this `<div>` tag, you would simply use the following line of code:

```
document.all("border")
```

By appending the ID of the `<div>` to the `all` collection, this line of code returns a reference to the `<div>` with that ID.

note

Firefox actually supports `document.all`, but not in the way you might expect. If you use `document.all` in a script loaded into Firefox, you'll see a friendly message notifying you that you've used a nonstandard syntax and referring you to the `document.getElementById` method.

This works great in Internet Explorer, but it won't work in other browsers because only Internet Explorer supports the `all` collection of the document object. Therefore, a better method to get a reference to the `<div>` tag is to use the `getElementById` method of the document object. The following code gets a reference to the `<div>` tag, and it works in all modern browsers:

```
document.getElementById("border")
```

This code returns a `div` element that references the `<div>` tag called `border`. After you have that reference, you can then programmatically interact with the `<div>` tag, as you will see later.

Many other objects exist in a browser's DOM. By reviewing developer information provided by the browser you are targeting, you should be able to easily take advantage of what the DOM has to offer. The best way for you to start down that road is to write a little code, so let's write a few sample scripts that implement some real-world scenarios.

Writing Simple Scripts

Now that you have a general idea of how to write JavaScript code, you are ready to look at a few examples of how you can use JavaScript in your pages. In this section, you learn how to hide and show page elements using JavaScript, how to access elements on a page and read and change their attributes, and how to check information entered into a form before it's submitted. These three tasks are the most common tasks taken on by JavaScript developers.

Showing and Hiding Page Elements

One of the most common techniques in browser scripting is changing page content based on certain conditions, such as when the mouse pointer passes over a particular graphic. This type of effect is easy to implement with Expression Web Behaviors, but you might want to edit the code that Expression Web generates. You also might find that a Behavior doesn't do exactly what you need and decide to implement your own script. In these cases, understanding how this type of effect is achieved with JavaScript is invaluable.

In this example, you create a page with a list of links on the left side. When you pass over a link, text on the page changes to indicate the nature of the link you are pointing to. All this is accomplished using DHTML, which is programmed using JavaScript. Do the following:

1. Create a new site or open an existing site.
2. Create an empty page.
3. Insert a table with one row, two columns, no border, and a width of 100%.
4. Right-click the left column and select Cell Properties.
5. Check the Specify Width check box.



note

The finished pages and files for all these scripts can be downloaded from the website for this book at www.informit.com/.

6. Select the In Pixels option and enter **150** for the width.
7. Click OK.
8. Save your page.

The left column contains links to parts of the site. When you hover over each link, text describing that link appears in the right column of the table. To implement this, you need to insert a `div` to hold the text description for each link. When you hover over a link, you display the `div` for that link and hide all the other `divs`.

Enter the following text in the left column and press the Enter key after each item:

- Home Page
- Our Products
- About Us
- Contact Us

In a real-world site, you would link each of these to its respective pages in the site; but for now, simply link each one to the page you are currently editing so you will have a hyperlink to work with.

Now you need to create some `<div>` tags to hold the text for each link. To do this, you use the layers feature in Expression Web.

1. If the Layers panel is not visible, select Panels, Layers to display it.
2. Add a new layer to the page.
3. Size and position the layer so it appears in the right column of your table.
4. Right-click the layer and select Copy.
5. Right-click the layer again and select Paste.
6. Right-click the layer and select Paste two more times so there are four total layers—one right on top of the other.
7. Right-click the first layer in the Layers panel and select Modify ID.
8. Change the first layer's ID to Home.
9. Change the second layer's ID to Products.
10. Change the third layer's ID to About.
11. Change the fourth layer's ID to Contact.

**tip**

Expression Web uses absolute positioning for layers. All the layers you inserted are stacked on top of one another. Therefore, any text added to a `div` will appear in the same place on the page because each `div` sits on its own layer at the same position on the page.

12. Right-click the Home layer and select Set Visibility: Hidden.
13. Repeat step 12 for the remaining layers.
14. Save the page.

Your page should now look like Figure 22.3.

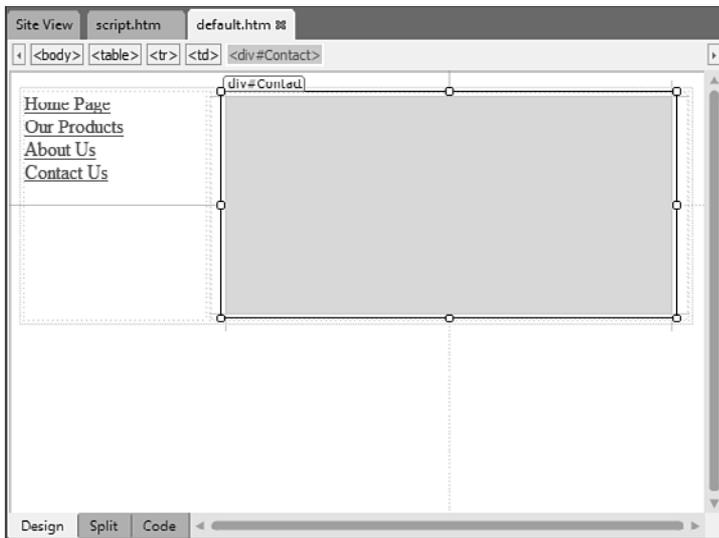
➔ *For more information on using layers, see Chapter 23, "Using Layers."*

When you make layers hidden using the panel, Expression Web simply adds a CSS property to the `<div>` tags to make them hidden. That property is the `visibility` property. For example, the following `<div>` tag is hidden:

```
<div id="Products" style="visibility: hidden;">I am hidden.</div>
```

**tip**

When a `div` is not absolutely positioned, hiding it using the `visibility` property makes it invisible, but the browser still reserves space for the `div`. The result is an empty area where the `div` is on the page. To prevent this, use the `display` property for a `div` that is not absolutely positioned. When the `display` property is set to `none`, the `div` is not displayed and the browser closes up the space where the `div` used to be.

**Figure 22.3**

The page is now ready for you to add some code.

When the page containing this `<div>` tag is loaded, the `div` will be invisible.

For each layer, enter some text that describes the content for the link. To do this, first click the layer in the Layers panel and then click inside the layer. Enter any text you choose for each layer. You might want to change the visibility to Visible temporarily while you enter the text so you can see what you're typing.

When someone visits your site, you want the text for the Home layer to be visible right away. If the site visitor then hovers over one of your other links, you want the layer for that link to be displayed. To do that, you need to enter some code.

Switch to Code View and add the following JavaScript code before the closing `</head>` tag:

```
<script language="javascript" type="text/javascript">
  function hideAllDivs() {
    document.getElementById('Home').style.visibility = 'hidden';
    document.getElementById('Products').style.visibility = 'hidden';
    document.getElementById('About').style.visibility = 'hidden';
    document.getElementById('Contact').style.visibility = 'hidden';
  }

  function changeVisibility(layer) {
    document.getElementById(layer).style.visibility = 'visible';
    return true;
  }
</script>
```

 For more information on entering code in Code View, see Chapter 4, “Using Page Views.”

Two JavaScript functions exist in this code. The first one, `hideAllDivs`, sets the `visibility` property of each `div` to `hidden`. Remember that when you set a layer to be invisible, Expression Web sets an inline style by adding a `style` attribute to the `<div>` tag. The value of that style attribute is `visibility: hidden`. To programmatically set an inline style on an element, you use the `style` attribute of the element. In the `hideAllDivs` function, you are setting an inline style for each `div`, and that inline style sets the `visibility` property to `hidden`.

The second function, `changeVisibility`, makes whatever layer name is passed to it visible by setting the `visibility` property to `visible`. For example, to make the `Products` layer visible, you would call the `changeVisibility` function using the following code:

```
changeVisibility('Products');
```

When the `changeVisibility` function is called with this line of code, the `layer` variable in the function is assigned the value `Products`, and the `getElementById` function then returns a reference to the `Products` layer on the page. The `style` attribute is then used to set the `visibility` property to `visible`.

To finish the page, you need to add some code that calls these functions at the appropriate times. First, you need to ensure the `Home` layer is visible when the page is first loaded. To do that, call the `changeVisibility` function for the `Home` layer when the page loads.

Make sure you're still in Code View and edit the `<body>` tag of your page so it resembles the following:

```
<body onload="changeVisibility('Home');">
```

The `onload` event of the `<body>` tag is triggered automatically as soon as the page has finished loading. When an event is triggered, that event is said to have fired. By adding a call to the `changeVisibility` function in the `onload` event of the `<body>` tag, you cause the `Home` layer to become visible when the `onload` event is fired.

The final step is to create hyperlinks so that clicking the text you entered in the left cell causes the correct layers to appear and disappear as the mouse moves over them. To add this functionality, you need two events: the `onmouseover` event and the `onmouseout` event. The `onmouseover` event is fired when the mouse moves over the element, and the `onmouseout` event is fired when the mouse moves off the element.

Select **Our Products** in the left cell, and then right-click and select **Hyperlink**. Select `default.html`, and then click **OK**. Now switch to **Source view** and locate the `<a>` tag for the **Products** link and edit it so it resembles the following:

```
<a href="default.htm"
onmouseover="hideAllDivs();changeVisibility('Products');"
onmouseout="hideAllDivs();changeVisibility('Home');">
```

The `onmouseover` event first calls the `hideAllDivs` function. This causes any visible `div` to be hidden in preparation for displaying the **Products** layer. It then calls the `changeVisibility` function and passes `Products` to it. This causes the **Products** layer to become visible. Note that a semicolon appears between the two function calls.

When the mouse is moved off the **Products** link, the `onmouseout` event is fired. This event again calls the `hideAllDivs` function, which hides the **Products** layer. It then calls the `changeVisibility` function to make the **Home** layer visible again because we're on the home page.

To finish the page, add a hyperlink to the **About** and **Contact** text just as you did with **Products**, and then edit the hyperlinks as you just did with the **Products** hyperlink. Make sure you pass the correct layer name to the `changeVisibility` function in the `onmouseover` event. After you edit your hyperlinks, save the page and preview it in your browser to see your layers swapped out as you hover over the hyperlinks.



divs Don't Change on Rollover

If you roll the mouse over my links and the text doesn't change, you probably have a typographical error in your code. Look in the status bar of the browser and make sure that an error doesn't appear in your script. If it does, carefully review the code you entered to make sure you haven't made a mistake.

This can also be caused by having scripting turned off in your browser. To check this in Internet Explorer, select **Tools, Internet Options** and click the **Security** tab. Select the appropriate zone (select **Internet** if using a URL with dots in it and **Intranet** if using a URL without dots) and click the **Custom** button. Make sure the **Enabled** option is selected for **Active Scripting**. At the time of this writing, other browser companies are preparing to release major updates, so check for scripting in other browsers. Consult the documentation provided with the browser.

Keep in mind that if you have a disk-based site open in Expression Web, you need to click the yellow information bar in your browser and allow scripts to run when previewing pages.

➔ *For more information on add-ins in Expression Web, see the online Chapter 36, Expression Web 4 Add-in Basics.”*

Accessing and Changing Attributes

JavaScript code is often used to access attributes of HTML tags. Using JavaScript, you can read the value of a particular attribute and also change the value of an attribute. Pages that use image swapping use this technique to change the image file displayed when the mouse hovers over an image. In this section, you write some JavaScript code to swap an image when your mouse hovers over it.

The HTML `` tag has an attribute called `src` that specifies the image file to be displayed. By using JavaScript to change the `src` attribute of an image tag, you can easily create the effect of swapping one image with another. To do this, you need to perform two primary tasks: preload the images and write the code to swap the images.

Before you get started, you need a couple of images to work with. You can use the `ewdlogo.jpg` and `ewdlogo_over.jpg` available on this book's website at www.quepublishing.com, or you can use your own images. Just make sure the images are the same size so the rollover effect works correctly. Save whatever images you decide to use in the `images` folder of your site.

Preloading Images

Your first step is to add JavaScript code to preload the images that will be swapped. You want to preload the images because if you don't, when your site visitors hover over your original image, they will have to wait for the browser to download the second image before the images are swapped. This delay can take several seconds on a slow Internet connection, and that delay will make your effect seem unprofessional.

Preloading images with JavaScript is an easy task. Open a new page and enter the following JavaScript code before the closing `</head>` tag:

```
<script language="javascript" type="text/javascript">
  grayImg = new Image();
  grayImg.src = "images/ewdlogo.jpg";
  colorImg = new Image();
  colorImg.src = "images/ewdlogo_over.jpg";
</script>
```



tip

You can also add the Mark of the Web to your page to allow scripts to run from a disk location. Expression Web comes with an add-in that makes adding the Mark of the Web easy. Select Insert, HTML, Mark of the Web to add it.



tip

You can also use the Preload Images behavior to preload images. However, because I feel that understanding what goes on when you use that behavior is important, I have included the code necessary to preload images.

As a matter of fact, the Swap Image and Swap Image Restore behaviors can be used to implement the example demonstrated here. However, by learning how the code underlying those behaviors works, you'll be able to build scripts yourself that don't lock you into one particular implementation.

This code defines two variables called `grayImg` and `colorImg`. It then sets these variables equal to a new `Image` object. An `Image` object is an object that represents an HTML `` tag. You then set the `src` attribute of the new `Image` object to the image file you want to display for that object. After this code runs, you will have two `Image` objects: one for the `ewdlogo.jpg` image (the initial image) and one for the `ewdlogo_over.jpg` image.

Note that this code does not exist within a function call. That's because you want this code to run when the page loads. You could place this code within a JavaScript function and call it in the `onload` event of the `<body>` tag, but doing that would cause the script to run immediately *after* the page loads. If your site visitor were to mouse over the image as soon as the page finishes loading, you wouldn't get the benefit of preloading the image.

Writing a Function to Swap Images

The next step is to write a function to swap the images. Because you might want to reuse this function in other pages, it makes sense to write it so it isn't specific to the images you are using in this example. Instead, the function should be written as a generic function that can swap images based on the information passed to it.

Edit your script to include the `swapImage` function as follows:

```
<script language="javascript" type="text/javascript">
  grayImg = new Image();
  grayImg.src = "images/ewdlogo.jpg";
  colorImg = new Image();
  colorImg.src = "images/ewdlogo_over.jpg";

  function swapImage(imgID, imgObj) {
    if (document.images) {
      document.images[imgID].src = imgObj.src;
    }
  }
</script>
```

The `swapImage` function takes two parameters: `imgID` and `imgObj`. The `imgID` variable contains the `id` attribute of the image that's being swapped. This allows you to refer to the correct page element. The `imgObj` variable is the object name for the image you want to display in place of the original image. When the function runs, the `src` attribute of the original image is changed to the `src` attribute for the swapped image. Because the images have been preloaded, the result of this function is that the image instantly changes.

Adding the Images

Now the JavaScript code is in place. All that's left is to insert the original image onto the page and then add some JavaScript function calls to the `` tag. Insert the `ewdlogo.jpg` image onto your page. Using the Quick Tag Editor, edit the `` tag so it appears as follows:

```

```

➡ *For more information on the Quick Tag Editor, see Chapter 8, “Using the Quick Tag Tools.”*

The `` tag now contains code for the `onmouseover` event that calls the `swapImage` function and passes `'ewdlogo'` and `colorImg` to it. This tells the JavaScript function you are changing the image for the tag with an `id` attribute of `ewdlogo`. It also tells the JavaScript function that you want to change the `src` attribute of the tag so it's equal to the `src` attribute of the `colorImg` object. The `src` attribute of the `colorImg` object is set when the images are preloaded, so when the mouse hovers over the grayscale picture of the Expression Web graphic, it changes automatically to a color picture right before your eyes. The `onmouseout` event calls the `swapImage` function again to change the images back. Using this exact same procedure, you are now equipped to write your own JavaScript rollover buttons.



Images Don't Swap Instantly

Internet Explorer can be configured so it checks the web server each time a file is requested to see whether a newer version is on the server. This check can cause a delay in the display of rollover images.

To correct this, select **Tools, Internet Options**. Make sure the **General** tab is selected and click the **Settings** button. In the **Check for Newer Versions of Stored Pages** section, select the **Automatically** option and click **OK**.

Form Field Validation

JavaScript is also commonly used to validate form fields in pages. In this section, you write JavaScript that validates an HTML form and ensures that data was entered in each form field. It also checks to ensure that only numeric characters are entered in a particular field and that no more than three digits are entered in an age field.

Creating the Form to Validate

First, you need to create an HTML form to be validated by your script:

1. Create a new page in your site.
2. Insert an **Input (Text)** form control from the **Form Controls** section of the **Toolbox**.
3. Insert another **Input (Text)** control on the form under the first one.
4. Type **Enter your name:** above the first text box.
5. Type **Enter your age:** above the second text box.

6. Double-click the first text box and change the name to txtName.
7. Double-click the second text box and change the name to txtAge.
8. Insert an Input (Submit) control under the second Input (Text) control.
9. Save the page.

Your page should now resemble Figure 22.4.

note

If you'd prefer to use a completed sample instead of entering the code yourself, you can download the code from the website for this book at www.informit.com.



Figure 22.4

The finished HTML form that will be validated with JavaScript.

Adding the JavaScript Validation Function

Now you need to enter some JavaScript code to validate the form. Here's the JavaScript function to validate both form fields:

```
<script language="javascript" type="text/javascript">
function validateForm(theForm) {
    var txtName;
    var txtAge;
    var nums = '0123456789';

    txtName = theForm.elements[0];
    txtAge = theForm.elements[1];
```

```

if ((txtName.value == '') || (txtAge.value == '')) {
    alert('Please specify both your name and your age.');
```

```

    return false;
}
for (var i = 0; i < txtAge.value.length; i++) {
    if (nums.indexOf(txtAge.value.charAt(i)) == -1) {
        alert('You can only specify numeric data for age.');
```

```

        return false;
    } else if (txtAge.value.length > 3) {
        alert('You cannot possibly be that old.');
```

```

        return false;
    }
}
return true;
}
</script>
```

Go ahead and add this script to your page just before the closing `</head>` tag.

This script is the most complicated yet, but it's not quite as complex as it appears at first glance. The first three lines set up three variables: one for the `txtName` form field, one for the `txtAge` form field, and one for the numeric characters you will validate the `txtAge` field against.

Next, the script sets the `txtName` and `txtAge` variables equal to their respective form fields. To get a reference to each form field, you use the variable called `theForm`. This variable holds a reference to whatever has been passed into the `validateForm` function. As you will see later, a reference to the form itself is passed to this function. The `elements` collection contains one object for each form field in the form. The first form field is called `elements[0]`, the second `elements[1]`, and so on. The first element in your form is the `txtName` form field, so the `txtName` variable is assigned to `theForm.elements[0]`. The `txtAge` variable is then assigned to `theForm.elements[1]`, the `txtAge` form field.

Validating the Form Fields

When you have a reference to both the `txtName` and `txtAge` form fields, you must check to ensure that both contain data. Do this by checking to see whether their `value` property is an empty string with the following line of code:

```
if ((txtName.value == '') || (txtAge.value == ''))
```

The `value` property returns the data entered into the form field. If the `value` property returns an empty string, you know the user hasn't entered any data and you must display an appropriate message using the `alert` method. (Notice that to check whether one value is equal to another value in JavaScript, you use double equals signs. The double pipe symbol (`||`) is the logical OR operator in JavaScript.) Therefore, if either `txtAge` or `txtName` contain no data, validation will fail.

note

The `++` symbol means to increment the value to the left by 1. This syntax is used in many languages other than JavaScript, including C and C++. In fact, C++ got its name from the fact that its developers believed it to be one better than C.

The next validation to perform is checking whether the `txtAge` form field contains any non-numeric values. To do this, you use a string variable (`nums`) that contains all the valid numerical values. You then check each character in the `txtAge` form field against that string variable. Here is the code segment that performs the check:

```
for (var i = 0; i < txtAge.value.length; i++) {  
    if (nums.indexOf(txtAge.value.charAt(i)) == -1) {
```

The first line sets up a `for` loop. A `for` loop runs through a particular code segment repeatedly as long as a particular condition is met. When you define a `for` loop, you specify three items that control how the loop is executed: a variable that indicates how many times the loop has run, a condition that must be met for the loop to continue running, and an incrementer for the loop counter that adds 1 to its present value each time the loop runs.

In the loop example, the variable that indicates how many times the loop has run is called `i`, and it is initialized to 0 at the beginning of the loop. The condition is then specified so that as long as `i` is less than the length of whatever is entered into the `txtAge` form field, the code segment will continue to be executed. Finally, `i` is incremented with the `i++` statement, which adds 1 to its present value.

When this code runs, the loop executes once for every character in the `txtAge` form field. Each time it runs, it checks a single character in the `txtAge` form field (using the `charAt` function) to see whether it contains any value other than one of the numbers in the `nums` variable. It does this using the `indexOf` function. The `indexOf` function returns the position of one string within another string. If the string is not found, a value of `-1` is returned. Each character in the `txtAge` form field should be found somewhere within the `nums` string variable. If it is not, you know it is not a numeric value and you must display the appropriate message.

The final check determines whether the length of the text entered into the `txtAge` form field exceeds three digits. If it does, you must display a message letting the user know he has lied about his age.

If any of the previous validations fail, you must return a value of `false` from the `validateForm` function because validation has failed. After a value is returned from a function, processing of that function stops. Therefore, the last line in the function must return a value of `true` because you know if you've gotten that far, validation has succeeded.

Adding the Call to the JavaScript Function

You need to add one final bit of code to make this all work. You need to add a call to the `validateForm` function. You do that in the `onsubmit` event of the form. Using the Quick Tag Editor, edit the `<form>` tag as follows:

```
<form method="post" onsubmit="return validateForm(this);">
```

The `onsubmit` event fires when the form is submitted. In this event, you call the `validateForm` function and pass a reference to the form using the `this` keyword. The `this` keyword always contains a reference to the particular element containing it. It is a convenient way to pass a reference to an element of a function. Because you used `return` when calling the function, the `return`

statement inside the `validateForm` function returns either `true` or `false`. If the value is `false`, the form will not be submitted. If the value is `true`, the form will submit as usual.

Save the page now and preview it in your browser. Submit the form without entering any data and examine the result. Next, enter your name in both fields and note that you are told that the `txtAge` form field can only contain numeric data. Finally, enter an age of `1000` and try to submit the form.

This is a simple validation script. In a real-world environment, you would want your validation script to be much more robust than this. However, this script gives you a solid foundation on the methods used when validating forms.

Debugging

So far, the assumption has been that all the JavaScript you've entered will run without problems. As long as you've entered it correctly, it will, but that's because I've already debugged it for you! In the real world, code almost never runs successfully on the first try. Sometimes you get lucky and your code runs without debugging, but almost all code requires some level of debugging to get the desired results.

There are two types of problems you can encounter with code: syntax errors and logic errors. Syntax errors are often the easiest errors to resolve. For example, the following line of code generates an error:

```
document.write('Welcome!');
```

Remember, JavaScript is case-sensitive. The `write` method does not start with an uppercase `W`, so when this line executes, it generates an error.

Logic errors are much harder to track down. A logic error occurs when code is written so that it runs without syntax errors, but the code produces undesired results. Logic errors are hard to track down because when an error occurs, it often points to a part of code that is nowhere near where the actual error is located. Suppose, for example, you have a function that returns a specific numeric value and that function contains a logic error. You have code in a completely different area that relies on the number returned from the first function. When the code runs and an error occurs, the error message might point to the section of code using the number returned from the function, not the function itself.

There are many approaches to debugging JavaScript code. One is to use the JavaScript `alert` method to display messages at certain places in the code. For example, if a function returns a specific numeric value, you can place an `alert` method at the end of the function call and display the value the function returns, as shown in the following example:

```
function debugTest() {
    var i;
    i = document.getElementById('txtYears').value;
    alert(i);
    return i;
}
```

When this function runs, an alert dialog box displays, indicating the value of `i`. An even more effective way to debug client script is to use the script debugging functionality included in Internet Explorer 8. Select Tools, Developer Tools, and then click the Script tab. For more information on debugging with Developer Tools in Internet Explorer 8, visit msdn.microsoft.com/en-us/library/dd565625%28VS.85%29.aspx.

Easier Scripting with jQuery

As you might have noticed, using JavaScript to manipulate the browser's DOM is sometimes a little tricky. Different browsers work a little bit differently, so you have to take that into account when writing your JavaScript. It can also be frustrating when you have to write several lines of code to access a particular element on a page. Most developers have experienced this frustration, and thankfully, there is a solution: the jQuery library.

The jQuery library consists of a JavaScript file that can be linked to as an external script file. jQuery adds powerful methods for accessing and manipulating DOM elements, and when you use jQuery, you don't have to worry about browser interoperability. jQuery takes care of that for you.

After you've downloaded the jQuery library, you can link the script to a page just like you did earlier in this chapter. After you do that, you'll immediately have access to all the power of jQuery. Even better, as long as you have Service Pack 2 or later installed, you'll have IntelliSense for jQuery inside of Expression Web 4.

Coverage of JavaScript programming using jQuery is outside of the scope of this book, but there are great tutorials and other information available at www.jquery.com.



note

You can download jQuery from www.jquery.com. There are two versions: a developer version, and a production version. The production version is compressed so that it's smaller and will take less time to download. Unless you plan on making changes to the jQuery library (or debug the library), I recommend you get the production version.

USING LAYERS

Introduction to Layers

In the early days of the Web, designers were content with having a page with columns of text for their hyperlinks. Pages were boring and drab, but they served their purpose. As the Internet became more popular, sites became more interactive and dynamic. Instead of plain text and graphics, it is much more common now to see dynamic, graphically oriented pages with content that changes in real time as you interact with the page.

A web designer can use many techniques to create a page in which certain areas of the page change as the user interacts with the page. Many web designers use `<div>` tags to implement this kind of functionality because by using the `style` attribute of the `<div>` tag, you can easily control the `<div>` tag's position, size, and many other attributes.

Expression Web uses absolutely positioned `<div>` tags to implement layers so you have fine control over their location on the page. When an item on a page is absolutely positioned, the tag used to render that item contains attributes that specify exactly where that item is to appear on the page. The position is specified using three attributes:

- `left`—The distance between the left side of the page and the left side of the item.
- `top`—The distance between the top of the page and the top of the item.
- `z-index`—The position of the item in relation to other absolutely positioned items if you were to stack the items one on top of the other. A higher `z-index` indicates that an item is higher in the stack.

The *z-index* is what defines a `<div>` tag as an actual layer. Because the *z-index* allows you to stack `<div>` tags in layers on a page, absolutely positioned `<div>` tags are known as layers in Expression Web.

Expression Web uses the Layers panel to provide an intuitive interface with which to add and configure layers on your page. Even with layers, you will not be required to manipulate code directly.

Over the course of this chapter, you create a page with a dynamic menu system and other dynamic content using layers, behaviors, and interactive buttons. Better yet, you create it all without looking at a single line of code!

➔ *For more information on using behaviors in Expression Web, see Chapter 21, “Using Behaviors.”*

➔ *For more information on using interactive buttons, see Chapter 20, “Using Interactive Buttons.”*

Inserting and Configuring Layers

Layers are inserted and formatted using the Layers panel. You have other ways you can insert a layer, but the Layers panel is the most efficient because it is also used to format and arrange layers. If the Layers panel is not visible, simply select Panels, Layers to display the Layers panel, as seen in Figure 23.1. From the Layers panel, you can click the Insert Layer button or the Draw Layer button to insert a new layer into the page.

In addition to the Insert Layer and Draw Layer buttons, the Layers panel consists of three columns. The leftmost column, the Layer Visibility column, allows you to control whether a particular layer is visible and to see at a glance which layers are visible and which ones are not. The second column, the Layer Z-Index column, provides a quick reference to the *z-index* of each layer. The rightmost column, the Layer ID column, displays the ID for each layer.



caution

If you change the properties of a layer and turn off absolute positioning for the layer, or if you edit a page in Code View and remove the `style` attribute, the `<div>` is no longer considered a layer and Expression Web will no longer provide you with the tools to edit it as a layer.



note

Absolutely positioned page elements can often wreak havoc on page layout. The example presented in this chapter is meant to teach you how to use layers in Expression Web, but you will want to carefully test any use of layers before using them in a real site.



note

If you'd prefer to use a completed copy of the page described in this chapter, see the file `layers.htm` in the `Examples\Ch23\Files` folder on the website that accompanies this book for the completed example.

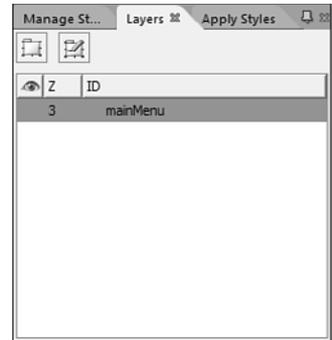


tip

By default, the Layers tab appears inside the same panel as the Apply Styles, Manage Styles, and Behavior panels.

Figure 23.1

The Layers panel is used to insert and configure layers.



Adding Content to a Layer

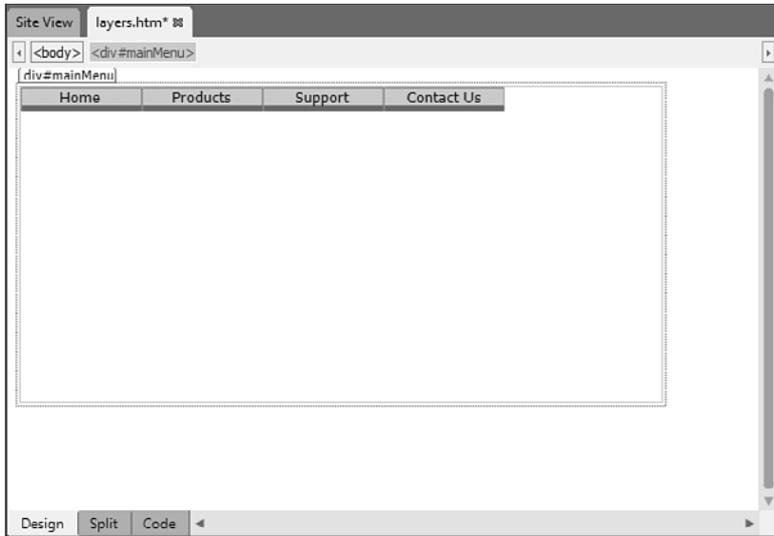
Let's begin creating the page you'll use throughout this chapter to learn about layers in Expression Web:

1. Create a new blank page.
2. If the Layers panel is not visible, select Panel, Layers to display it.
3. Click the Draw Layer button and draw a layer the same width as your page and half the height of your page by clicking in the upper-left corner of the page and dragging the mouse to draw the layer.
4. Click inside the layer to place the insertion point in it.
5. Select Insert, Interactive Button to insert an interactive button.
6. Select the Border Bottom 1 button.
7. Enter **Home** for the text of the button and click OK.
8. Continue this process by repeating steps 6 and 7 to create buttons for Products, Support, and Contact Us. Your page should look like Figure 23.2 when you are finished.
9. Double-click the layer to display the Layers panel if it's not already displayed.
10. Right-click layer1 and select Modify ID to rename the layer.
11. Change the layer name to **mainMenu**.
12. Right-click the mainMenu layer and select Modify Z-Index.
13. Change the z-index for the mainMenu layer to **3**.



tip

Select the ID for each layer carefully and try to choose an ID that describes the layer's purpose. You will use the ID of the layer to refer to the layer in behavior-related dialogs and if you are writing client-side code. By naming your layers with descriptive names such as mainContent and navButtons, you will be less likely to confuse them later.

**Figure 23.2**

The page with one layer and four interactive buttons to use for navigation.

Resizing a Layer

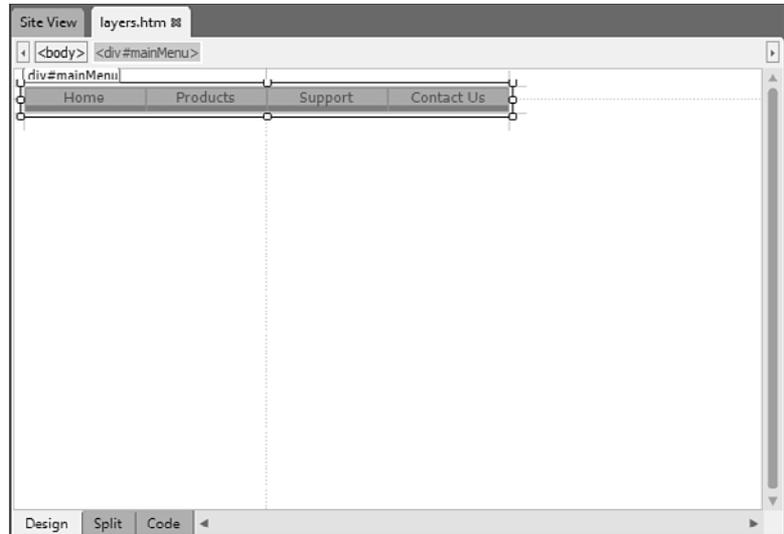
The layer you have just inserted needs to be resized so it is the same size as the four interactive buttons inside it. Change the width and height of the layer to match the width and height of the interactive buttons. To do that, follow these steps:

1. Move the mouse over the border of the layer until the mouse pointer changes to a four-way arrow.
2. Click the border to activate the sizing handles. The sizing handles appear as small white circles at each corner and in the middle of each side of the layer.
3. Move the mouse pointer over a sizing handle until the mouse pointer changes to a two-way arrow.
4. Click and hold the left mouse button while you drag the sizing handle to change the width or height of the layer.

Resize the mainMenu layer until it matches the width and height of the interactive buttons you added previously. You should now have a new layer containing four interactive buttons, as shown in Figure 23.3. The buttons represent the top-level menu. You next create submenus that display when a user hovers over the Products or Support menu buttons.

Figure 23.3

These interactive buttons will be the top-level menu for your dynamic menu built using layers.



Creating and Working with Child Layers

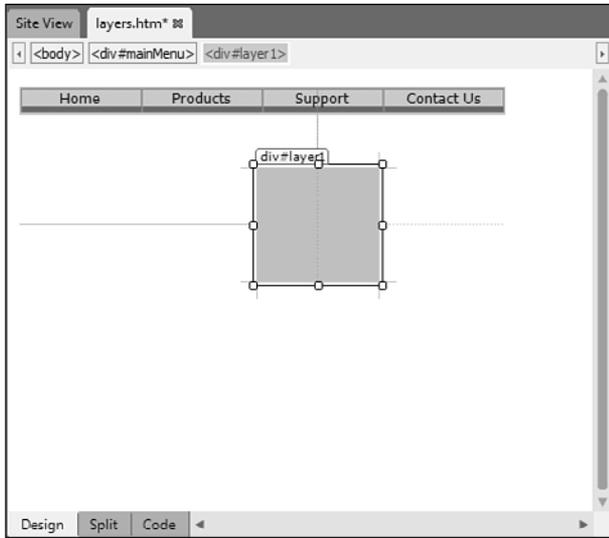
Next, you need to create a layer to hold each submenu. You then control the visibility of the layer based on the position of the mouse. To do that, you'll use the Behaviors and Layers panels.

One of the great things about layers is that because they are absolutely positioned, you can design them anywhere on the page and then simply drag and drop them where you want them to appear when your page is complete. In the case of your menu, you will create three separate layers. As you'll soon see, being able to design these layers and then place them in their final positions makes the development of menus very simple.

The mainMenu layer is the parent layer for all the other layers in the menu system. By making the mainMenu layer the parent, all the layers that make up your menu will be attached to the mainMenu layer. If you move the mainMenu layer, it will move the entire menu system and keep all your layers in place.

You now need to create the first child layer for the mainMenu layer. To do that, follow these steps:

1. Select Panel, Layers to display the Layers panel if it's not already visible.
2. Select the mainMenu layer so the new layer will be inserted as a child layer.
3. Click the Insert Layer button to insert a new layer.
4. Because the new layer is inserted directly on top of the existing layer, select the layer and drag it below the mainMenu layer, as shown in Figure 23.4.

**Figure 23.4**

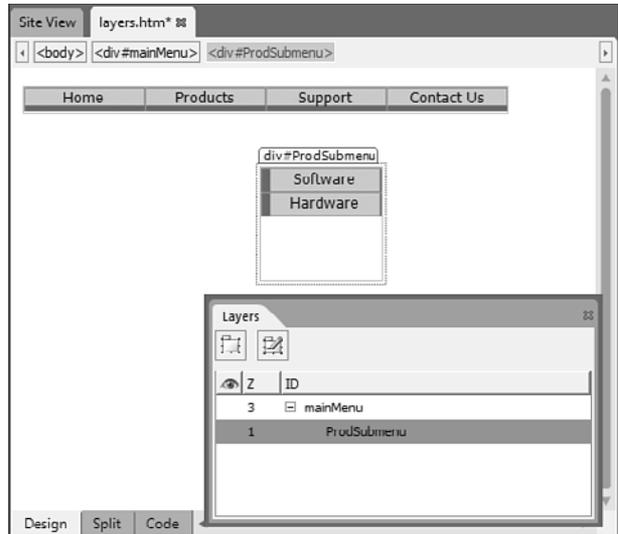
The new layer is positioned right under the mainMenu layer.

5. Insert an interactive button into the layer by selecting Insert, Interactive Button.
6. Select the Border Left 1 button.
7. Enter **Software** for the text of the button and click OK.
8. Press Shift+Enter to insert a line break.
9. Insert another Border Left 1 interactive button immediately below the first button.
10. Enter **Hardware** for the text of the button and click OK.

The mainMenu layer now has a child layer called layer1, and that layer contains two interactive buttons. Resize the layer so that its height and width match the height and width of the interactive buttons.

You should now rename the layer and give it a more descriptive ID. Right-click the layer1 layer in the Layers panel and select Modify ID. Type **ProdSubmenu** for the layer ID. Your page should now resemble the page in Figure 23.5. Note that the ProdSubmenu layer appears slightly indented below the mainMenu layer in the Layers panel, and there is a minus sign next to the mainMenu layer. This indicates that the ProdSubmenu layer is a child layer.

Figure 23.5
The ProdSubmenu layer is a child layer of the mainMenu layer.



The layer is still not in its final position. When you've finished this page, the ProdSubmenu layer will appear just below the Products menu button. It's easiest to design your layers first and then move them into their correct positions later. Because the ProdSubmenu layer is a child layer to the mainMenu layer, it will always move with the mainMenu layer. Therefore, when you place it in its final position, you'll never have to worry about it losing its position relative to the mainMenu layer.

The final pop-up menu you need to create is the Support menu. Copy the ProdSubmenu layer and paste it as a new layer. Rename the new layer **SupportSubmenu**. This layer also needs to be a child of the mainMenu layer. To make it a child layer, click the SupportSubmenu layer in the Layers panel and drag and drop it on top of the mainMenu layer in the Layers panel. The layer is now a child layer at the same level as the ProdSubmenu layer.

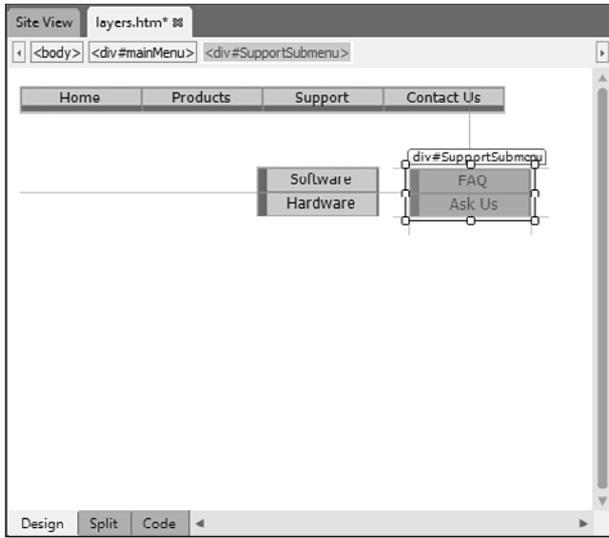
Now, change the interactive buttons in the new SupportSubmenu layer. Double-click the first button and change its text to **FAQ**. Double-click the second button and change its text to **Ask Us**. You now have all the layers complete for your interactive menu, and your page should look like Figure 23.6.

note

When you insert a child layer, Expression Web simply inserts the `<div>` tag for the layer nested within the parent `<div>` tag.

note

When using an interactive button in a real application, you must also configure the button to link to another page via a hyperlink. In the example you're building in this chapter, you don't configure a hyperlink because you're only designing one page to illustrate the use of layers.

**Figure 23.6**

The layers for your dynamic menu have all been inserted.

Positioning Layers

As mentioned previously, the layers on your page are not in their final positions. However, because layers are often stacked on top of each other, they are difficult to edit when placed in their final positions prior to finishing their content. That's why you added all the items to your layers while they were all separated. Now it's time to move the layers into their final positions.

The dynamic menu you are building will display a submenu of products when you hover over the Products button and a submenu of support options when you hover over the Support button. You should position the ProdSubmenu layer so it appears underneath the Products button and slightly overlaps the bottom of the button. You also should position the SupportSubmenu layer so it appears underneath the Support button.

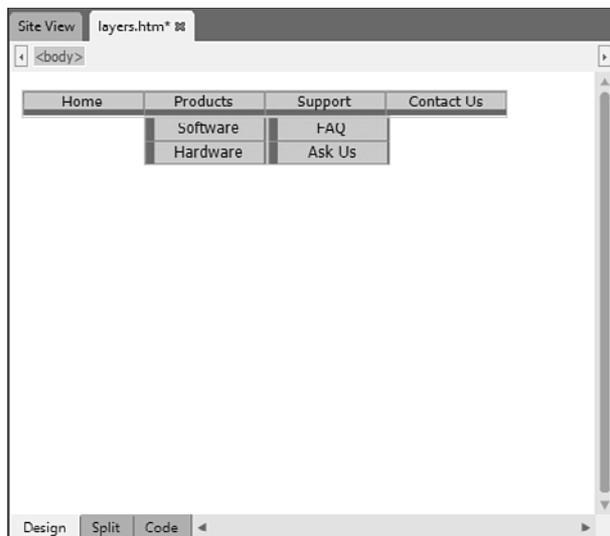
Drag the ProdSubmenu layer so it appears underneath the Products button and the SupportSubmenu layer so it appears underneath the Support button, as shown in Figure 23.7. Make sure the ProdSubmenu and SupportSubmenu layers do not overlap. After you have positioned the ProdSubmenu and SupportSubmenu layers, select the mainMenu layer and drag it to the position where you want your menu to appear on the page. Notice that as you drag the mainMenu layer, the positions of the ProdSubmenu and SupportSubmenu layers do not change in their relation to the mainMenu layer. That's the benefit of making those layers children of the mainMenu layer.



tip

If you have multiple layers on top of one another, selecting the layers in Design View can be difficult. By clicking a layer in the Layers panel, you can easily control the layers you are working on.

Figure 23.7
The completed user interface for your dynamic menu.



Setting Layer Properties with Behaviors

Now you're ready to add interactivity to your menu. When the page first loads, the only layer you want to be visible is the mainMenu layer. When a user hovers over the Products button, the ProdSubmenu layer should display. When a user hovers over the Support button, the SupportSubmenu layer should appear.

In addition to configuring these layers to appear at the correct time, you also need to configure when each layer should disappear. You wouldn't want your menus to appear and stay visible forever. When a user moves the mouse out of a menu, you want the menu to disappear.

The perfect way to implement this interactivity is to use behaviors. By using the Change Property and Change Property Restore behaviors, you can easily configure your layers to appear and disappear when you need them to.

➡ For more information on using behaviors, see Chapter 21, "Using Behaviors."

Setting the Visibility of Layers

When your page loads, the only layer you want the user to see is the mainMenu layer. All the other layers should be invisible. The Layers panel provides an easy method of controlling the visibility of layers using the Layer Visibility column.

The Layer Visibility column is empty by default, but by clicking next to a specific layer, an eyeball icon will appear, as shown in Figure 23.8. The open eye icon means that the layer is visible. The three visibility states for each layer in the Layers panel are as follows:

- **Default**—Indicated by the absence of an eye icon
- **Visible**—Indicated by an open eye icon
- **Invisible**—Indicated by a closed eye icon

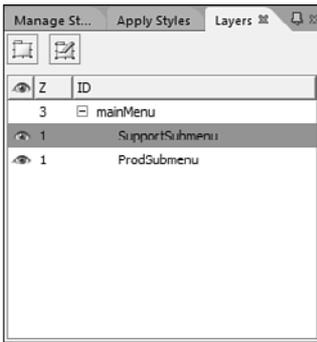


Figure 23.8

The Layer Visibility column provides full control over layer visibility.

Click the eye icon on the ProdSubmenu layer so it appears closed. The Products submenu now appears hidden when the page is first browsed. Click the Layer Visibility column next to the SupportSubmenu layer twice so it is invisible as well. Now save your page and browse it. You will see that the only layer visible is the mainMenu layer, as shown in Figure 23.9.

Adding Layer Interactivity

Now you need to add some interactivity. Remember that the ProdSubmenu layer needs to be made visible when a user hovers over the Products button. To add that interactivity, use the **Change Property** behavior to change the **Visibility** property for the ProdSubmenu layer.

➔ *For more information on the Change Property behavior, see “Expression Web Behaviors,” p. 345.*



tip

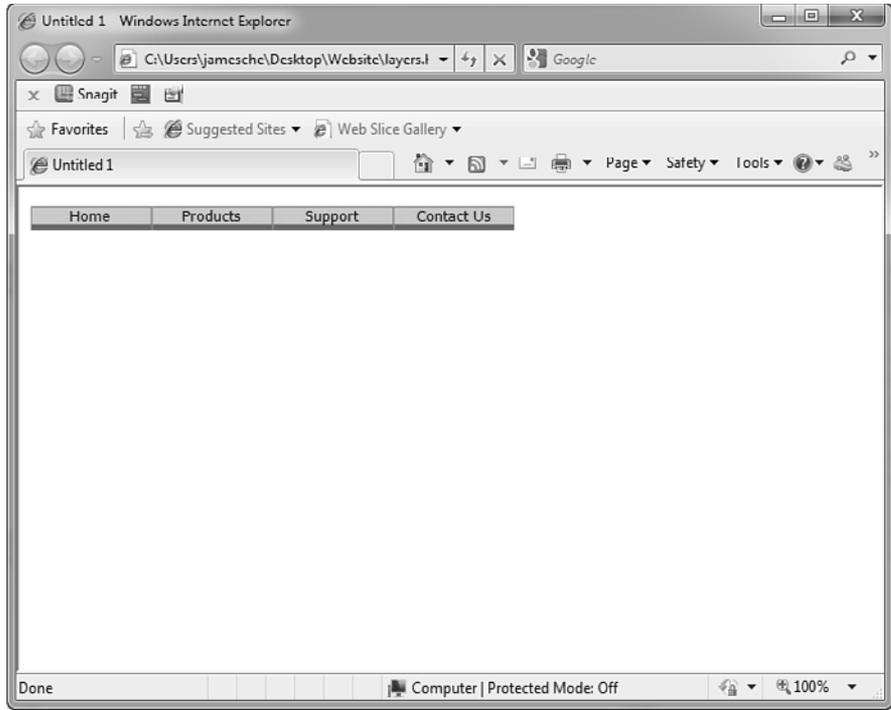
Even when a layer is set to be hidden, it is visible in Design View in Expression Web if it is selected in the Layers panel.



note

Behaviors will already be listed for each interactive button. These are the behaviors Expression Web added automatically to make your interactive buttons work.

Figure 23.9
The main menu appears, but all other layers are hidden.



Configuring the Products Submenu

Select Panels, Behaviors to display the Behaviors panel if it's not already visible. To add interactivity for the Products submenu, follow these steps:

1. Select the Products button.
2. In the Behaviors panel, click the Insert button.
3. Select Change Property from the Behaviors menu.
4. Select the Select Element radio button.
5. Select div from the Element Type drop-down.
6. Select ProdSubmenu from the Element ID drop-down.
7. Click the Visibility button.
8. Select the Visible radio button and click OK.
9. Make sure the Restore on Mouseout Event check box is unchecked.

10. Click OK to add the behavior.
11. Locate the behavior in the Behaviors panel and change the Events from `onclick` to `onmouseover`. The behavior changes position in the Behaviors panel when the event is changed.

Now if you browse the page, the ProdSubmenu layer appears when you hover over the Products menu. However, there is a problem. The Products submenu will never disappear unless you reload the page. You want the Products submenu to disappear when the user moves the mouse off the menu.



Behaviors Don't Work

When you insert a behavior, the event defaults to `onclick`. If you are expecting the behavior to work when you hover the mouse on a page element, but you forgot to change the `onclick` event to `onmouseover`, the behavior will not work.

Make sure you always change the event if necessary.

Switch to the Layers panel and select the ProdSubmenu layer to make it visible so you can apply behaviors to it. Without deselecting the ProdSubmenu layer, switch to the Behaviors panel. To configure the ProdSubmenu layer so it disappears at the correct time, follow these steps:

1. Select Insert, Change Property to display the Change Property dialog.
2. Leave the Current Element option selected.
3. Click the Visibility button and select the Hidden option.
4. Click OK.
5. Make sure the Restore on Mouseout Event box is unchecked.
6. Click OK.
7. In the Behaviors panel, change the Events column from `onclick` to `onmouseout`.

Now the Products button is configured to show the submenu when you hover over it, and the ProductSubmenu layer is configured to hide itself when you move the mouse outside the menu.

Save and test your page to see how it's working so far. Hover over the Products button; you should see the ProductSubmenu layer appear. Move the mouse over the ProductSubmenu layer and you should notice a problem. When you hover over the buttons on the ProductSubmenu layer, the entire menu disappears! This is obviously not what you want to happen. To correct this, you need to add a behavior to each button on the ProductSubmenu layer.



note

Several behaviors will already be listed for the Software button. These are the behaviors Expression Web added automatically to implement the functionality of your interactive button.

Switch back to Expression Web, activate the Layers panel, and click the closed eye for the ProdSubmenu layer to make it visible again. Switch to the Behaviors panel and configure the behaviors for each button as follows:

1. Select the Software Interactive Button.
2. In the Behaviors panel, select Insert, Change Property.
3. Select the Select Element option.
4. Select div from the Element Type drop-down.
5. Select ProdSubmenu from the Element ID drop-down.
6. Click the Visibility button.
7. Select the Visible option and click OK.
8. Make sure the Restore on Mouseout Event check box is unchecked.
9. Click OK.
10. Change the `onclick` event to `onmouseover`.

Perform the same steps for the Hardware interactive button. After configuring the Hardware button, switch to the Layers panel and make the ProdSubmenu layer invisible again. Then, save and preview your page in the browser. Hover over the Products button to display the Products submenu; then hover over the Software and Hardware buttons. Move the mouse below the Products submenu to ensure that the menu disappears as expected.

One problem still remains. If you hover over the Products button and then move the mouse off the Products button to any other button or off the menu itself without hovering over the submenu, the ProdSubmenu layer remains visible. You actually want the ProdSubmenu layer to disappear when a user hovers over one of the other mainMenu layer buttons or when the user moves the mouse off the menu altogether. To correct that, follow these steps:

1. Switch back to Expression Web.
2. Select the Home button.
3. Switch to the Behaviors panel.
4. Select Insert, Change Property.
5. Select the Select Element option.
6. Select div from the Element Type drop-down.
7. Select ProdSubmenu from the Element ID drop-down.
8. Click the Visibility button.

9. Select the Hidden option and click OK.
10. Make sure the Restore on Mouseout Event check box is unchecked.
11. Click OK.
12. Change the `onclick` event to `onmouseover`.

Make the same behavior configuration change for the Support button. After you've done that, create a new behavior for the mainMenu layer to hide the ProdSubMenu layer as follows:

1. Select the mainMenu layer.
2. From the Behaviors panel, select Insert, Change Property.
3. Select the Select Element option.
4. Select `div` from the Element Type drop-down.
5. Select `ProdSubMenu` from the Element ID drop-down.
6. Click the Visibility button.
7. Select the Hidden option and click OK.
8. Make sure the Restore on Mouseout Event check box is unchecked.
9. Click OK.
10. Change the event from `onclick` to `onmouseout`.

Save your page and browse it to preview the menu. The Products submenu should now work perfectly—it should display and hide itself in response to the correct events.

Configuring the Support Submenu

The next step is to configure the Support submenu. The steps are the same as those you completed for the Products submenu:

1. Make the SupportSubMenu layer visible.
2. Create a behavior for the Support button to display the SupportSubMenu layer in the `onmouseover` event.
3. Create a behavior for the FAQ and Ask Us buttons to display the SupportSubMenu layer in the `onmouseover` event.
4. Create a behavior for the SupportSubMenu layer that hides the SupportSubMenu layer in the `onmouseout` event.

5. Create a behavior for the Products and Contact Us buttons that hides the SupportSubmenu layer in the onmouseover event.
6. Create a behavior for the mainMenu layer that hides the SupportSubmenu layer in the onmouseout event.
7. Hide the SupportSubmenu layer.

Save the page again and preview it. Your dynamic menu should be fully functional at this point.

Z-Order Anomalies

As mentioned in this chapter, the z-order of a layer affects whether a layer appears on top of other layers. However, as in many other areas of web design, z-order doesn't always work the way you might expect.

To fully understand how z-order works in Internet Explorer, you must first understand that two types of elements can be placed on a page: windowed elements and windowless elements. Each of these element types uses an entirely separate z-order hierarchy. Additionally, windowed elements always appear on top of windowless elements. That means a windowless element with a z-order value of 4 appears on top of a windowless element with a z-order value of 3, but all windowed elements appear on top of both windowless elements regardless of z-order.

Why is this important? It's important because some common elements such as drop-down boxes are windowed elements. Therefore, if you have a drop-down box on your page and are expecting a layer that displays part of a menu system to appear on top of the drop-down, you'll be out of luck. The drop-down always appears on top of the menu layer.

To work around this behavior, make sure any windowed elements do not appear close enough to windowless elements so that they overlap. If you absolutely must place two in close proximity, you can create a behavior that hides the windowed element when the windowless element appears.

This page intentionally left blank

USING FORM CONTROLS

Understanding HTML Forms

In the early days of the Internet, communication was strictly one-way. You visited a site and pulled down information to your browser. A few years later, HTML forms were introduced, and for the first time, web designers had a means of collecting information from those visiting their sites. Nowadays, almost every site uses HTML forms in one way or another.

An HTML form consists of one or more form controls such as text boxes, drop-down lists, or buttons. After information is selected or entered into the form, it is generally submitted to the web server for processing.

➔ *For more information on manipulating a page using JavaScript, see Chapter 22, “Client Scripting.”*

To process form data on a web server, code that runs on the web server must be written. Technologies such as PHP, ASP, and ASP.NET are often used to process that form data. Expression Web allows you to connect an HTML form to server-side code for processing, but it also provides a few ways for you to process a form without writing any code at all.



caution

Unless precautions are taken, providing forms on a site can be a security risk. In most cases, data provided by users via a form will be manipulated using code on the web server. It is possible for a malicious user to provide data in a form that can be damaging to your site or the web server on which it runs. When using forms, always make sure you validate the data that users are providing and ensure that it's valid and in the expected format.

- ➔ For more information on using ASP.NET to work with form data, see Chapter 25, “Using Standard ASP.NET Controls.”
- ➔ For more information on using PHP to work with form data, see Chapter 32, “Using PHP.”

Using Form Controls in Expression Web

Form controls in Expression Web are located in the Toolbox. As shown in Figure 24.1, 16 form controls are available.

note

Some sites use forms to allow users to manipulate an existing page. In those cases, the web server doesn't process the data. Instead, the site designer typically uses JavaScript to determine what data was entered and what choices were made and then responds appropriately by changing the page's content, by requesting a different page.

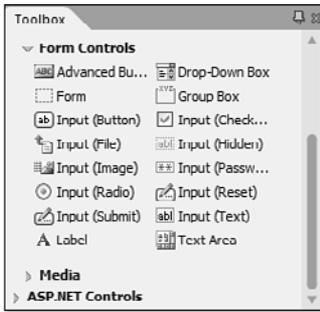


Figure 24.1 The Toolbox contains 16 form controls, making it easy to create HTML forms.

Table 24.1 lists the form controls that are available.

Table 24.1 Form Controls

Control	HTML Tag	Details
Advanced Button	<button>	Enables you to embed HTML content inside a button. Some browsers don't display this control correctly.
Drop-Down Box	<select>	Enables you to create a drop-down box of preset choices from which a user can make a selection.
Form	<form>	The container control for all other form controls. This control also provides access to the form handlers provided by Expression Web.
Group Box	<fieldset>	Segments part of a form into a separate container. This is often used to avoid confusion on large forms and to make forms more accessible.

Control	HTML Tag	Details
Input (Button)	<input>	Creates a generic HTML button that can be used for JavaScript and others. While buttons are most often used for submitting or resetting a form, this control is a generic button whose purpose is defined by the designer.
Input (Checkbox)	<input>	Creates a standard HTML check box. The check box is typically used as a true/false control.
Input (File)	<input>	Creates a text box with a Browse button so a user of the form can browse for a file on his or her local computer to submit along with the form.
Input (Hidden)	<input>	Creates a text box that is not visible to the user. Hidden text boxes enable you to include data in a form that is not visible to the user.
Input (Image)	<input>	Enables you to use an image as a button.
Input (Password)	<input>	Creates a text box that displays dots or asterisks in place of text when data is entered into it. These are commonly used for password fields.
Input (Radio)	<input>	Creates a radio button. A form that uses this control will have at least two radio buttons grouped together, and only one of those radio buttons can be selected at a time.
Input (Reset)	<input>	Creates a button that resets the form. Resetting a form causes all form fields to revert to the values they had when the page was originally loaded.
Input (Submit)	<input>	Creates a typical button that submits the form.
Input (Text)	<input>	Creates a standard HTML text box.
Label	<label>	Associates a text label with a form control. This control is typically used for accessibility.
Text Area	<textarea>	Creates a multirow and/or multicolumn text box.

Expression Web enables you to easily save form results to a file, send results to an email, or save form results into a database. We'll cover each of these options in detail.

Creating a Form

The obvious reason for including an HTML form on a page is so you can collect data from visitors to your site. After you've collected that data, Expression Web provides the capability of storing it in a file (such as a text file, an XML file, or an HTML file), sending the results to a specific email address, or saving the results to a database.

Let's create a simple form that enables us to collect site feedback from users. To do that, create a new site or open an existing site.



note

To save results to a database, you need to use an ASP page. However, the first example requires a static HTML page, so just create an HTML page at this point.

The site can be a disk-based site, but to test the form's functionality, you need to publish your site to a server running the FrontPage Server Extensions.

➡ *For more information on publishing a site to a web server with the FrontPage Server Extensions, see Chapter 14, "Publishing a Site."*

1. Create a new HTML page.
2. From the Form Controls section of the Toolbox, insert a new Form control.
3. Place the insertion point inside the form.
4. Insert a new table on the page. Set the table width to 550 and select the In Pixels radio button. Leave all other settings at their defaults.

➡ *For more information on adding and configuring tables, see Chapter 5, "Using Tables."*

5. Place the insertion point in the upper-left cell of the table.
6. Enter the text **Enter your e-mail:**.
7. Press Tab to move to the next cell to the right.
8. From the Form Controls section of the Toolbox, add an Input (Text) control.
9. With the Input (Text) control still selected, locate the Name attribute in the Tag Properties panel.

➡ *For more information on using the Tag Properties panel, see Chapter 7, "Editing Tag Properties."*

10. Change the Name attribute to Email.
11. Locate the Style attribute in the Tag Properties panel.
12. Click the button in the value column of the Style attribute to display the Modify Style dialog.
13. Select the Position category in the Modify Style dialog.
14. Set the Width to 200 px (shown in Figure 24.2) and click OK.

➡ *For more information on using the Modify Style dialog, see Chapter 18, "Managing CSS Styles."*

15. Place the insertion point in the leftmost cell of the second row.
16. Enter the text **Enter comment:**.
17. Press Tab to move to the next cell to the right.

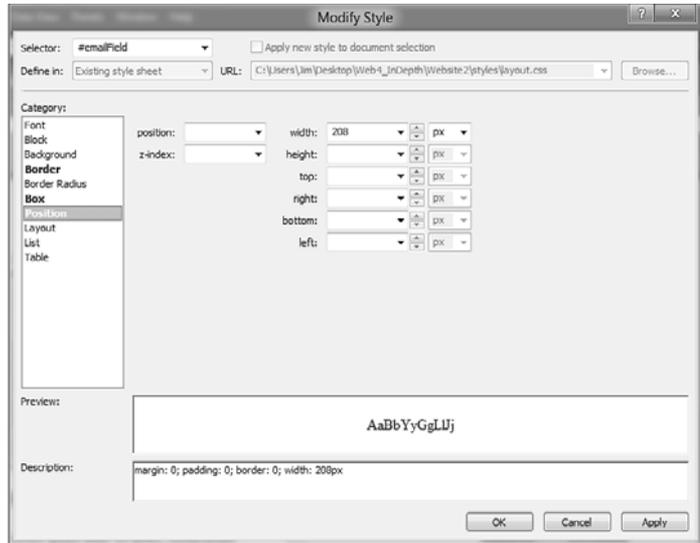


caution

Many hosts no longer offer the FrontPage Server Extensions, and many other hosts are in the process of phasing them out. Before you invest a lot of time in building forms that require the FrontPage Server Extensions, you may want to investigate other technologies such as ASP.NET or PHP.

Figure 24.2

The Modify Style dialog is a quick way to change the appearance of a form element.



18. From the Form Controls section of the Toolbox, insert a Text Area control.
 19. With the Text Area control still selected, locate the Name attribute in the Tag Properties panel.
 20. Change the Name attribute to **Comment**.
 21. Locate the Style attribute in the Tag Properties panel.
 22. Click the button in the value column of the Style attribute to display the Modify Style dialog.
 23. Select the Position category in the Modify Style dialog.
 24. Enter a value of **375 px** for the width and **100 px** for the height and click OK.
 25. Select Table, Insert, Row Below to insert a new table row. (If the Insert menu item is disabled, click once on the Text Area control to select it first.)
 26. Place the insertion point in the bottom-right corner cell.
 27. From the Form Controls section of the Toolbox, insert a new Input (Submit) control.
 28. Move off the button (into the cell) and press Ctrl+R to right-align the button in the table cell.
- Your page should now look like Figure 24.3.

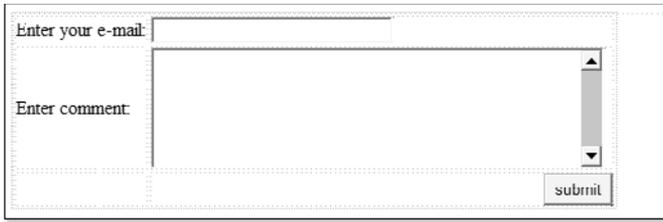
A screenshot of a simple web form. It features a text input field labeled "Enter your e-mail:" and a larger text area labeled "Enter comment:". A "submit" button is located at the bottom right of the form.

Figure 24.3
This simple form will be used to collect feedback from site visitors.

Saving Form Results to a File or Email

By default, a form created in Expression Web does nothing. When you browse the form and submit it, nothing happens. Let's add some functionality to our feedback form by configuring it to save results to a file and an email address.

Right-click the form you created previously and select Form Properties from the menu to display the Form Properties dialog, as shown in Figure 24.4.

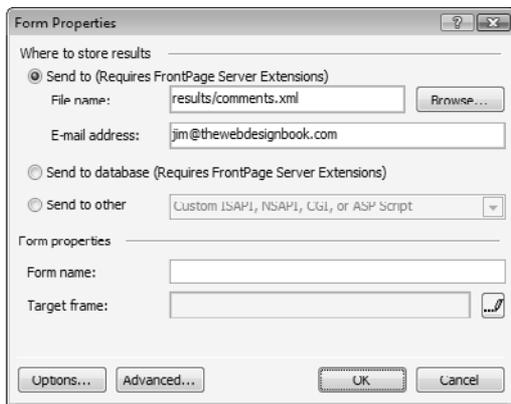
A screenshot of the "Form Properties" dialog box. The "Where to store results" section has three radio button options: "Send to (Requires FrontPage Server Extensions)" (selected), "Send to database (Requires FrontPage Server Extensions)", and "Send to other". The "Send to" option is configured with "File name: results/comments.xml" and "E-mail address: jim@thewebdesignbook.com". The "Send to other" option is set to "Custom ISAPI, NSAPI, CGI, or ASP Script". The "Form properties" section includes "Form name:" and "Target frame:" fields. Buttons for "Options...", "Advanced...", "OK", and "Cancel" are at the bottom.

Figure 24.4
Form handlers are configured in the Form Properties dialog. Most options here require the FrontPage Server Extensions.

Three methods of saving form results are available in the Form Properties dialog:

- **Send To**—This option enables you to save form results to a file and/or send them to an email address.
- **Send to Database**—This option enables you to save form results to a database. You can also have Expression Web create a Microsoft Access database for you.
- **Send to Other**—This option enables you to use a custom form handler.

caution

If you change the selected method in the Form Properties dialog, any options configured will be reset to the defaults and cannot be retrieved. If you inadvertently change the method and lose your configuration, click Cancel in the Form Properties dialog. You can then go back to the Form Properties dialog and your previous options will be restored.

You can configure options for each of these methods using the Options button shown previously in Figure 24.4. The options available differ depending on the method you choose.

To configure a form to save results to a file and email, do the following:

1. Select the Send To radio button (the first method).
2. Enter `_private/results.txt` in the File Name text box.
3. Enter your email address in the Email Address text box.
4. Click the Options button to display the Saving Results dialog, as shown in Figure 24.5.

When using the Send To method of saving form results, the Saving Results dialog contains a series of four tabs.



caution

To save results to an XML file, the remote server must be running Windows SharePoint Services. Because Windows SharePoint Services are typically used in a corporate environment, most shared hosting companies don't offer them. If you're running your site in a shared hosting environment, check with your host to be sure.

Figure 24.5

The Saving Results dialog provides many options for saving form results.



File Results Tab

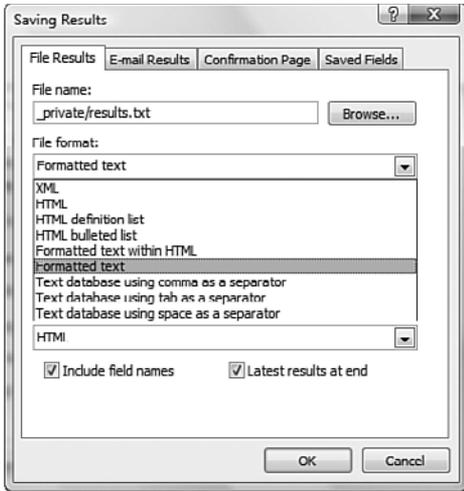
The File Results tab provides options for saving results to a file. In addition to specifying the filename, you can choose from many file types, as shown in Figure 24.6. If you'd like to save results to two different file formats, you have the option of choosing a second file.

Form field names can be included in your form results by checking the Include Field Names check box. If you choose not to include field names, only the data entered into the form field will be saved. When saving results to a text database file format, field names will be saved as a header row in the results file.



tip

The text database formats are useful if you intend to import the form results into a database or spreadsheet later. To save your form results to a database directly, use the Send To Database form handler provided by Expression Web.

**Figure 24.6**

You have many file types from which to choose when saving results.

By default, the latest form results are appended to the bottom of the file to which they are saved. Some file types (XML and HTML file types) allow you to choose between appending results to the file or adding them to the top of the file via the Latest Results at End check box. If the check box is checked, results are appended to the bottom of the file; otherwise, they are added to the top of existing results.



Latest Results at End Check Box Disabled

The following file types do not allow you to store form results at the top of the file:

- Formatted Text
- Text Database Using Comma as a Separator
- Text Database Using Tab as a Separator
- Text Database Using Space as a Separator

If you have selected one of these file types, the Latest Results at End check box will be checked and disabled. Select a different file type.

Email Results Tab

The Email Results tab provides configuration options for email results (see Figure 24.7).

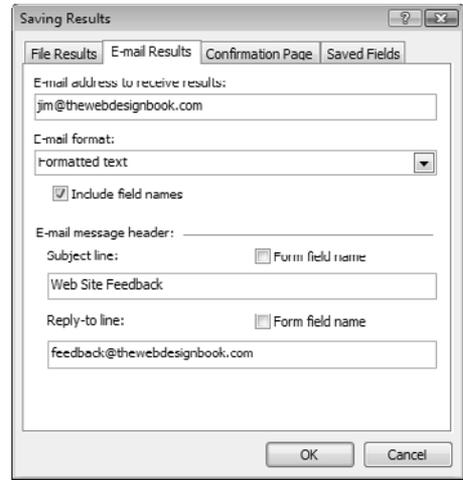
Enter the email address you would like form results sent to in the Email Address to Receive Results text box. It would be nice to



caution

Carefully check the email address you enter. Expression Web does not check to be sure you enter a valid email address.

Figure 24.7
Email options are configured in the Email Results tab.



also be able to send a copy of the results to the user filling out the form, but there isn't an automated way of doing that.

Choose the format of the email in the Email Format drop-down. The formatting options are the same as those presented in the File Results tab. If you want field names to be included in the email, check the Include Field Names check box.

To specify a subject for the email, enter it into the Subject Line text box. You can also use the value of a form field for the subject. For example, if you had a drop-down field named Subject in your form that allowed users to select the subject of their comment, you could use the value from the drop-down as the email subject by checking the Form Field Name check box and typing **Subject** in the Subject Line text box. To configure a form with this capability, follow these steps:

1. Create a new page or open an existing page.
2. Add a new Drop-Down Box control from the Form Controls section of the toolbox.
3. Double-click the Drop-Down Box control to display the Drop-Down Box Properties dialog.
4. Enter **Subject** in the Name text box.
5. Click the Modify button to modify the choice in the Drop-Down Box.
6. Enter **Web Site Feedback** in the Choice text box.



caution

Make sure that you don't use the DropDownList control in the ASP.NET section of the toolbox. You want the HTML control instead.



tip

If you already entered an email address in the Form Properties dialog, that email address will already appear in the Email Address to Receive Results text box.

7. Select the Selected radio button.
8. Click OK to dismiss the Modify Choice dialog.
9. Click the Add button to add a new choice.
10. Enter **Company Feedback** in the Choice text box.
11. Click OK to dismiss the Modify Choice dialog.
12. Click OK to dismiss the Drop-Down Box Properties dialog.

You can now check the Form Field Name check box next to the Subject Line and specify **Subject** as the Subject Line in the Saving Results dialog.

The Reply-To Line text box allows you to specify the email address used when a recipient of the email responds. You can use a form field value for the Reply To address by checking the Form Field Name check box and entering the name of the form field in the Reply-To Line text box.



Asked to Remove Email Recipient

If you click OK in the Form Properties dialog after configuring your form to send to an email address you might be prompted that the server isn't configured for email and be asked to remove the email recipient. If the site that is open in Expression Web is a disk-based site or if the server it is on is not configured to send email via the FrontPage Server Extensions, you will receive this warning.

If you're working with a disk-based site, email should work after you publish the site to a server running the FrontPage Server Extensions. If you are using a site on a web server, check with the hosting company or server administrator. They'll need to configure email support before you can send form results to an email address.

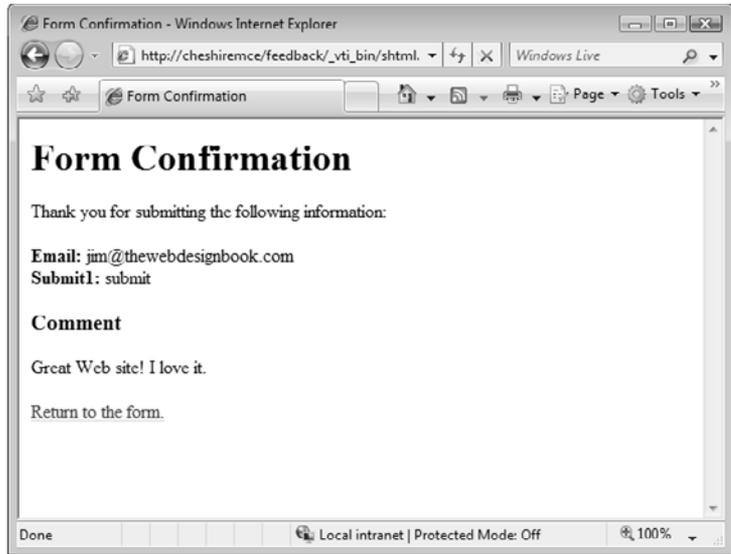
Confirmation Page Tab

When a user submits data in a form, he or she sees a default confirmation page, as shown in Figure 24.8. Although it's certainly a functional confirmation page, it's not very attractive and likely won't match the layout or appearance of your site.

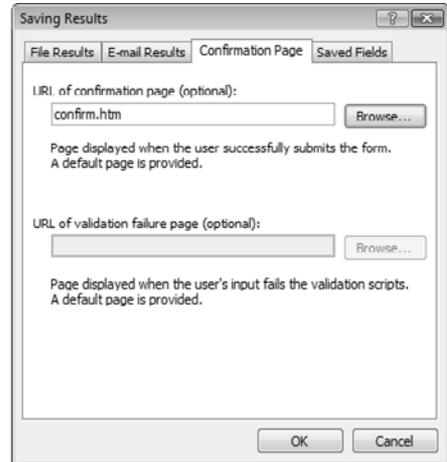
If you have a page in your site that you'd prefer to use as a confirmation page, you can specify the URL in the URL of Confirmation Page text box in the Confirmation Page tab (see Figure 24.9). A custom confirmation page gives a more professional appearance by matching the layout and design of your site. However, the confirmation page must be a static HTML page, which prevents you from displaying any customized content to the user such as confirmation of the data submitted.

Figure 24.8

A default confirmation page is provided, but it's generic and not very attractive.

**Figure 24.9**

The Confirmation Page tab enables you to specify a custom confirmation page and a validation failure page.



If validation has been enabled on any of the form fields in your form, you can also specify the URL of a page to be displayed when validation fails. The validation failure page will be displayed only if validation takes place on the server.

note

If you are saving the form results into a file and you plan to import the data into a database at a later time, you might want to prevent the Submit button from being saved so as not to pollute your data.

**tip**

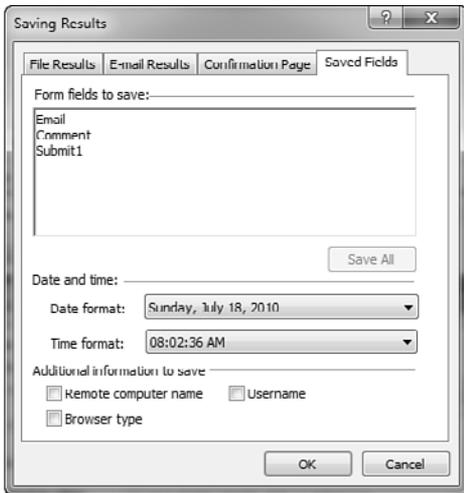
The username will be saved only if the user is required to log in before submitting the form. If the user is browsing anonymously, the username will not be available.

**Cannot Enter Validation Failure Page**

If none of the form fields in your form is configured to use validation, you'll be unable to specify a validation failure page. Configure validation on your form first and you'll then be able to specify a validation failure page.

Saved Fields Tab

By default, Expression Web saves all form fields when a form is submitted. You can override this behavior using the Saved Fields tab, as shown in Figure 24.10.

**Figure 24.10**

The Saved Fields tab controls how much data gets saved when a form is submitted.

As you can see in Figure 24.10, Expression Web is saving not only the two text fields from the form, but also the Submit button. If you want to remove the Submit button from the saved fields, simply delete Submit1 from the list of saved fields and click OK.

**caution**

The database created is a Microsoft Access 2002 database. If you use a later version of Microsoft Access to edit the database, do not save it or convert it to the later version. Doing so will prevent it from working correctly with your forms.

You also have the choice of saving the date and time a form was submitted by selecting a date format and a time format from the appropriate drop-downs. You can also choose to save the remote computer name, browser type, or username by checking the corresponding check box.

Saving Form Results to a Database

If you'd prefer to keep form results stored in a database, Expression Web provides you with the ability to send form results directly into a database when a form is submitted using the Send To Database form handler, as shown in Figure 24.11.

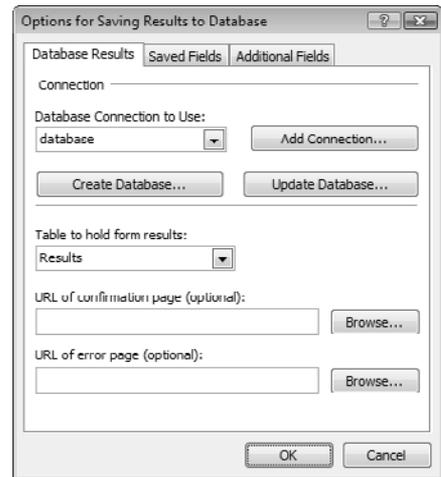


caution

When the FrontPage Server Extensions create the database for your form, they also add a `global.asa` and `_fpclass/fpdbform.inc` file to your site. Do not remove these files; otherwise, it will break your form.

Figure 24.11

The Send to Database form handler is a simple way to save form results directly into a database.



Let's configure our HTML form to save results into a Microsoft Access database:

1. Open `default.htm` and save it as `database.asp`.
2. Right-click the form and select Form Properties.
3. Select the Send To Database option.
4. Click the Options button to display the Options for Saving Results to Database dialog, shown previously in Figure 24.11.

When the dialog is first displayed, no database connections are available in the Database Connection to Use drop-down. You can create a new connection by clicking the Add Connection button,



tip

When saving form results to a database, the confirmation page must be an ASP page.



note

You won't be able to access the Saved Fields or the Additional Fields tabs until you've created a database to store your form results.

or you can click the Create Database button to have the FrontPage Server Extensions create a Microsoft Access database along with a connection to that database.

In this case, we want to let the FrontPage Server Extensions create a database for us, so click the Create Database button to generate a Microsoft Access database with a field for each form field.

After the database has been created, the Database Connection To Use and Table To Hold Form Results drop-downs will be populated accordingly. You also have the option of selecting a custom confirmation page and an error page. The error page will be displayed if there is an error while trying to save results to the database.

When a database gets created by the FrontPage Server Extensions, the form fields are mapped to appropriate fields in the database. If you want to review or modify the mappings, click the Saved Fields tab. As shown in Figure 24.12, each form field is currently mapped to a corresponding database field. To modify a mapping, select the form field in the list and click Modify. You can remove a mapped field by selecting the field in the list and clicking the Remove button.



Figure 24.12

Form fields are mapped to database fields automatically. You can modify or remove these mappings easily.

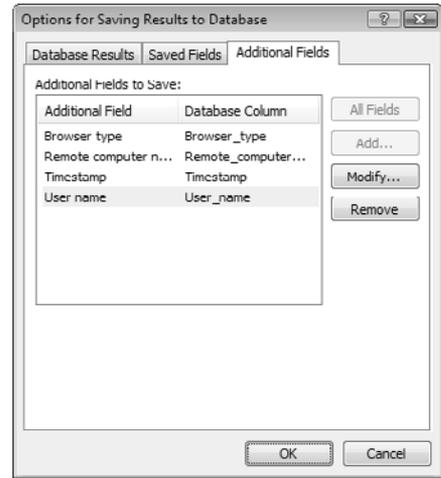


Form Fields Not Mapped

If you create a database and then click Cancel to exit the Form Properties dialog, the form fields will be unmapped. You need to manually map them before you can continue.

The Additional Fields tab provides access to several additional fields that can be saved with your form, as shown in Figure 24.13. These fields are also mapped to the appropriate database columns automatically.

Figure 24.13
Expression Web automatically saves additional fields to your database.



Updating a Database with New Fields

If you add new fields to your form after the database has been created, you'll need to add the new fields to the database. Expression Web does this for you automatically. Follow these steps:

1. Open `database.asp`.
2. Place the insertion point in the table row where the Submit button is located.
3. Select Table, Insert, Row Above to insert a new table row.
4. In the left cell of the new row, enter the text **Your site:**.
5. In the rightmost cell of the new row, insert a new Input (Text) form control.
6. Using the Tag Properties panel, set the width of the control to **350 px** and the name to **URL**.
7. Right-click the form and select Form Properties.
8. Click the Options button.
9. Click Update Database.

note

For details on how to manipulate cookies with JavaScript, read *Special Edition Using JavaScript* from Que Publishing.

tip

A hidden form field doesn't necessarily have to be configured using client script. You can hard-code a value in a hidden form field as well.

Expression Web automatically adds the new field to the database and configures the mappings for you.

Hidden Form Fields

All the forms we tested in this chapter dealt with form data that was visible to the user of the form. However, sometimes you might want to include data with your form that you don't want users of the form to see. For example, suppose you have some client-side script that determines whether cookies are enabled in the user's browser. When a user submits your form, you also want to submit True if cookies are enabled or False if they're not. However, having a form field on your form that says True or False wouldn't be appropriate.

In such cases, a hidden form field is a great solution. After you determine whether cookies are enabled, you can use a client script to set the value of the hidden form field. That value will then be submitted along with the other form fields in your form.

A hidden form field is just like a text input field except it's not visible in the browser. The following HTML code creates a hidden form field called `cookies`:

```
<input type="hidden" name="cookies" value="false" />
```

To create a hidden form field in Expression Web:

1. Right-click the form and select Form Properties.
2. Click the Advanced button to display the Advanced Form Properties dialog.
3. Click the Add button to add a new hidden form field.
4. Enter a name and value for the form field.

Figure 24.14 shows a hidden form field called `cookies` configured in the Advanced Form Properties dialog.

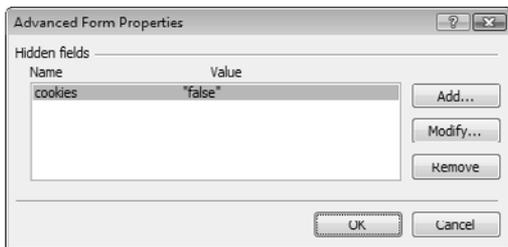


Figure 24.14 Hidden form fields are configured using the Advanced Form Properties dialog.

USING STANDARD ASP.NET CONTROLS

ASP.NET: A Brief Introduction

Today's sites bear little resemblance to the sites that appeared when the Web first began. After all, the concept of regular people like you and me actually *owning* a computer never occurred to the scientists who were devising what J. C. K. Licklider of MIT referred to as the "Galactic Network." Instead, they used the new Internet as a means to share research between scientific and educational institutions. The sole emphasis was on connecting computers.

Nowadays, we don't give a second thought to the connectivity of computers. Just about everyone in the modern world has Internet access, and many of those people have high-speed connections. The Internet revolution is among us, and the ubiquitous connectivity we now enjoy has transformed the Internet into something entirely different from the original intent.

We now expect every company and every organization to have a site. We want to pay our bills online. We want to be able to change our cable or satellite service online. We want to get details on our cell phone service online. It has become obvious that static pages no longer fill the bill.

To meet all these expectations, companies must implement some means of accessing their back-end data systems via the Internet. Many technologies are available today for doing just that, and Microsoft's latest offering in that arena is ASP.NET.

If you haven't used ASP.NET yet, you're in for a real treat! The folks on the ASP.NET product team have put a lot of work into making it approachable for developers of all levels. They've also packed it with great features

that web developers have been screaming for. Expression Web fully supports the rendering of ASP.NET controls, and it also supports configuring properties of controls, including IntelliSense for ASP.NET controls in Code View.

Creating ASP.NET Pages

It should be obvious that before you can use ASP.NET controls, you need to create an ASP.NET page. ASP.NET pages are called Web Forms, and they have an .aspx file extension. When you create an ASP.NET Web Form, you also choose a language for the page. Expression Web provides the choices of C# (pronounced *C-sharp*), and Visual Basic (VB).



note

This book does not teach you how to program in ASP.NET because that is outside the scope of a book on Expression Web. If you want to learn how to fully utilize the power of ASP.NET as a programmer, read *Sams Teach Yourself ASP.NET 4 in 24 Hours: Complete Starter Kit* from Sams Publishing (ISBN 0132171686).

Choosing an ASP.NET Language

Developers who are new to ASP.NET might have a difficult time deciding on a language choice. As far as functionality is concerned, there is little difference. The choice you make will be primarily based on your comfort with the syntax.

C# is a language that Microsoft introduced in 2002. The syntax of C# is most familiar to developers who have experience in C, C++, or JavaScript. It is also case-sensitive, which may be a bit of an annoyance to VB developers.

VB shares syntax with VBScript and the legacy VB 6 and earlier. However, the commonalities stop there. This VB is what was previously referred to as VB.NET. However, the syntax and semantics will be familiar to anyone who has worked with VB.

You'll often see developers debating which language is better. At the end of the day, there is little difference between the capabilities of each, and in almost all cases, one will not afford you any operational benefit over another. Choose the language that you feel most comfortable with and you'll be fine.

To create a new ASP.NET Web Form in Expression Web, do the following:

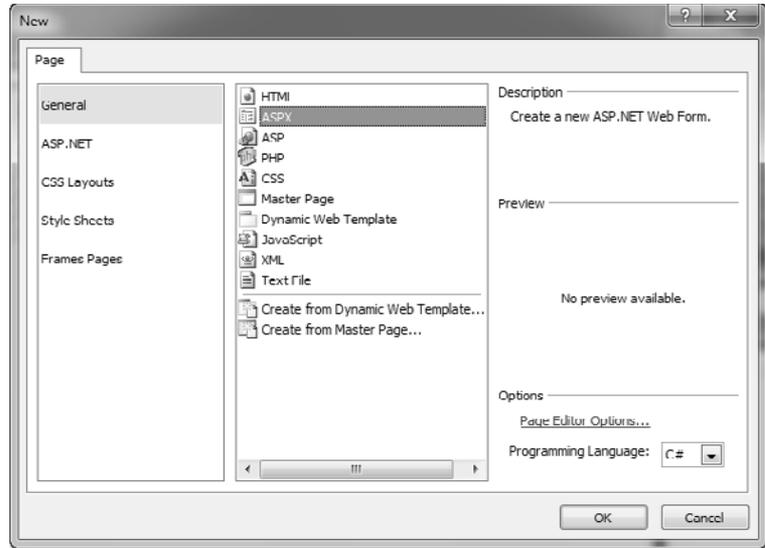
1. Click File, New, Page.
2. In the New dialog, select ASPX from the list of file types as shown in Figure 25.1.
3. Select your language in the Programming Language dropdown.
4. Click OK.



tip

Expression Web doesn't provide very good support for writing ASP.NET code. It does have great support for rendering ASP.NET controls, however. What you'll likely want to do is design your layout and set basic control properties in Expression Web, and then use Visual Studio or Visual Web Developer Express Edition to write your C# or VB code.

Figure 25.1
Create a new ASP.NET Web Form using the New dialog.



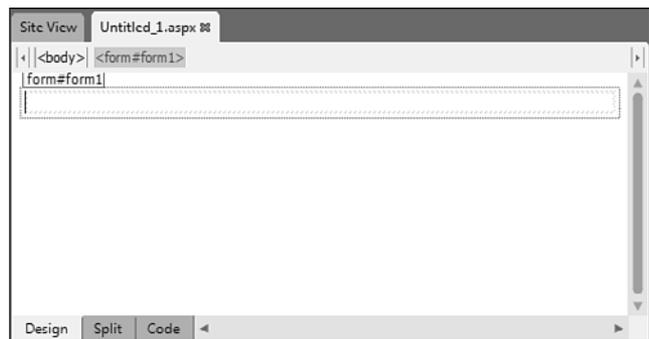
After your ASP.NET Web Form has been created, you'll see an empty form, as shown in Figure 25.2. Any ASP.NET controls you add to your page will be inserted into this form.

You're now ready to start adding some ASP.NET controls to your page, so let's have a look at the Standard ASP.NET controls available.

note

If you select File, New, ASPX instead of File, New, Page, you won't have the option of selecting a language. Instead, Expression Web chooses the language that you last used, or C# if it's the first time you've created an ASP.NET page.

Figure 25.2
When an ASP.NET Web Form is created, it automatically has a form within it into which all ASP.NET controls will be inserted.



The Basics of ASP.NET Controls

All ASP.NET controls are available via the Toolbox in Expression Web. The Toolbox is divided into two sections, as shown in Figure 25.3: HTML and ASP.NET Controls.



Figure 25.3

The Toolbox in Expression Web makes it easy to add ASP.NET controls to your page.

The ASP.NET Controls section of the Toolbox has several subsections. We'll cover the Standard controls in this chapter and other controls in subsequent chapters. Following is a brief description of the sets of controls in the ASP.NET section of the Toolbox:

- **Standard**—Standard ASP.NET controls such as radio buttons, text boxes, drop-downs, and so on
- **Data**—Controls that are designed to retrieve and display data on an ASP.NET Web Form
- **Validation**—Controls that allow you to configure form field validation on your ASP.NET Web Forms
- **Navigation**—Controls that are specifically designed for navigating in a site
- **Login**—Controls that allow you to easily create a membership system for an ASP.NET site
- **WebParts**—Controls that allow for the insertion and manipulation of ASP.NET Web Parts
- **AJAX**—Controls that use Microsoft's ASP.NET Ajax

➔ *For more information on ASP.NET Data controls, see Chapter 34, "Displaying and Editing Database Data with ASP.NET."*

➔ *For more information on ASP.NET Validation controls, see Chapter 29, "Form Validation Using ASP.NET."*

➔ *For more information on ASP.NET Navigation controls, see Chapter 26, "Using ASP.NET Navigation Controls."*

➔ *For more information on ASP.NET Login controls, see Chapter 28, "Developing a Login System Using ASP.NET."*

➔ *For more information on ASP.NET Web Parts, see Chapter 30, "Using ASP.NET Web Parts."*

➔ *For more information on ASP.NET Ajax Extensions, see Chapter 31, "Using ASP.NET Ajax."*

Many of the Standard ASP.NET controls are similar to their HTML counterparts. For example, the ASP.NET `TextBox` control is functionally the same as the HTML text input control, and the ASP.NET `DropDownList` control is functionally the same as the HTML Drop-Down Box control. Other ASP.NET controls (such as the `Calendar` control) have no HTML control counterpart.

When you insert ASP.NET controls into a page, Expression Web generates the ASP.NET code necessary for ASP.NET to render the controls. For example, when you insert a `TextBox` control onto an ASP.NET Web Form, the following code is added in Code View:

```
<asp:TextBox runat="server" id="textbox1"></asp:TextBox>
```

This is obviously not normal HTML code, and a web browser won't understand this code. Therefore, when this page is browsed, the ASP.NET runtime environment renders the code as an HTML `<input>` tag, similar to the following:

```
<input name="textbox1" type="text" id="textbox1" />
```

A `TextBox` control is a simple example of how ASP.NET renders controls. Not all controls are this simple. For example, an ASP.NET `Calendar` control looks simple within Code View in Expression Web:

```
<asp:Calendar runat="server" id="calendar1"></asp:Calendar>
```

But when the page is browsed, ASP.NET builds some pretty complex HTML to render the `Calendar` control correctly in the web browser. As a matter of fact, so much code gets rendered by ASP.NET for a control like the `Calendar` that it would take more than a page of this book to show it all to you. By using ASP.NET to implement a calendar on your page, you can avoid having to write a large amount of code yourself.

**tip**

We've already covered the Tag Properties pane in Chapter 7, "Editing Tag Properties," but just in case you skipped that part, I'll give you a tip here: When a property description appears as bolded in the Tag Properties pane, it means the value of that property has been changed from the default value.

Understanding Control Properties

ASP.NET controls use control properties to affect their behavior and appearance. Control properties can be specified either by selecting the control and using the Tag Properties pane to manipulate the properties or by changing the properties directly in Code View.

- ➔ *For more information on using the Tag Properties pane, see Chapter 7, "Editing Tag Properties."*
- ➔ *For more information on using Code View in Expression Web, see Chapter 4, "Using Page Views."*

Another Method of Accessing Control Properties

There is actually a third way to access the properties of an ASP.NET control—programmatically. However, in this book, we are going to work with changing properties via the Tag Properties pane because ASP.NET allows you to take advantage of most controls without writing any code.

Let's add an ASP.NET control to a page and set some properties on that control:

1. Launch Expression Web and create a new disk-based site.
2. Click File, New, Page and select ASPX from the list of file types to create a new ASP.NET Web Form. Click OK.
3. Save the page as `default.aspx`.
4. Scroll down to the `Label` control in the Standard ASP.NET controls section of the Toolbox.
5. Double-click the `Label` control in the Toolbox to add it to the Web Form. Your page should now look like Figure 25.4.
6. Save and browse the page to ensure that the `Label` control displays correctly. You should see the word "Label" displayed in the browser.

➔ *For more information on creating sites in Expression Web, see Chapter 2, "Creating, Opening, and Importing Sites."*



caution

There is a `Label` control in the HTML Form Controls section of the Toolbox. Make sure you don't add it. Instead, you want to add the `Label` control in the Standard ASP.NET Controls section.



note

You might have found it strange to create a new disk-based site in step 1 even though this is an ASP.NET site. Expression Web uses the Microsoft Expression Development Server to browse ASP.NET pages in a disk-based site. Using this method is the easiest way to design and test ASP.NET sites.



Figure 25.4

The ASP.NET `Label` control should be inside the form.



ASP.NET Controls Do Not Appear in the Browser

For ASP.NET pages to be rendered as HTML code that the browser can understand, they must be handled by the ASP.NET engine; otherwise, the browser will see tags it doesn't understand and will display nothing at all. To resolve this problem, make sure your ASP.NET page has an .aspx file extension. That will cause it to be executed by ASP.NET so it displays correctly. You also need to make sure that your host supports ASP.NET.



XML Error When Browsing ASP.NET Page

The code that makes up ASP.NET controls in your Web Form is actually composed of XML tags. However, when things are working as they should, ASP.NET transforms those tags into regular HTML code.

If you see an error that the XML page cannot be displayed when browsing your page, it usually means that you are trying to browse the page using a file path such as C:\websites\page.aspx. ASP.NET pages must be browsed on a web server for them to work correctly.

Expression Web ships with the Microsoft Expression Development Server, which you can use to browse your ASP.NET pages when your site is a disk-based site. To make sure the Microsoft Expression Development Server is being used:

1. Click Site, Site Settings and then click the Preview tab.
2. The Preview Using Web Site URL option should be selected.
3. The Use Microsoft Expression Development Server check box should be checked.



For more information on the Microsoft Expression Development Server, see Chapter 33, "Using the Microsoft Expression Development Server."

Now you have a Label control that displays the word "Label" on the page. The next step is to manipulate the properties of the control to make it useful.

1. Select the Label control on the Web Form.
2. In the Tag Properties pane, change the Text property of the Label control to An ASP.NET Label, as shown in Figure 25.5.

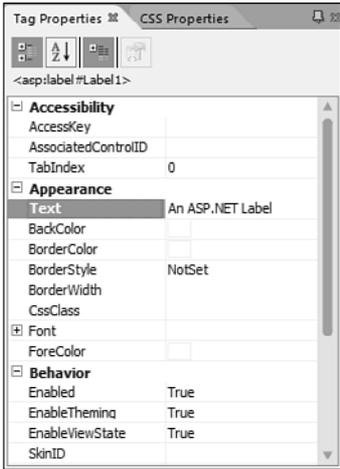
Your label should look like Figure 25.6. When you browse this page in a browser, it will now display the text "An ASP.NET Label."



tip

You might have noticed that Expression Web allows you to place ASP.NET controls only inside the form that is automatically inserted onto the page when it's created. All ASP.NET controls must be placed within a form that is configured to run on the server. Expression Web prevents you from shooting yourself in the foot by not allowing you to place ASP.NET controls outside a form. It also re-creates the form automatically if you accidentally delete it.

Now let's have a look under the covers and see exactly what happened in the code for `default.aspx` when we changed the `Text` property. Switch over to Code View and you'll see that the `<asp:Label>` tag has a new `Text` attribute with a value of `An ASP.NET Label`, as shown in Figure 25.7.

**Figure 25.5**

The Tag Properties pane provides easy access to an ASP.NET control's properties.

**Figure 25.6**

The ASP.NET Label control now has a more useful presentation thanks to a change in the `Text` property.

Specifying property values inside the control's tag in this manner is referred to as declarative syntax. As it turns out, ASP.NET allows you to do some powerful things using declarative syntax, as you'll see later when we get into accessing data with ASP.NET.

Now that you have a general idea of how to set properties for ASP.NET controls, let's dig deeper into the Standard ASP.NET controls.

Figure 25.7

The Label control's tag in Code View now has a Text property set to An ASP.NET Label.

```

Site View | default.aspx*
<body>
  7 <title>Untitled 1</title>
  8 </head>
  9
 10 <body>
 11
 12 <form id="form1" runat="server">
 13   <asp:Label id="Label1" runat="server" Text="An ASP.NET Label">/asp:Label>
 14 </form>
 15
 16 </body>
 17
 18 </html>
 19
Design | Split | Code

```

An Overview of the Standard ASP.NET Controls

Table 25.1 lists all the Standard ASP.NET controls in the Toolbox.

Table 25.1 The ASP.NET Standard Controls

Control Name	Description
AdRotator	Allows for rotating banner ads
BulletedList	Provides programmatic access to a bulleted list
Button	A basic button
Calendar	Allows for easy date selection/display
CheckBox	A basic check box
CheckBoxList	A group of associated CheckBox controls
ContentPlaceHolder	Used with ASP.NET Master Pages
DropDownList	A basic drop-down list
FileUpload	A text box and Browse button for file uploads
HiddenField	A basic hidden form field
Hyperlink	Renders as a typical hyperlink
Image	Displays an image in the browser
ImageButton	Uses an image as a button
ImageMap	Allows for the creation of image maps
Label	Displays static text
LinkButton	A button or link that runs code on the server
ListBox	A multiline text box

Table 25.1 Continued

Control Name	Description
Literal	A placeholder for HTML content
Localize	A placeholder for localized text
MultiView	A container for multiple View controls
Panel	A container for multiple controls
PlaceHolder	A placeholder for dynamically added controls
RadioButton	A basic radio button
RadioButtonList	A list of RadioButton controls
Substitution	Allows part of a page to be exempt from caching
Table	A basic table
TextBox	A basic text box
View	A control in a MultiView that contains other controls
Wizard	A step-based wizard control
Xml	Allows for the display of XML data

➔ *For more information on ASP.NET Master Pages, see Chapter 27, “Using ASP.NET Master Pages and User Controls.”*

➔ *For more information on XML data, see Chapter 34, “Displaying and Editing Database Data with ASP.NET.”*

We won't go into detail on these basic controls, but the sections that follow cover the more complex ones. In these sections, we discuss enough of the general details to give you a good base knowledge to use any of the ASP.NET controls available in Expression Web.

The AdRotator Control

Banner advertisements aren't just for the big-name sites. The entrepreneurial nature of the Internet has made them popular with the little guys as well. Many e-commerce sites on the Internet offer affiliate programs that pay you a commission on referred sales. To effectively take advantage of such programs, you need to implement some sort of advertising on your site, and banner ads have consistently proven to be an effective choice.

The AdRotator control uses an XML file called an advertisement file to feed it information to display banners and links. This XML file must be in the format shown in Listing 25.1.

Listing 25.1 XML Coding for AdRotator Control

```

1 <?xml version="1.0" encoding="utf-8" ?>
2 <Advertisements>
3   <Ad>
4     <ImageUrl>http://www.site.com/banners/banner2.jpg</ImageUrl>
5     <NavigateUrl>http://affiliates.site.com?aff=jimco</NavigateUrl>
6     <AlternateText>Get a really cool product by clicking here.</AlternateText>
7     <Height>60</Height>
8     <Width>468</Height>
9     <Keyword>Technical</Keyword>
10    <Impressions>1000</Impressions>
11  </Ad>
12 </Advertisements>

```

The file can have any number of `<Ad>` blocks. Table 25.2 lists the XML elements in the advertisement file and their use.

**tip**

URLs that you enter in the advertisement file must be encoded. Therefore, if your link contains special characters such as an ampersand, you must use the HTML-encoded version of that character. In the case of an ampersand, `&` should be entered as `&`.

Table 25.2 XML Elements in an Advertisement File

Element Name	Description	Required
ImageUrl	The URL for the image that is displayed. This can be a relative or absolute URL.	Yes
NavigateUrl	The URL to which the user will navigate when the banner is clicked.	No
AlternateText	The text that appears in a tooltip when the user hovers over the banner. Also read by screen readers.	No
Height	The height of the banner.	No
Width	The width of the banner.	No
Keyword	A keyword that allows the control to filter based on categories.	No
Impressions	A value that defines how likely a specific banner will display in relation to other ads.	No

The `Impressions` element uses an arbitrary value that has meaning only in relation to the values specified for other ads. In other words, if one ad has an `Impressions` value of 500 and a second ad has an `Impressions` value of 1000, the second ad is twice as likely to appear.

The `Keyword` element is a powerful element and one that you will probably find many uses for. Suppose you have an `AdRotator` control on two different pages. One of those pages features computer articles and the other features articles on finance. By specifying a keyword of `Computers` for ads of interest to readers of your computer articles and a keyword of `Finance` for ads of interest to readers of your finance articles, you can ensure that the most relevant ads are displayed to your users.

Creating a Simple AdRotator Page

Let's create a simple Web Form that displays rotating banner ads using the AdRotator control. You'll need a few graphic files with the same dimensions to create the page. If you don't have any handy, you can find an assortment of banners on the website that accompanies this book.

Begin by creating a new ASP.NET Web Form in a new site. Save the Web Form as `adrotator.aspx`. Now we're ready to set up the rotating banners. Do the following:

1. Add a new AdRotator control to the Web Form by either double-clicking on the AdRotator control in the Toolbox or by dragging and dropping an AdRotator control onto the page.
2. You should see the AdRotator Tasks pop-up, as shown in Figure 25.8. Ignore that dialog and save `adrotator.aspx`.



Figure 25.8

The AdRotator Tasks pop-up enables you to configure a data source. However, we're not going to use it here.

Next we need to add some images to use in the AdRotator control. Create a new folder in your site called `images` and either copy the images from the `Chapter25\Banners` directory on the website into that folder or add your own images.

Creating the Advertisement File

The next step is to create an advertisement file to feed to the AdRotator control. Remember that an advertisement file is an XML file in a specific format that tells the AdRotator control how to behave, including what banners to display and where to link for each banner.

1. Create a new XML file by clicking File, New, Page and selecting XML from the General section, as shown in Figure 25.9.

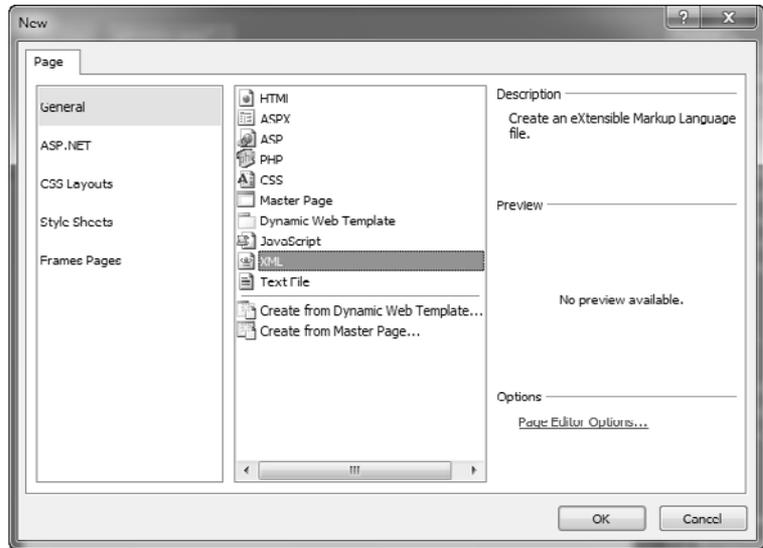


tip

Some people think that the AdRotator control rotates ads at a preset interval while a page is being displayed. That's not the way it works. It chooses a random ad to display when the page is loaded. The chosen ad will not change until the page is reloaded.

Figure 25.9

A new XML file is needed to contain an advertisement file for your AdRotator control.



2. Add the following XML code to the new XML file:

```
<Advertisements>
  <Ad>
    <ImageUrl>~/images/ComputerTown.jpg</ImageUrl>
    <NavigateUrl>http://www.jimcosoftware.com</NavigateUrl>
    <Keyword>Computers</Keyword>
    <AlternateText>The best computers in town!</AlternateText>
    <Impressions>100</Impressions>
  </Ad>

  <Ad>
    <ImageUrl>~/images/dsw.jpg</ImageUrl>
    <NavigateUrl>http://www.microsoft.com</NavigateUrl>
    <Keyword>Computers</Keyword>
    <AlternateText>All the software you need.</AlternateText>
    <Impressions>100</Impressions>
  </Ad>

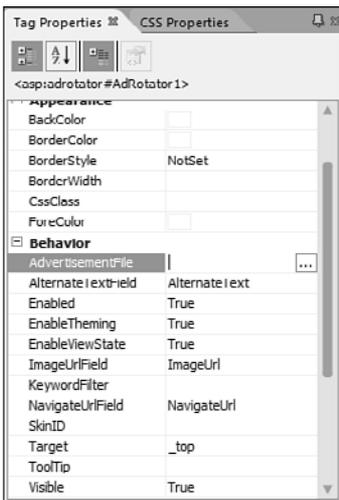
  <Ad>
    <ImageUrl>~/images/Taxman.jpg</ImageUrl>
    <NavigateUrl>http://www.asp.net</NavigateUrl>
    <Keyword>Finance</Keyword>
    <AlternateText>Don't be late for April 15th.</AlternateText>
```

```
<Impressions>100</Impressions>
</Ad>
</Advertisements>
```

3. Save the XML file as `ads.xml`.
4. Switch back to the `adrotator.aspx` page and click the `AdRotator` control you added previously to select it.
5. In the Tag Properties pane, click the `AdvertisementFile` property and click the button, as shown in Figure 25.10.

**tip**

If you track the effectiveness of the ads you use, you can use the `Impressions` property to ensure that the most effective ads appear more often.

**Figure 25.10**

The Tag Properties pane is a convenient way to set the `AdvertisementFile` property of the `AdRotator` control.

6. Browse to the `ads.xml` file that you saved in step 3 and select it.
7. Click `Open` to set the `AdvertisementFile` property of the `AdRotator` control.

Now save the `adrotator.aspx` page and browse it to see your ads in action. If you refresh the page a few times, you'll see the ads change.

Notice that even though the `AdRotator` control appears small in Design View in Expression Web, it resizes itself to match the size of your banner ads. To aid in the layout of your page, you might want to change the `Height` and `Width` properties of the `AdRotator` control to match the banners you are using. You will then easily be able to see how much page real estate is being taken up by the control while you're designing your page.

In this example, the ad that is chosen is truly random because the `Impressions` element is the same for all the ads. (Even though we configured a `Keyword` element for each ad, we haven't implemented any keyword filtering yet.) You can change the probability that a particular ad will display by changing the value of the `Impressions` element for that ad.

Open the `ads.xml` file in Expression Web and change the value of the `Impressions` element for one of the ads from this:

```
<Impressions>100</Impressions>
```

to this:

```
<Impressions>500</Impressions>
```

and then save `ads.xml` and browse the `adrotator.aspx` page again. Refresh the page a few times, and you'll notice that the ad you changed shows up much more often than the other ads.

As noted previously, you can use the `Keyword` element in your advertisement file to ensure that ads that aren't relevant to readers of a particular page in your site are not displayed. When we created the `ads.xml` file earlier, we added a `Keyword` element for each of our ads. We'll now use the `KeywordFilter` property of the `AdRotator` control to restrict the ads that are displayed based on the specified keyword.

1. Change the `Impressions` element that you modified earlier back to the original value of 100.
2. Save the `ads.xml` file.
3. Open the `adrotator.aspx` Web Form in Expression Web.
4. Click the `AdRotator` control to make it active.
5. In the Tag Properties pane, change the `KeywordFilter` property to `Computers`, as shown in Figure 25.11.
6. Save the `adrotator.aspx` page.

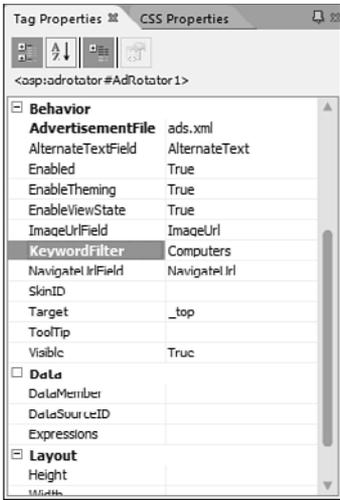
Now browse the `adrotator.aspx` page and refresh it several times. Notice that only the ads with a keyword of `Computers` are displayed. The finance ad will not be displayed on this page.

 **note**

Some of you who have worked with ASP.NET (and possibly some who haven't) might be wondering what happens to your ads if the page that's being displayed is cached. The developers of ASP.NET thought about that, too. They wrote the `AdRotator` control so that it explicitly prevents the caching of ads. Even if you cache the page on which the `AdRotator` control appears, you are always guaranteed to get fresh ads.

 **note**

The `Calendar` control is most powerful when paired with an ASP.NET code file so that you can access the selected date programmatically.

**Figure 25.11**

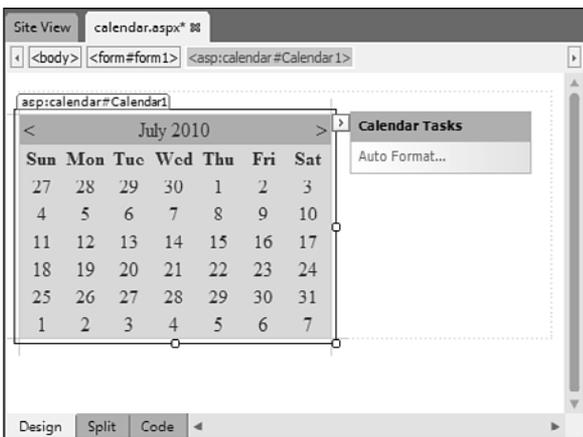
The `KeywordFilter` property enables you to ensure that only relevant ads are displayed.

The Calendar Control

The ASP.NET Calendar control serves two purposes: It displays a date in a familiar format and accepts user input of a date. The Calendar control can also be formatted using one of many autofor-mating styles.

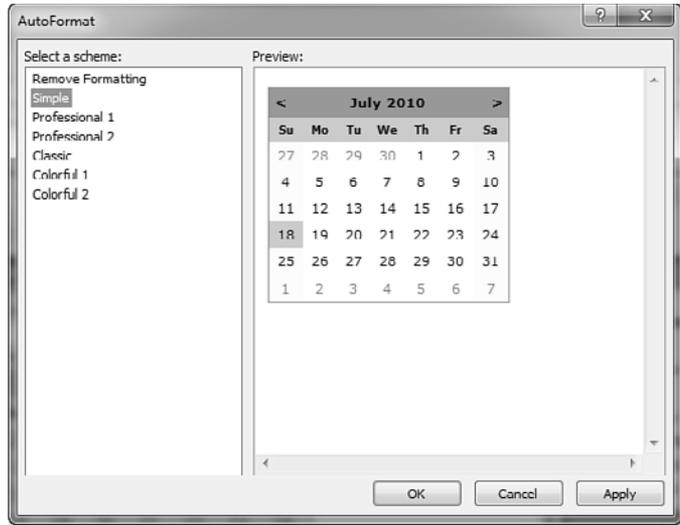
Formatting the Calendar Control

When you first insert a Calendar control onto a page, you are presented with the Calendar Tasks pop-up, as shown in Figure 25.12. By clicking the `AutoFormat...` link in the pop-up, you can easily apply a preset style to the Calendar control, as shown in Figure 25.13.

**Figure 25.12**

The Calendar Tasks pop-up appears automatically when a control is inserted or when you click the arrow button next to the control.

Figure 25.13
Formatting the Calendar control in an attractive style is simple using the AutoFormat feature.



Calendar Control Properties

The Calendar control exposes many properties to control not only its appearance, but also its behavior. Let's have a look at some of the properties that are available when using the Calendar control.

CaptionAlign

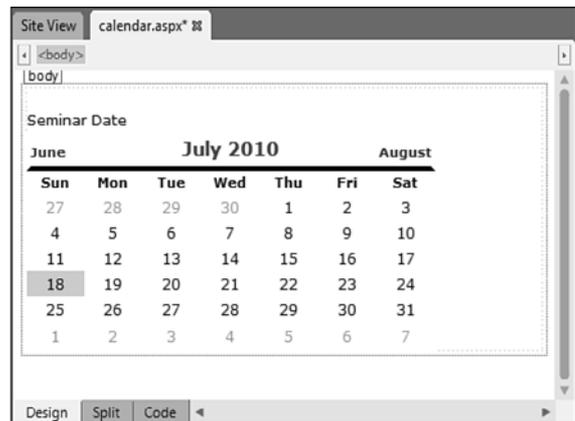
The CaptionAlign property controls the alignment of the text that is specified in the Caption property. Valid values are NotSet, Top, Bottom, Left, and Right. Figure 25.14 shows a Calendar control with a left-aligned caption of Seminar Date.



tip

In the English language, the Shortest property displays the same values as the FirstTwoLetters property. Other languages might not display the same value.

Figure 25.14
The caption of a Calendar control is a convenient way to display the purpose of your calendar.



DayNameFormat

The `DayNameFormat` property defines how day names appear on your calendar. This property applies only when the `ShowDayHeader` property is set to `True`.

Table 25.3 lists the valid values for this property.

Table 25.3 DayNameFormat Property Values

Property Value	Description
Full	Displays the full day name (that is, Monday)
Short	Displays the short day name (that is, Mon)
FirstLetter	Displays only the first letter of the day (that is, M)
FirstTwoLetters	Displays the first two letters of the day (that is, Mo)
Shortest	Uses the shortest format possible to provide a unique value for each day (that is, Mo)

FirstDayOfWeek

The `FirstDayOfWeek` property configures which day appears as the leftmost day of the week in the `Calendar` control. Valid values are `Default`, `Sunday`, `Monday`, and so on. When set to `Default`, `Sunday` appears as the first day of the week.

NextMonthText

The `NextMonthText` property configures the text for the link that moves to the next month when the `ShowNextPrevMonth` property is `True` and the `NextPrevFormat` property is set to `CustomText`. By default, the value is `>`, which appears as `>` in the `Calendar` control.

The `Calendar` control, shown previously in Figure 25.12, shows the default value of `>` for the `NextMonthText` property as displayed in the `Calendar` control.

NextPrevFormat

The `NextPrevFormat` property controls the format of the `NextMonthText` and `PrevMonthText` properties. Valid values are `CustomText`, `ShortMonth`, and `FullMonth`. When the value is set to `CustomText`, the format is dictated by the `NextMonthText` property.

PrevMonthText

The `PrevMonthText` property is identical to the `NextMonthText` property except that it configures the text for the link that moves to the previous month when the `ShowNextPrevMonth` property is



tip

When you are specifying dates for properties of the `Calendar` control, the `.NET DateTime` data type is used. Fortunately, Expression Web provides a user-friendly calendar control of its own that allows you to easily select the date you want.

True and the `NextPrevFormat` property is set to `CustomText`. The default value is `<`, which appears as `<` in the Calendar control.

SelectedDate

The `SelectedDate` property configures the date that is selected when the Calendar control first loads in the web browser. You can also use this property in server-side ASP.NET code to determine the date that has been selected in the control.

SelectionMode

By default, the Calendar control allows you to select a specific date. However, the `SelectionMode` property allows you to configure the Calendar control so you can also select an entire week using the `DayWeek` value, an entire month using the `DayWeekMonth` value, or nothing at all by using the `None` value.

SelectMonthText

The `SelectMonthText` property allows you to configure the link text that is clicked to select the month displayed in the Calendar control. By default, the value is `>>`, which appears as `>>`. This property is applicable only when the `SelectionMode` property is set to `DayWeekMonth`.

SelectWeekText

The `SelectWeekText` property configures the link text that is clicked to select an entire week. By default, the value is `>`, which appears as `>`.

This property is applicable only when the `SelectionMode` property is set to `DayWeek` or `DayWeekMonth`.

ShowDayHeader

The `ShowDayHeader` property is a Boolean property (`True` or `False`) that controls whether the day of the week appears as a header in each column of the Calendar control. The default value is `True`.

ShowGridLines

The `ShowGridLines` property is a Boolean property (`True` or `False`) that controls whether the Calendar control displays gridlines with each date in its own cell. The default value is `False`.

ShowNextPrevMonth

The `ShowNextPrevMonth` property is a Boolean property (`True` or `False`) that controls whether the Calendar control displays links that allow for navigation between the currently displayed month and the previous/next month. The default value is `True`.

ShowTitle

The `ShowTitle` property is a Boolean property (True or False) that controls whether the month is displayed on the Calendar control. The default value is True.

TitleFormat

The `TitleFormat` property controls the appearance of the calendar's title when the `ShowTitle` property is set to True. The `TitleFormat` property can be set to either `MonthYear` or `Month`. In Figure 25.14, the value of this property is `MonthYear`.

UseAccessibleHeader

The `UseAccessibleHeader` property causes ASP.NET to render the Calendar control with `Title` attributes on the HTML tags for accessibility purposes. The `Title` attribute shows up as a tooltip when the HTML element is hovered over. It is also read by screen readers.

VisibleDate

The `VisibleDate` property controls what month is displayed when the Calendar control is loaded. Figure 25.15 shows the calendar that Expression Web provides to aid in setting this property. Note that only the month of the date you select is applicable.

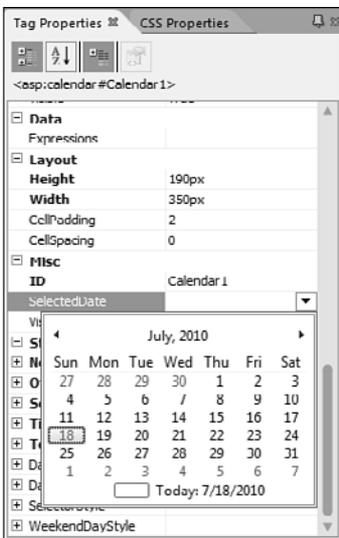


Figure 25.15

Expression Web provides a nice calendar from which you can select dates for applicable properties of the Calendar control.

The Wizard Control

Collecting data is a popular task for a web developer. The traditional approach to accomplishing this task is to build large forms that are often cumbersome. When a large amount of data needs to be collected, web developers often use forms that span multiple pages.

E-commerce sites are famous for this kind of approach, usually during the checkout process of a web transaction.

A wizard is a much better approach because it provides an interface that's similar to a regular Windows application, and it presents the user with a linear approach to collecting data.

The ASP.NET Wizard control takes care of all the plumbing so that if a user has to click the Previous button in the wizard to correct a previous entry, none of the data the user has entered into the wizard is lost. The Wizard control is an incredibly powerful control that can be utilized easily without writing any code at all (see Figure 25.16).



tip

In many cases, leaving the StepType at the default setting of Auto is sufficient.

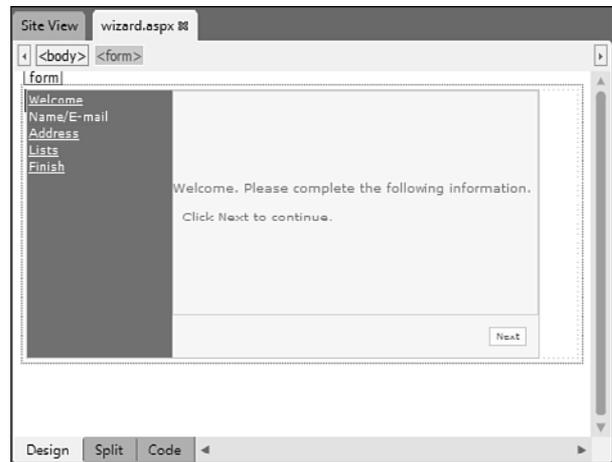


note

If you want, you can review the completed files from this chapter in the Examples\Ch25\Website folder on the website that accompanies this book.

Figure 25.16

The Wizard control is a powerful way to present or collect data in a step-by-step interface, and no code is required.



Wizard Steps

A Wizard control is made up of individual wizard steps, and each step is a particular type of step depending on its position within the wizard. Table 25.4 lists the types of steps in the Wizard control.

Table 25.4 Wizard Control Step Types

Step Type	Description
Auto	The step type is determined by its position within the wizard.
Complete	This is the last step in the wizard. No buttons are displayed in this step.
Finish	This is the final step where user information is collected. Previous and Finish buttons are displayed in this step.
Start	The first step in the wizard. This is often an introduction step. A Next button is displayed in this step.
Step	A normal step in the wizard.

Creating a Simple Wizard

To gain a full understanding of how the Wizard control works, let's create a simple wizard using the control and test some of the options available.

We'll create a simple wizard that does the following:

- Displays a simple welcome message in step 1
- Collects the user's name and email address in step 2
- Collects the user's street address in step 3
- Displays check boxes for different mailing lists in step 4
- Displays a Finish message in step 5

Start by creating a new disk-based site and a new ASP.NET Web Form. Save the Web Form as `wizard.aspx`. After you've done that, you're ready to add the Wizard control and begin customizing it:

1. Double-click the Wizard control in the Toolbox to add a new wizard to the page.
2. If the Wizard Tasks pop-up isn't visible, click the arrow button at the upper right of the Wizard control to display it.
3. Click the AutoFormat link in the Wizard Tasks pop-up.
4. Select Professional from the list of schemes, and click OK.

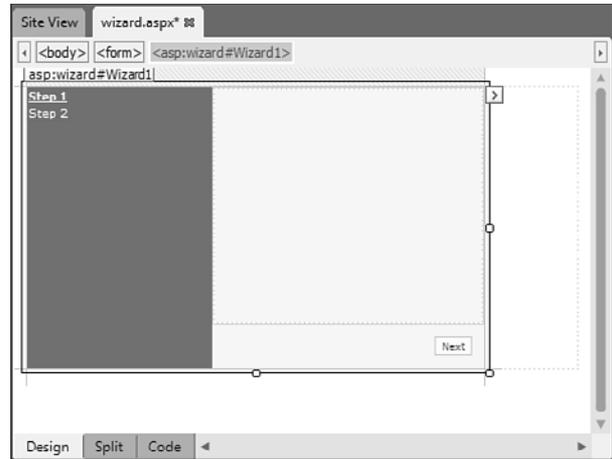
Your page should now look like Figure 25.17. (I've resized the control to make it easier to work with.)

note

Notice that the Add button has a downward-pointing arrow at the right edge of the button. Clicking that arrow allows you to choose between a WizardStep and a TemplatedWizardStep. We haven't talked about TemplatedWizardSteps yet, so if you click the arrow on the Add button, select WizardStep from the menu that appears.

Figure 25.17

You can give your Wizard control a more professional look by applying one of the preset schemes.

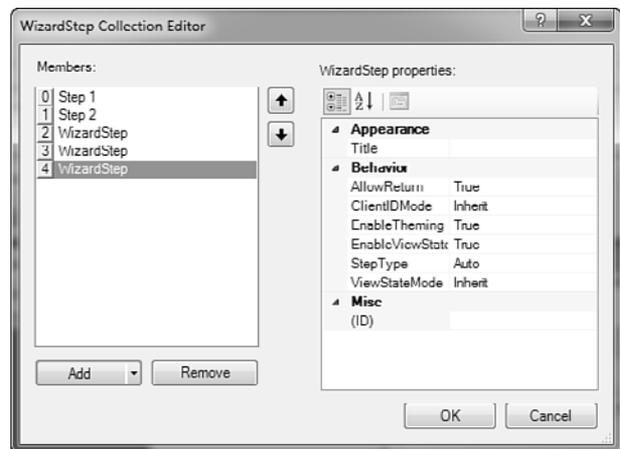


Note that two steps are created for you automatically. In this case, we need five steps, so we need to create three additional steps:

1. If the Wizard Tasks pop-up isn't visible, click the arrow button at the upper right of the Wizard control to display it.
2. Click the Add/Remove WizardSteps link to display the WizardStep Collection Editor.
3. Click the Add button three times to add three steps. The WizardStep Collection Editor should now look like Figure 25.18.

Figure 25.18

The WizardStep Collection Editor makes adding, editing, and removing WizardSteps fast and easy.



4. Select Step 1 in the list of steps and change the Title property to Welcome, as shown in Figure 25.19.

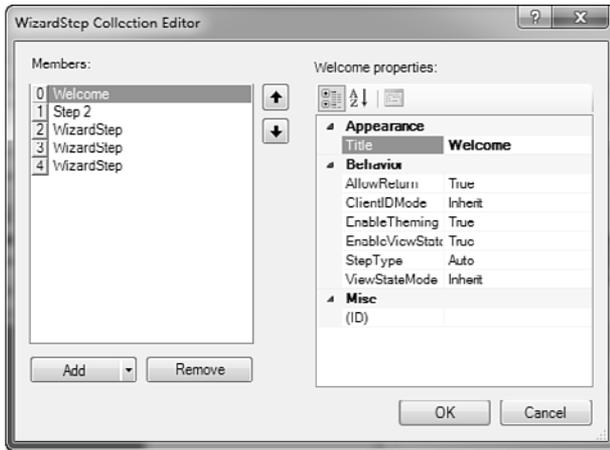


Figure 25.19
Editing the Title property for each step makes your wizard easier to use.

5. Select the second step in the list of steps and change the Title property to Name/E-mail.
6. Select the third step in the list of steps and change the Title property to Address.
7. Select the fourth step in the list of steps and change the Title property to Lists.
8. Select the final step in the list and change the Title property to Finish.
9. Click OK.

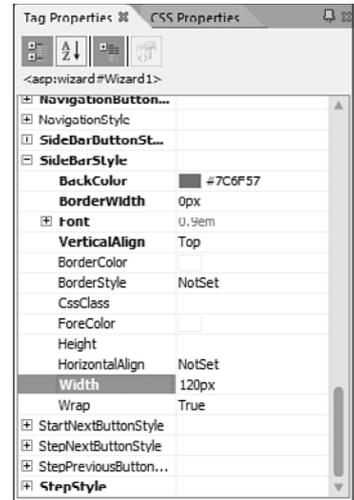
By default, the Wizard control is too small to contain the controls that we need to add for our wizard, so you need to adjust the size. Using the Tag Properties pane, set the Width property of the Wizard control to 425px and the Height to 225px. After you make that change, the sidebar at the left side of the Wizard control will be far too wide. To correct that, expand the SideBarStyle section of the Tag Properties pane and change the Width property to 120px, as shown in Figure 25.20.

Your wizard should now look like Figure 25.21. Notice that the Title property that you set in the WizardStep Collection Editor is displayed in the sidebar so that the user can clearly identify the current step when navigating through the Wizard control.

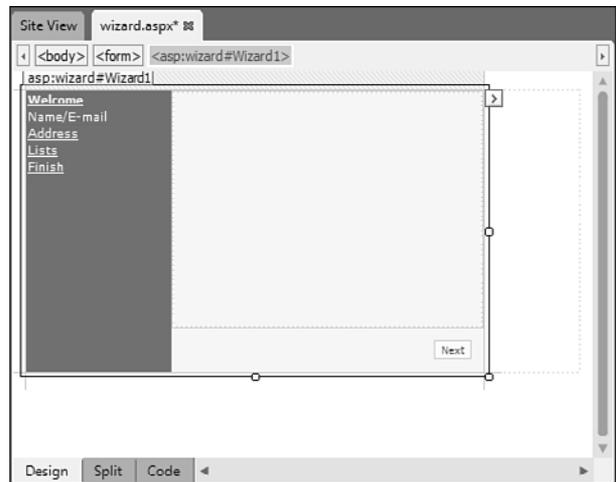
Now we're ready to add the necessary controls to the Wizard control so that we can collect the information we need. Expression Web makes that easy by allowing you to navigate to each step in the Wizard control in the designer, just as you would in your web browser.

Figure 25.20

The sidebar of the Wizard control must be resized so that it doesn't take up so much room. You can adjust it using the `SideBarStyle` in the Tag Properties pane.

**Figure 25.21**

The title of each step is clearly displayed in the sidebar of the Wizard control.



Step 1: Welcome

Click **Welcome** in the sidebar to switch to the first step. You need to add a brief welcome message here. Click in the clear area at the right of the sidebar and enter the following text:

```
Welcome. Please complete the following information.  
Click Next to continue.
```

Step 2: Name/Email

Click Name/E-mail in the sidebar to switch to the second step. We'll add a couple of `TextBox` controls here to collect the user's name and email address. Do the following:

1. Click inside the clear area at the right of the sidebar and add the text `Enter your name:`.
2. Make sure the insertion point is located at the right of the text you entered in step 1. Double-click the `TextBox` control in the Toolbox to add a new text box.
3. Click at the right of the text box so it's not selected, and then press `Enter` twice.
4. Enter the text `Enter your e-mail:`.
5. Make sure the insertion point is located at the right of the text you just entered. Double-click the `TextBox` control in the Toolbox to add a new text box.

Your Wizard control should now look like Figure 25.22.

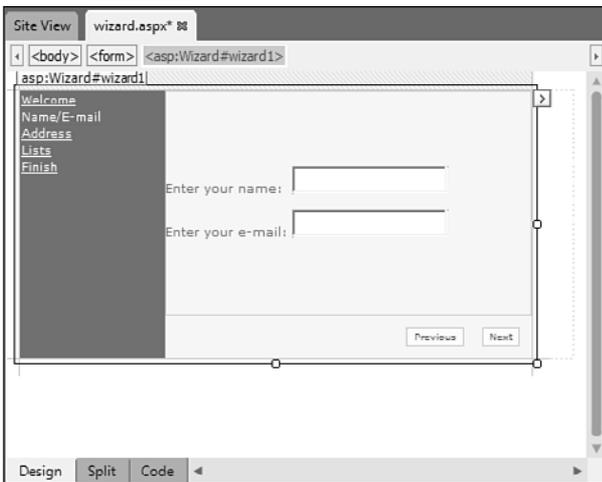


Figure 25.22

Step 2 of the Wizard control is now ready to accept a name and an email address.

Step 3: Address

We're now ready to move onto step 3 and create the interface for collecting the street address. Click Address in the sidebar to move to step 3 of the wizard:

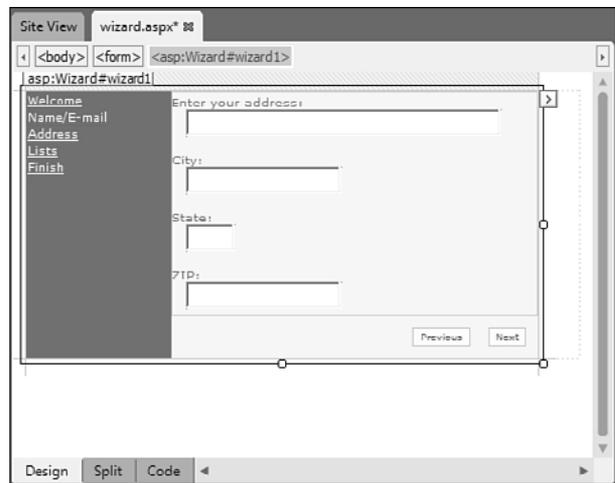
1. Click inside the clear area at the right of the sidebar and add the text `Enter your address:`.
2. Press `Enter` to insert a new line.
3. Double-click the `TextBox` control in the Toolbox to add a new text box.

4. Drag the right side of the `TextBox` control to make it wide enough to accommodate a street address (approximately 260px).
5. Click at the right of the text box so that it's not selected, and press `Enter` twice.
6. Enter the text **City:** and then press `Enter`.
7. Double-click the `TextBox` control in the Toolbox to add a new text box.
8. Click at the right of the text box so that it's not selected, and press `Enter` twice.
9. Enter the text **State:** and then press `Enter`.
10. Double-click the `TextBox` control in the Toolbox to add a new text box.
11. Drag the right side of the `TextBox` control and make it approximately 40px in width.
12. Click at the right of the text box so that it's not selected, and press `Enter` twice.
13. Enter the text **ZIP:** and then press `Enter`.
14. Double-click the `TextBox` control in the Toolbox to add a new text box.

Your Wizard control should now look like Figure 25.23.

Figure 25.23

Step 3 of the Wizard control contains many form fields used to collect the user's address.



Step 4: Lists

In step 4 of our wizard, we use check boxes to allow users to sign up for various mailing lists. A check in a box means that the user wants to subscribe to that particular list. Click **Lists** in the sidebar to move to step 4 of the wizard.

1. Click inside the clear area at the right of the sidebar and enter the text **Sign up for the following mailing lists:**.
2. Press Enter twice to insert two new lines.
3. Double-click the CheckBox control in the Toolbox to insert a new check box.
4. With the CheckBox control still selected, change the Text property in the Tag Properties pane to New Products.
5. Click at the right of the check box to ensure that it's not selected, and press Enter.
6. Double-click the CheckBox control in the Toolbox to insert a new check box.
7. With the CheckBox control still selected, change the Text property in the Tag Properties pane to Expression Web Tips.
8. Click at the right of the check box to ensure that it's not selected, and press Enter.
9. Double-click the CheckBox control in the Toolbox to insert a new check box.
10. With the CheckBox control still selected, change the Text property in the Tag Properties pane to Monthly Newsletter.

Your Wizard control should now look like Figure 25.24.

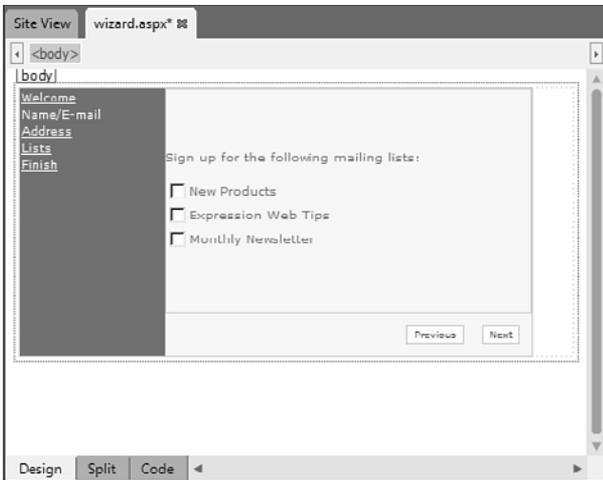


Figure 25.24

Step 4 of the wizard makes mailing list selection easy with a series of check boxes.

Step 5: Finish

The final step should simply display a message thanking the user for subscribing. Click **Finish** in the sidebar to move to step 4 of the wizard. Only the task in the Note to the right is left to complete the wizard:

You're now ready to preview your wizard, but first switch back to the **Welcome** step by clicking **Welcome** in the sidebar. You need to do this because the wizard will always start on the step that is selected in the designer, even if that step is not the first one. After you've selected the **Welcome** step, save and browse the wizard .aspx page to test your **Wizard** control. As you fill out the information in the **Wizard** control, try clicking the **Previous** button. You'll see that the **Wizard** control always maintains the information that you've entered so you never have to worry about reentering information if you need to go back to a previous step.



note

Click inside the clear area at the right of the sidebar and enter the text `Thank you for signing up for our mailing lists!`.



note

The **Wizard** control you've created here doesn't actually do anything with the information you've collected. In a real-world application, you would likely want to save the information from the wizard in a database.

Making ASP.NET Work for You

You've experienced just a tiny taste of the capabilities of ASP.NET in this chapter. There are some powerful and easy-to-configure controls in the standard Toolbox, but ASP.NET offers much more, some of which we will cover in the next few chapters.

Even after reading all the material in this book, you still need to experiment and try your hand at some real applications to grow your knowledge. The best way to learn a complex technology such as ASP.NET is to use it for one of your sites. Hopefully you'll find that the knowledge base we've given you here will serve you well as you go out on your own.

Formatting with Styles

Expression Web enables you to apply direct formatting to ASP.NET controls like the **Wizard** control. However, you're going to be much better off in the long run if you use CSS to format your controls.

The **Wizard** control has many properties that allow you to take advantage of CSS styles:

- **NavigationButtonStyle**—Controls the formatting of the navigation buttons (**Next**, **Previous**, **Finish**, and so on).
- **SideBarButtonStyle**—Controls the formatting of the sidebar buttons. In the sample wizard you created, the sidebar buttons are the links that appear in the sidebar.
- **SideBarStyle**—Controls the formatting of the sidebar.
- **StepStyle**—Controls the formatting of the wizard's steps.

Numerous other styles enable you to format your **Wizard** control exactly the way you want.

➡ *For more information on CSS in Expression Web, see Chapter 18, "Managing CSS Styles."*

This page intentionally left blank

USING ASP.NET NAVIGATION CONTROLS

Overview of Navigation Systems

What is the most overlooked aspect of most sites? If you said it's the navigational structure, you're exactly right. Many web designers spend a considerable amount of time creating great-looking graphics and carefully choosing just the right colors, and then they quickly slap down an inefficient site navigation structure.

A good web designer will tell you that content is the most important aspect of any site. I certainly wouldn't disagree with that, but I'd give navigational structure the same importance as content. After all, what good is great content if your site's viewers are unable to locate it?

One of the most popular navigation techniques in use today is the hierarchical menu system. You've no doubt seen these menus on many sites. When you hover over a particular topic, a menu pops up, allowing you to dig deeper into the site's content, as shown in Figure 26.1.

Other navigational systems are more appropriate for more complex structures. For example, Microsoft's MSDN site contains an enormous amount of hierarchical content. The sheer volume of content would quickly overwhelm a menu system such as the one shown in Figure 26.1. In these situations, a tree view is a much better system because it is, by nature, hierarchical. As shown in Figure 26.2, the tree view system works perfectly for a site like MSDN.



Figure 26.1
The menu on the Jimco Books site is implemented using a DHTML menu system. Hover over a topic and you see a menu for that topic.

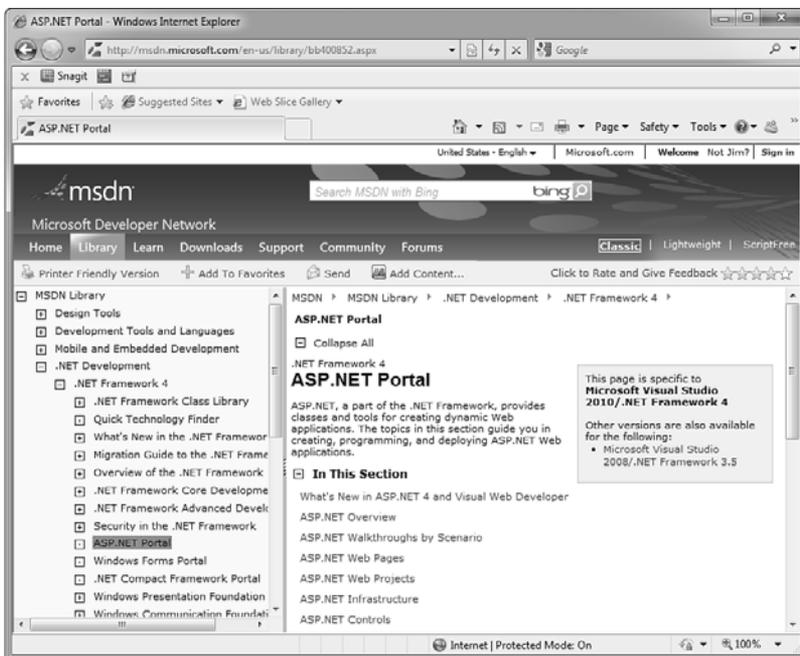
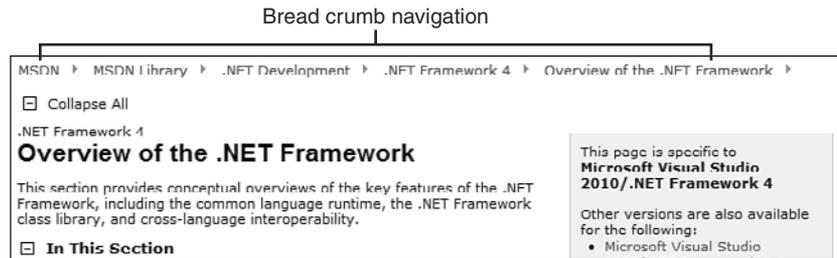


Figure 26.2
The tree view system is the perfect solution to presenting large amounts of hierarchical information.

Either of these navigation systems is nicely supplemented by a bread crumb system. Remember the story of Hansel and Gretel? Hansel left a trail of bread crumbs along the path so he and Gretel would be able to find their way back to where they started. The bread crumb navigation system got its name from Hansel's famous technique, and while it didn't work out so well for Hansel and Gretel, bread crumbs can make a huge difference for your site visitors. Figure 26.3 shows the use of bread crumb navigation on the MSDN site.

Figure 26.3

The bread crumb navigation system used on MSDN lets you easily determine where you've been. Each time you dig deeper into the site, a new link is added to the bread crumb navigation system.



You might be asking yourself at this point just how difficult it is to build one or more of these cool navigation systems. With ASP.NET and Expression Web, it's so easy you likely won't believe it. ASP.NET offers a `Menu` control for creating professional menus, a `TreeView` control for creating tree view-style navigation systems, and a `SiteMapPath` control for easily creating bread crumb navigation systems. You can use each of these controls without writing any code at all, but they all expose advanced functionality as well for those web designers who are more adept at programming.

Creating a Sitemap File

Chances are that you're going to want to use a couple of different ASP.NET navigation controls at the same time. For example, you might want to use the `Menu` control for your navigational menu system and the `SiteMapPath` control to provide bread crumb navigation links. Imagine the nightmare you'd have to deal with if you had to manage the links in each of those controls manually as your site grows and changes!

Fortunately, ASP.NET allows you to define your site's navigational structure in one place, and then all the navigation controls automatically read that structure to produce your menu, bread crumb links, and tree view nodes. ASP.NET uses a special XML file called a *sitemap file* for this purpose.

The code in Listing 26.1 shows a typical sitemap file.

Listing 26.1 A Sitemap File

```
<?xml version="1.0" encoding="utf-8" ?>
<sitemap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode>
<siteMapNode url="default.aspx" title="Home Page"
  description="Company Web Site Home" />
    <siteMapNode url="software.aspx" title="Software"
  description="Our Software">
      <siteMapNode url="apps.aspx" title="Applications"
description="Software Applications" />
        <siteMapNode url="addins.aspx" title="Add-ins"
description="Software Add-ins" />
    </siteMapNode>

    <siteMapNode url="reviews.aspx" title="Reviews"
description="Software Reviews"/>
    <siteMapNode url="about.aspx" title="About" description="About Us"/>
      <siteMapNode url="contact.aspx" title="Contact"
description="Contact Us" />
    </siteMapNode>
</siteMap>
```

Notice that a sitemap file has exactly one root `siteMapNode`. All other `siteMapNodes` are nested within the root `siteMapNode`.

The easiest way to use a sitemap file is to save it as `web.sitemap` in the root directory of your site. As we progress through the rest of this chapter, you get a good idea of how the sitemap file is used to build navigation structure into your site.

Using the ASP.NET Menu Control

As I mentioned previously, hierarchical menus have become the most common navigational element on the Web these days. Almost every site uses them, and it's pretty safe to say that all the sites operated by the major players use them. That level of popularity has caused enormous growth in the number of software programs that allow nonprogrammers to create and manage powerful navigation systems using menus.

When it comes to implementing a menu system in Expression Web, you have three choices: You can write your own code, you can use a third-party application to generate the code for your menus, or you can use a server-side solution such as the ASP.NET Menu control.



tip

Expression Web's Save dialog doesn't have a file type specifically for sitemap files. If you select XML as the file type, Expression Web adds an `.xml` file extension, thus making the file invalid as a sitemap file. Therefore, you should select All Files in the Save As Type dropdown in the Save dialog when saving your sitemap file.



tip

Some web designers are moving toward using pure CSS menus instead of DHTML because they are compatible with more browsers. However, pure CSS menus are not quite as robust as DHTML menus.

If you'd like to see a tutorial on creating pure CSS menus, you can view a great one at www.projectseven.com/tutorials/navigation/auto_hide/index.htm.

Choosing a Menu System

Web design is full of choices, so it should come as no surprise that there are plenty of choices in menu systems. Choosing the right system can be a burdensome task, but some general rules might make it easier.

If all your pages are ASP.NET pages, the ASP.NET Menu control is a good choice. By using the Menu control, you can take advantage of a sitemap file so all your navigation tools feed off the same data. You also can take advantage of other ASP.NET features such as themes and skins.

If you have pages that are not ASP.NET (HTML pages, ASP pages, PHP pages, and so on), you have two choices: use a third-party tool to create your menus or write all the code yourself.

I'm a big believer in using a prebuilt solution if one is available. Writing code for menus is a fairly large task. You might as well take advantage of someone else's development and tedious debugging. However, carefully check out the system you plan on using. Ask other web designers who've used it before you buy it, and don't purchase a system based on feedback from amateur web designers.

Creating a Test Site

Before we dig deeper into the ASP.NET Menu control, let's create a site that allows us to experiment with it and other navigation controls in ASP.NET:

1. Create a new site.
2. Delete the home page.
3. Create a new ASPX page.
4. Enter the text **Home Page** on the page and save it as `default.aspx`.
5. Create a new ASPX page.
6. Enter the text **Software Page** on the page and save it as `software.aspx`.
7. Create a new ASPX page.
8. Enter the text **Add-ins Page** on the page and save it as `addins.aspx`.
9. Create a new ASPX page.
10. Enter the text **Applications Page** on the page and save it as `apps.aspx`.
11. Create a new ASPX page.
12. Enter the text **Reviews Page** on the page and save it as `reviews.aspx`.
13. Create a new ASPX page.



You can download the completed site for this chapter from the website for this book at www.informit.com/register.

14. Enter the text **About Us** on the page and save it as `about.aspx`.
15. Create a new ASPX page.
16. Enter the text **Contact Us** on the page and save it as `contact.aspx`.
17. Create a new XML file.
18. Add the code from Listing 26.1 to the file.
19. Choose All Files (*.*) from the Save As Type drop-down.
20. Save the file as `web.sitemap`.

➔ For more information on creating a site, see Chapter 2, “Creating, Opening, and Importing Sites.”

Adding a Menu Control

The `web.sitemap` file is a special file that ASP.NET looks for when using navigation controls. If a `web.sitemap` file exists in the root of the site, ASP.NET uses it automatically to feed the navigation links in the controls you are using. You don't have to write any code at all. It just happens automatically.

To see what I mean by this, do the following:

1. Open the `default.aspx` page.
2. Make sure you are in Design View.
3. Place the insertion point below the text you entered on the page.
4. Insert a Menu control onto the page, as shown in Figure 26.4.
5. Click New Data Source from the Choose Data Source drop-down in the Menu Tasks dialog, as shown in Figure 26.5.
6. Select the Sitemap option in the Data Source Configuration Wizard, as shown in Figure 26.6, and click OK.

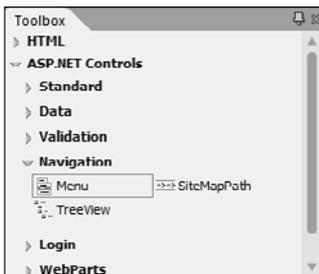
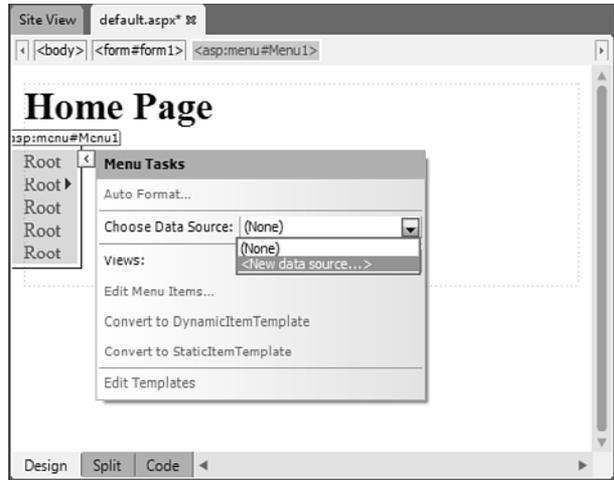


Figure 26.4

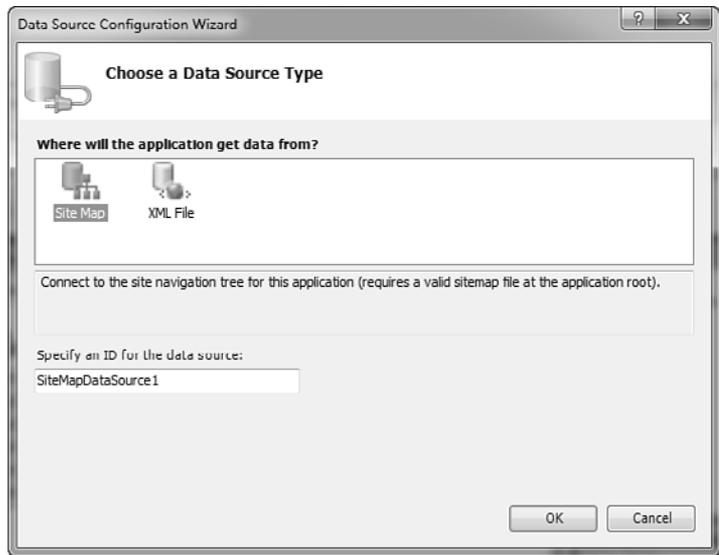
The Menu control is located in the Navigation section of the ASP.NET controls toolbox.

Figure 26.5

The Menu Tasks dialog makes it easy to configure a Menu control, including connecting it to a sitemap file in your site.

**Figure 26.6**

The Data Source Configuration Wizard allows you to connect controls on your page to data. In this case, we're connecting the Menu control to the sitemap file we created earlier.



Save the page. At this point, the Menu control is invisible when previewed in a browser. We'll fix that shortly.

One of the strange things about the ASP.NET sitemap file is that you are required to have exactly one root sitemap node. In the case of Listing 26.1, I created an empty sitemap node to satisfy that requirement. However, that causes the menu to be blank when you browse it.

The problem of a blank menu is easily resolved, but it requires you to go into Code View to fix it:

1. If it's not already open, open `default.aspx`.
2. Switch to Code View.
3. Locate the code for the `SiteMapDataSource` control. It looks like this:

```
<asp:SiteMapDataSource runat="server" ID="SiteMapDataSource1">
```

4. Change the code in step 3 to this:

```
<asp:SiteMapDataSource runat="server"  
ID="SiteMapDataSource1"  
ShowStartingNode="false">
```

5. Save the page.

Now browse the page again and you'll see a menu system displays correctly. However, it's not very pretty. Right now it consists of only a few text links. Luckily, you have quite a few options for sprucing up the appearance of a Menu control.



tip

Expression Web provides IntelliSense support for editing ASP.NET controls. If you place the insertion point right before the closing bracket on the `<asp:SiteMapDataSource>` tag and press the spacebar, you will be able to add the `ShowStartingNode` attribute using IntelliSense.



Cannot See Menu Control in Browser

Sometimes when you preview a page, the Menu control isn't there even after you've added it to the page.

The most common cause of this is a missing data source for the Menu control. Be sure you configure a data source using the Choose Data Source drop-down on the Menu Tasks dialog. If you've already configured a data source, make sure it is selected in the drop-down before you preview the page.

Formatting the Menu Control

When it comes to formatting the Menu control, you have several options. You can use the existing AutoFormat feature if you want to format it quickly. You can also use an ASP.NET theme, skin the control using the skinning features of ASP.NET, or use CSS styles.

Using the AutoFormat Feature

The quickest way to format the Menu control for a better appearance is by using the AutoFormat feature available from the Menu Tasks dialog, as shown in Figure 26.7.

Click the AutoFormat link; the AutoFormat dialog appears with four formats, as shown in Figure 26.8. You can remove any existing formatting by selecting the Remove Formatting option from the list. Choose a format you find attractive and click OK to apply it.

Figure 26.7

The Menu Tasks dialog contains a link for the AutoFormat function. This is by far the fastest way to format the Menu control for a better appearance.

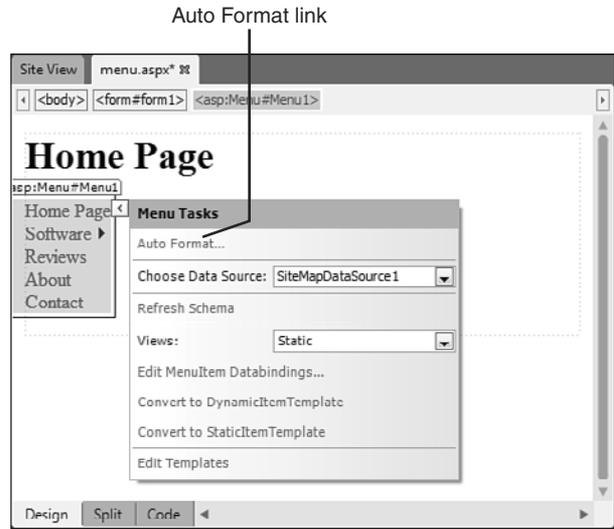
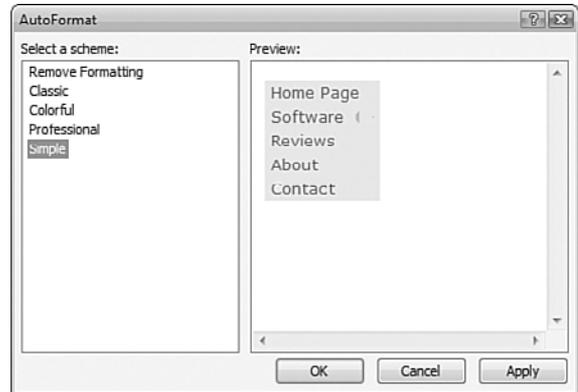


Figure 26.8

The AutoFormat dialog contains four formats for the Menu control. The Remove Formatting option allows you to easily remove any previously applied formatting.



Using CSS Styles

The Menu control exposes many properties that can be used to format the control using CSS. The Menu control uses two kinds of styles:

- **Dynamic styles**—Styles that apply to menu items that aren't visible by default. These menu items appear when another menu item is hovered over.

- **Static styles**—Styles that apply to menu items that are visible by default.

To format a Menu control using these styles, use the Tag Properties panel to alter the styles. For example, to change the style of dynamic menu items, change the settings for the `DynamicMenuItemStyle` property, as shown in Figure 26.9.

The DynamicMenuItemStyle

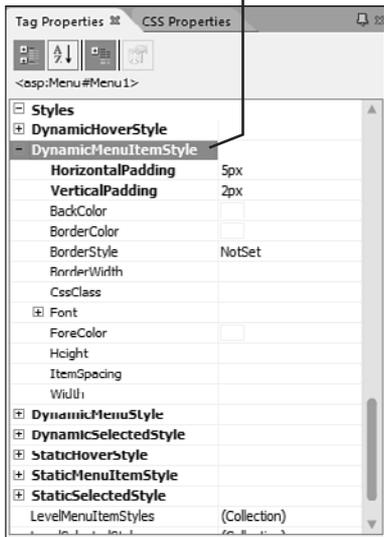


Figure 26.9

The `DynamicMenuItemStyle` property controls the appearance of dynamic menu items. The easiest way to modify it is via the Tag Properties panel.

➔ For more information on the Tag Properties panel, see Chapter 7, “Editing Tag Properties.”

➔ For more information on using CSS, see Chapter 17, “Creating Style Sheets.”

When you choose to automatically format the Menu control using the AutoFormat dialog, you are actually just setting some of the CSS properties of the menu. Open the AutoFormat dialog for the menu and select the Simple scheme. Now look at the properties that have been set in the Tag Properties panel. You should see several of the CSS properties set, as shown in Figure 26.10.

You can experiment with the styles available using the Tag Properties panel. Just keep in mind that changing a style for a dynamic item affects only those items that are not initially visible and does not affect the appearance of any items that are visible when the Menu control is first loaded.



note

For a complete discussion of using ASP.NET themes and skins, read my book *The Microsoft Expression Web Developer’s Guide to ASP.NET 3.5* from Que Publishing.

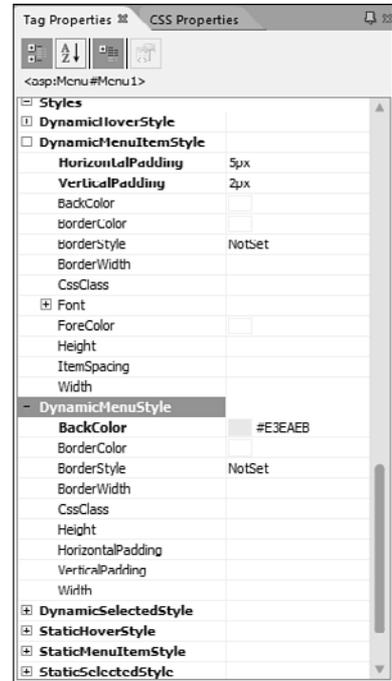


tip

When you set the properties of any ASP.NET control, Expression Web simply adds the property to the control’s tag in Code View. Therefore, you could set properties in Code View instead of using the Tag Properties panel. Expression Web gives you IntelliSense for all properties in Code View.

Figure 26.10

The Tag Properties panel shows that selecting an AutoFormat scheme simply presets many of the CSS properties for the Menu control.



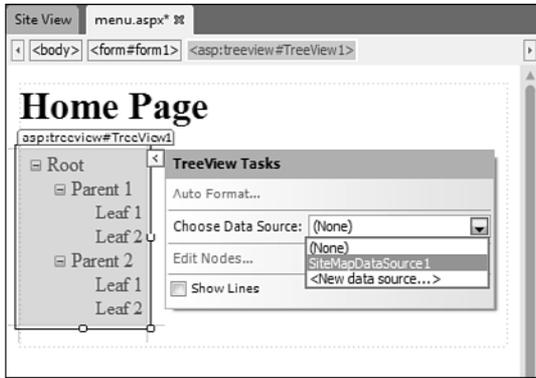
Using the ASP.NET TreeView Control

Now that you've dug through some of the features of the Menu control and how to use it on a page, you'll have no trouble at all using the TreeView control. In fact, it's similar to the Menu control and can use the same sitemap file you used with the Menu control.

To add a TreeView control to your page:

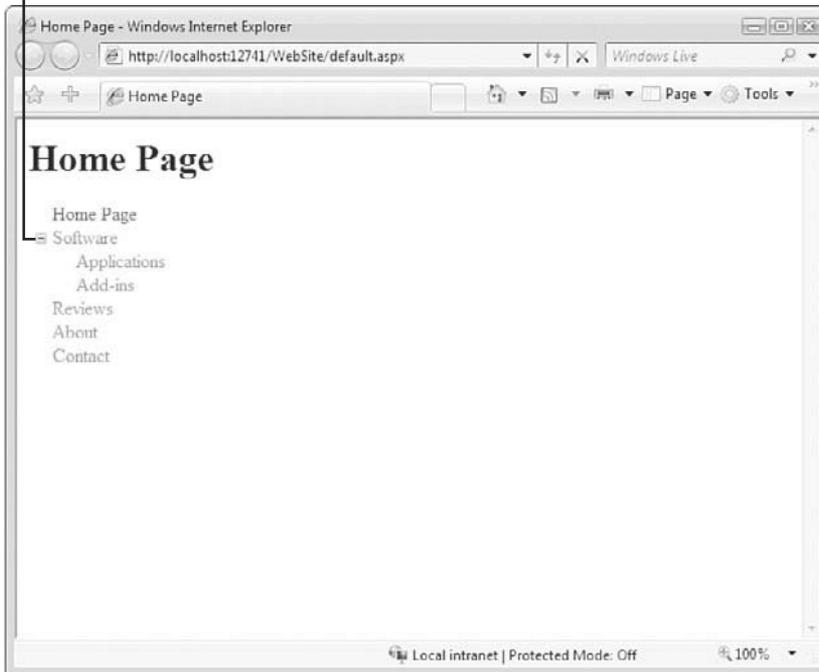
1. Open `default.aspx`.
2. Place the insertion point where you want the TreeView control to be inserted.
3. Double-click the TreeView control in the toolbox to add it to the page.
4. In the TreeView Tasks dialog, select SiteMapDataSource1 from the Choose Data Source dropdown, as shown in Figure 26.11.

Save the page and preview it in your browser. The TreeView control displays links to pages in your site, as shown in Figure 26.12.

**Figure 26.11**

The TreeView control uses the same SiteMapDataSource1 used by the Menu control.

Minus button

**Figure 26.12**

Once connected to the sitemap file we created earlier, the TreeView control displays links to all your pages in a hierarchical tree. Parent nodes contain a minus button that can be clicked to collapse the nodes beneath them.



TreeView Not Visible in Browser

Unlike the SiteMapPath control, both the Menu and TreeView controls require a data source on the page. (The data source is not visible in Design View.)

If you can't see the TreeView control, make sure a data source has been chosen from the Choose Data Source drop-down on the TreeView Tasks dialog. If a data source does not exist in the drop-down, create one as described earlier in this chapter.

Each link in a TreeView control is referred to as a *node*. There can be several types of nodes in a TreeView control:

- **Root node**—Each TreeView contains exactly one root node. The root node is the top-level node. There are no nodes above the root node.
- **Parent node**—A parent node has one or more nodes nested beneath it. In the TreeView control in Figure 26.12, the Software node is a parent node.
- **Leaf node**—A leaf node is any node that has no child nodes. The only node in Figure 26.12 that is not a leaf node is the Software node.

Notice that the Software node has a minus button next to it that can be clicked to collapse the nodes beneath it. You can then click the plus sign to expand the nodes again.

note

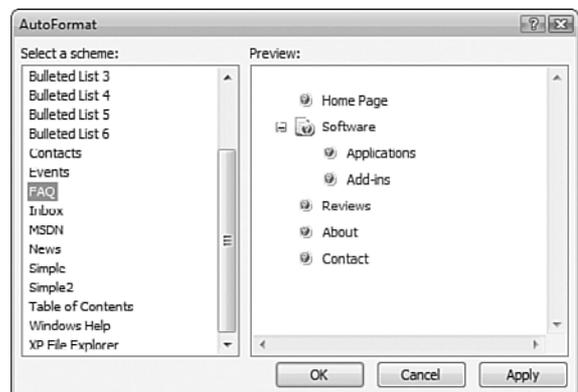
You might be wondering what the root node in Figure 26.12 is. Remember that we configured SiteMapDataSource1 so the root node is not shown. Therefore, the root node for the TreeView is not shown on the page.

Formatting the TreeView Control

The TreeView control can be easily formatted using the AutoFormat dialog, just as you did with the Menu control. However, as shown in Figure 26.13, there are many more choices in the TreeView control's AutoFormat dialog.

Figure 26.13

The TreeView control offers many formatting options in the AutoFormat dialog.



Using the TreeView Control's Properties

The `TreeView` control also has several unique formatting properties you can set in the Tag Properties panel.

ExpandDepth Property The `ExpandDepth` property controls the number of levels in the `TreeViewStyle` menu that are initially visible when your page loads. The default value is `FullyExpand`, which expands all nodes in the `TreeView` control. However, you can specify any number of levels to expand by default by specifying a number for the `ExpandDepth` property.

Notice in Figure 26.12 that the `Software` node is expanded so you can see the `Applications` and `Add-ins` nodes. To configure the `TreeView` so it is fully collapsed, set the `ExpandDepth` property to `0`.

ImageSet Property As shown in Figure 26.14, the `ImageSet` property configures the images used in the `TreeView` control. Notice that the values available for this property reflect the `AutoFormat` options available in Figure 26.13.

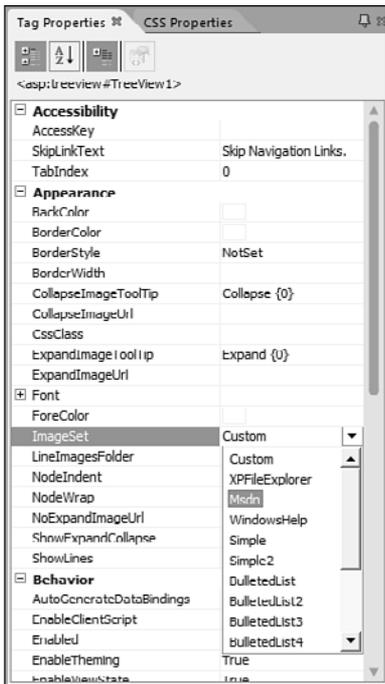


Figure 26.14

The `ImageSet` property controls the images used for the `TreeView` control. The values available match the `AutoFormat` options.

NodeIndent Property The `NodeIndent` property configures the number of pixels that nodes are indented. Note that only nodes that are children of other nodes are affected. In the case of the `TreeView` control we're using, only the `Applications` and `Add-ins` nodes are affected by the `NodeIndent` property.

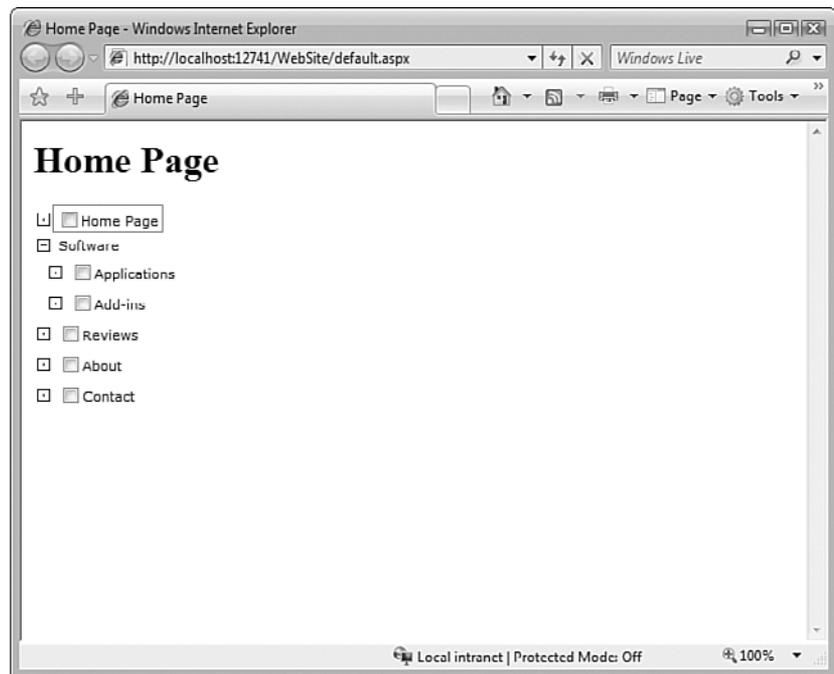
ShowCheckBoxes Property The `ShowCheckBoxes` property allows you to display check boxes on specific types of nodes. Valid values are `None`, `Root`, `Parent`, `Leaf`, or `All`. By showing check boxes for nodes, you can allow users to select multiple nodes in a `TreeView` control.

The `TreeView` control in Figure 26.15 has a `ShowCheckBoxes` property set to `Leaf`.

ShowExpandCollapse Property The `ShowExpandCollapse` property configures whether the plus and minus buttons are displayed for parent nodes. The `TreeView` control shown previously in Figure 26.12 has the `ShowExpandCollapse` property set to `True`.

ShowLines Property By default, there are no lines connecting nodes in a `TreeView` control. If you'd like lines to be drawn connecting the nodes, set the `ShowLines` property to `True` (see Figure 26.16).

Figure 26.15
If you show check boxes on `TreeView` control nodes, users can select multiple nodes. The `ShowCheckBoxes` property gives you control over which nodes display check boxes.



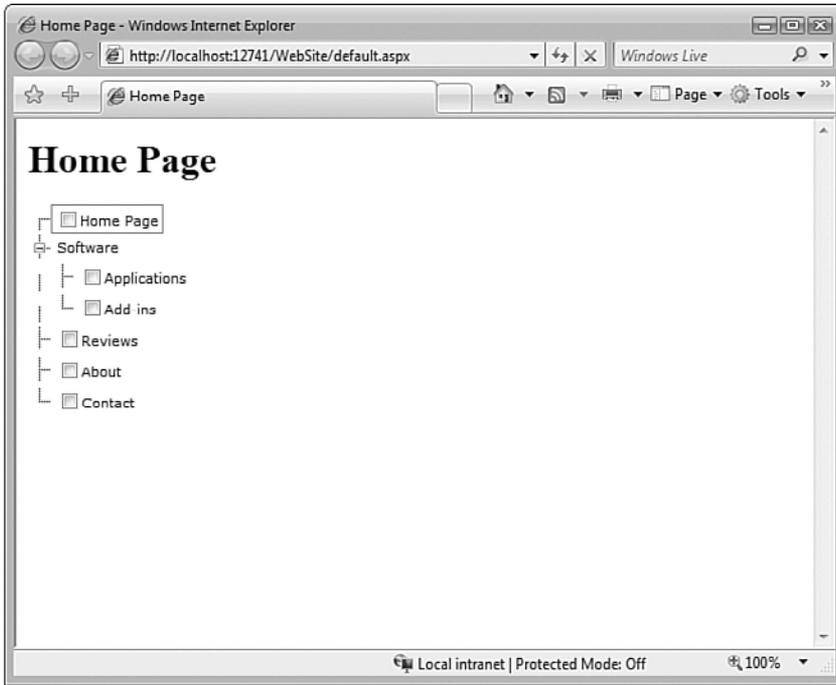


Figure 26.16
The ShowLines property controls the appearance of lines connecting TreeView control nodes.

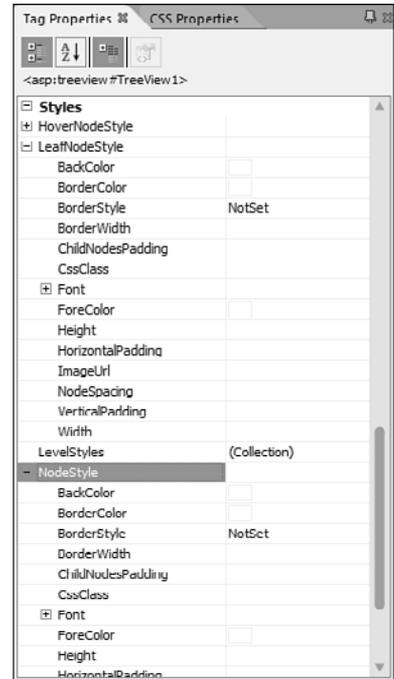
Using CSS Styles

The TreeView control provides a full range of CSS styles to make formatting with CSS easy. Figure 26.17 shows the CSS properties available for the TreeView control.

Just as with the Menu control, any AutoFormat options you choose for a TreeView will be applied using the CSS styles for the control. To tweak the configuration of formatting applied using the AutoFormat dialog, use the various style properties for the control.

Figure 26.17

The Tag Properties panel shows the many CSS styles available when using a `TreeView` control.



Using the ASP.NET SiteMapPath Control

The `SiteMapPath` control improves your site's navigation by providing users with a bread crumb-style navigation bar, as shown in Figure 26.18.

Unlike the `Menu` and `TreeView` controls, the `SiteMapPath` control doesn't require you to configure a data source as long as a `web.sitemap` file exists in the root of the site. Just drop the control on the page and it automatically uses the data in the `sitemap` file to create links.

To use the `SiteMapPath` control, do the following:

1. Open the `addins.aspx` page.
2. Place the insertion point on the page underneath the text you entered when the page was created.
3. Insert a `SiteMapPath` control from the toolbox.

That's all there is to it. After the `SiteMapPath` control is added to the page, it automatically connects to the `sitemap` file and displays the appropriate links.

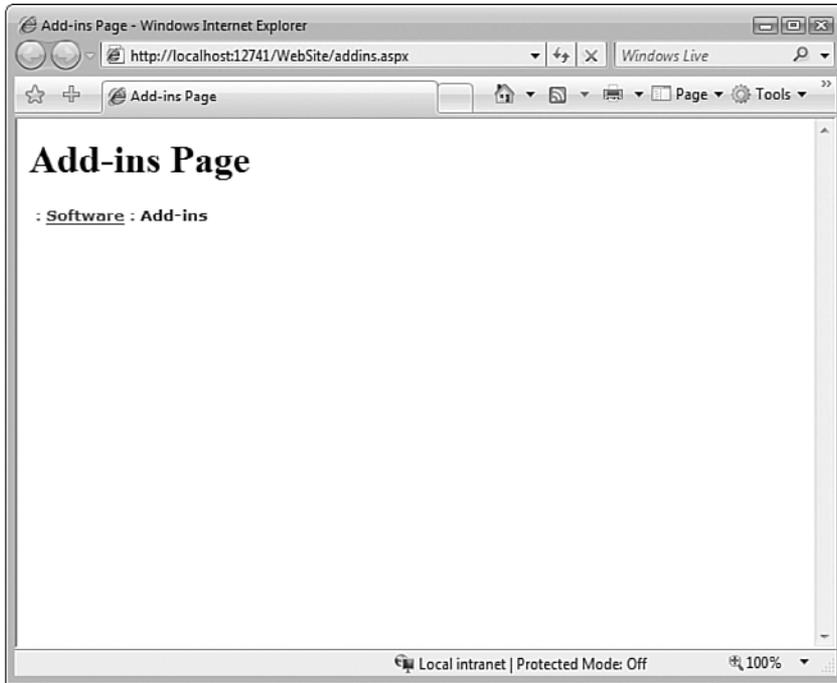


Figure 26.18
The SiteMapPath control is a convenient way to allow users to easily navigate your site. As a user navigates the site, the SiteMapPath control shows links to where she has been.

Formatting the SiteMapPath Control

As with the other controls we've looked at, the SiteMapPath control can be easily formatted using the AutoFormat dialog available from the SiteMapPath Tasks dialog. The AutoFormat schemes available are identical to the ones available for the Menu control.

In addition to the AutoFormat schemes, you can use several CSS style properties to affect the appearance of your SiteMapPath control, as shown in Figure 26.19.

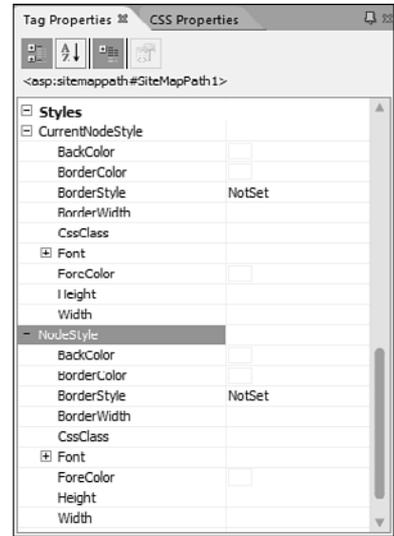
Using SiteMapPath Properties

Several properties affect the behavior and appearance of the SiteMapPath control:

- **ParentLevelsDisplayed property**—Controls the number of parent nodes displayed. If this value is set to the default value of -1, all parent nodes are displayed.
- **PathDirection property**—Controls the direction of the SiteMapPath control. The default value is RootToCurrent, which displays the root node at the left and the current location at the far right. The other valid setting is CurrentToRoot, which reverses the order of the nodes.
- **PathSeparator property**—Configures the character displayed between nodes. The default is the : (colon) character.

Figure 26.19

Just as with the other navigation controls, the `SiteMapPath` control has plenty of CSS style properties you can use to configure the look of the control.



The ASP.NET navigation controls make building a powerful and dynamic navigation system easy, and you can do it without writing a single line of ASP.NET code. You can, of course, significantly add to the functionality of these controls with ASP.NET code, but as you've seen in this chapter, it's not necessary to be a programmer to build perfectly functional site navigation with ASP.NET.

Improving Navigation with Master Pages

You've seen some powerful navigation features in this chapter. To make the best use of these navigation controls, however, you should consider using them with ASP.NET Master Pages.

➡ *For more information on using ASP.NET Master Pages, see Chapter 27, "Using ASP.NET Master Pages and User Controls."*

If you use navigation controls in combination with ASP.NET Master Pages, insert the navigation controls on one page only; ASP.NET automatically adds the controls to all your other pages when the pages are browsed. Using this method makes managing your navigation much more efficient because making a change is as easy as changing one page. Without Master Pages, a navigation change would require you to manually update every page that uses the navigation controls.

Note that you can also use Expression Web's Dynamic Web Templates if you choose. For a comparison of using Master Pages versus Dynamic Web Templates, see the sidebar "Master Pages Versus Dynamic Web Templates," Chapter 27.

➡ *For more information on using Dynamic Web Templates, see Chapter 19, "Using Dynamic Web Templates."*

This page intentionally left blank

USING ASP.NET MASTER PAGES AND USER CONTROLS

The Need for a Common Layout

Think for a moment about your Internet viewing habits. When you are clicking links on a site, how do you determine whether the information you are viewing is on the same site as the previous page? Most of you will probably agree that the most prevalent indicator is the style and layout of the site. In other words, if I'm viewing an article on the CNET site and I click a link, if the new page has the same layout and appearance as the previous page, my assumption is that it's the same site. That is exactly why a common layout is vital to the success of your site.

Web developers use many techniques to facilitate a common layout. One of the most common is the use of templates, such as Expression Web's Dynamic Web Templates. However, if you're using ASP.NET in your site, you might want to consider Master Pages instead because they are designed specifically for ASP.NET sites.

➔ *For more information on Dynamic Web Templates, see Chapter 19, "Using Dynamic Web Templates."*

The Master Page

A Master Page is a special kind of ASP.NET page that contains graphics, HTML code, ASP.NET controls, and code that is shared between pages.

ASP.NET Master Pages are similar to Dynamic Web Templates in that they facilitate the development of a common user interface and a common look and feel. However, they also represent a substantial step up from

Dynamic Web Templates because they fully support the ASP.NET code model and allow you to share not only user interface elements, but also code between pages.

Just like a Dynamic Web Template, you define areas where page-specific content will exist. When you create a page and attach it to a Master Page, you can add content on that page only in the areas that have been designated in the Master Page.

In a Dynamic Web Template, the areas where page-specific content can exist are called *editable regions*. In a Master Page, two ASP.NET controls are used instead: the Content control and ContentPlaceHolder control.

The ContentPlaceHolder control exists on the Master Page. Each ContentPlaceHolder control is mapped to a Content control on a page that is attached to the Master Page. That page is called the Content Page. A Master Page is not required to have ContentPlaceHolder controls on it, but without them, there is no way to add content to a page that uses that Master Page.

You can define all content in the Master Page if you want. The Content Page can then provide its own content or simply use the content that the Master Page provides. Expression Web fully supports all aspects of Master Pages, so you don't need to write any code to take advantage of this powerful feature.

To write server-side ASP.NET code for your Master Pages, you should use a tool made for that purpose, such as Microsoft Visual Web Developer Express Edition. Because of Expression Web's excellent support for ASP.NET, it's convenient to use both Expression Web and an ASP.NET application development tool in combination to build full-featured ASP.NET applications. However, Microsoft's inclusion of many of Expression Web's features in Visual Web Developer make it an easy transition for those who use Expression Web.

To create a new Master Page, select File, New, Page and select Master Page from the list of page templates, as shown in Figure 27.1.

After you've created a new Master Page, you should save it with a .master file extension. As I stated earlier, your Master Page doesn't have to contain a ContentPlaceHolder control, but if you save a Master Page that doesn't contain a ContentPlaceHolder control, Expression Web prompts you with a warning dialog, as shown in Figure 27.2.

 **note**

A Master Page has a file extension of .master. A Content Page, just like any other ASP.NET Web Form, has a file extension of .aspx. You cannot apply a Master Page to a page unless that page is an ASP.NET Web Form.

 **tip**

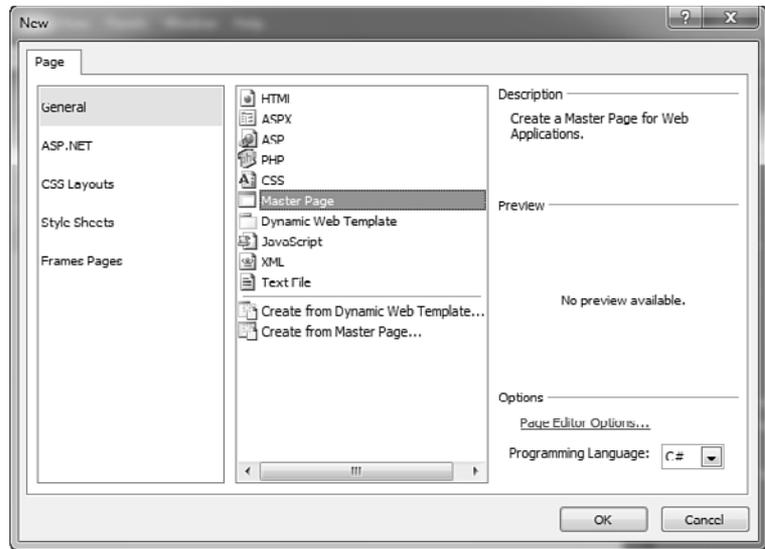
You can select either General or ASP.NET from the list of page types in the New dialog. The option to create a new Master Page exists from either one of these choices and there is no difference in the two. It's purely your choice.

 **tip**

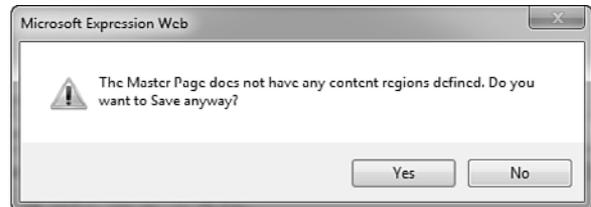
I think the message in Figure 27.2 is poorly worded. By calling a ContentPlaceHolder a "content region," Microsoft is making it easy to confuse ContentPlaceHolder controls with Content controls. Don't fall into that trap. Content controls exist on Content Pages. When Expression Web says "content region," what it really means is the ContentPlaceHolder control.

Figure 27.1

The New dialog contains a page template for a Master Page. When you select it, Expression Web creates a new Master Page with one ContentPlaceHolder control on it.

**Figure 27.2**

Expression Web enables you to save a Master Page that contains no ContentPlaceHolder controls, but it asks you just to be sure.



Let's have a quick look at the `<body>` of a Master Page:

```
<body>
  <form id="form1" runat="server">

    <asp:ContentPlaceHolder runat="server" id="contentplaceholder1">
      <p style="vertical-align: Top">This is content from the Master Page.</p>
    </asp:ContentPlaceHolder>

  </form>
</body>
```

This Master Page contains one ContentPlaceHolder control named contentplaceholder1. I've added some text to that ContentPlaceHolder control. The text I've added will also appear on any Content Pages, but because the content is inside a ContentPlaceHolder control, I can use my own content for that section in my Content Page. To understand what I mean by that, let's have a quick look at a Content Page.

The Content Page

A Content Page is a special kind of page. It cannot have any content in it unless that content is inside a Content control. When you first look at the HTML code of a Content Page, it will seem odd to see it completely empty except for the standard ASP.NET directives.

To create a new Content Page, select File, New, Page, and then select Create from Master Page in the New dialog, as shown in Figure 27.3.

After you click OK, you are prompted to select the Master Page, as shown in Figure 27.4.

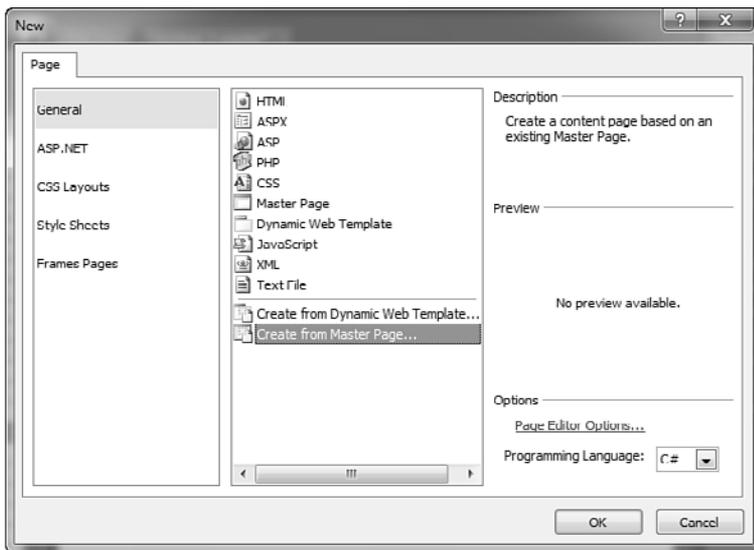


Figure 27.3
You can hook up a new ASP.NET Web Form to a Master Page by selecting the option to create a new page from an existing Master Page.

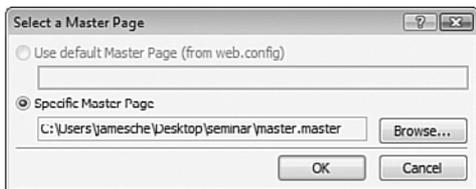


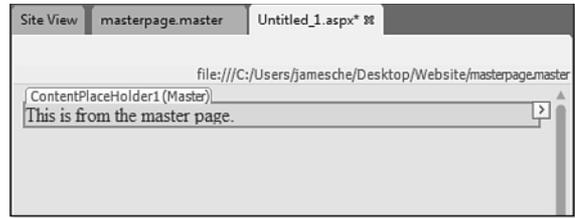
Figure 27.4
You must click Browse and select a Master Page. Expression Web won't let you just type the path.

After you select the Master Page and click OK, you see the new Content Page along with all the content it has pulled from the Master Page (see Figure 27.5).

Notice that the text that was in the ContentPlaceholder control in the Master Page is now located in a shaded area on the Content Page. That shaded area is the Content control. At the upper-left corner of the Content control is the ID of the ContentPlaceholder control on the Master Page that is feeding the Content control.

Figure 27.5

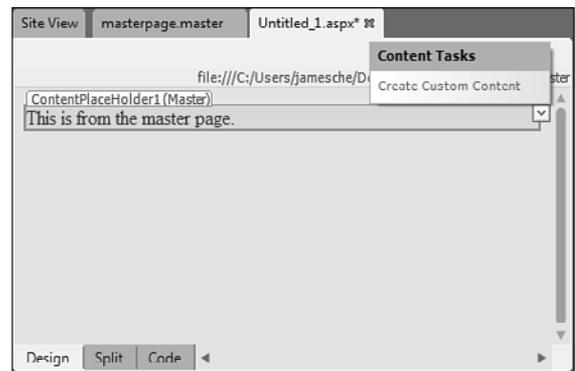
The new Content Page contains all the content from the Master Page. Any content in a Content control can be customized in the Content Page.



Note that Expression Web displays “(Master)” next to the ID, indicating that the Content control is currently configured to display content from the Master Page. To override that setting so you can change the content in the Content control, click the button at the upper right of the Content control and select Create Custom Content from the Content Tasks menu, as shown in Figure 27.6.

Figure 27.6

Click the button to display the Content Tasks pop-up. You can then choose to define your own content for the Content control.



When you click the Create Custom Content link, “(Master)” changes to “(Custom)” and you can click inside it and add your own content.

If you decide that you’d like the Content control to revert back to the content from the Master Page, you can simply click the button on the control to display the Content Tasks pop-up again, and then click the Default to Master’s Content link. When you do, Expression Web asks you to confirm your choice.

Figure 27.7 shows the Content Page from Figure 27.5 in Code View. Notice that the only code that exists in the Content Page is code that ASP.NET uses to identify properties of this particular page. No other HTML code appears in the page. Also notice that the name of the Master Page to which this page is attached appears at the upper-right corner of the page.

**tip**

Content controls always take up the entire width of a page unless they are inside a table or other container. They are also not resizable in Expression Web. They automatically resize themselves when a page is browsed.

If you want to manipulate the size of a Content control, insert it into a table cell. You can then resize the table cell to control the size of the Content control within it.

After custom content is defined for the page, Expression Web adds a Content control to the page and the HTML for the custom content is added inside the Content control, as shown in Figure 27.8.

```

1 <%@ Page language="C#" masterpagefile="masterpage.master" title="Content Page" %>
2

```

Figure 27.7
A Content Page contains nothing other than Content controls and the standard ASP.NET code at the top of the page. In this case, the page is getting all content from the Master Page, so no Content controls are present.

```

1 <%@ Page language="C#" masterpagefile="masterpage.master" title="Content Page" %>
2 <asp:Content id="Content1" runat="server" contentplaceholderid="ContentPlaceholder1">
3   <p>This is from the master page.</p>
4 </asp:Content>
5
6

```

Figure 27.8
Expression Web adds a Content control to the page when custom content is created. Notice the absence of standard HTML tags such as <head>, <html>, and <body>.



Parser Error When Browsing Content Page

If you're browsing your Content Page you might get an error in the page that says Parser Error Message: Only Content controls are allowed directly in a content page that contains Content controls. This error is poorly worded, but it means you cannot have any content on a Content Page other than Content controls. If you are seeing this error, you have text or an HTML tag that exists outside your Content controls.

Be sure you don't have any HTML tags such as <html> or <body> on your Content Page. These tags are provided by the Master Page and should not be included in your Content Page.

As a hint to what's wrong, the error page you are seeing will usually highlight the offending code in red inside a large yellow box.

Master Pages Versus Dynamic Web Templates

You might be wondering at this point whether you should select Master Pages or Dynamic Web Templates for your site. It actually depends on whether ASP.NET is available to you. If it is, Master Pages represent a significant improvement over the Dynamic Web Template model.

Master Pages, on the other hand, are not a design-time tool. A Master Page is combined with the content on a Content Page when a page is browsed. In a large site, this subtle difference is actually a huge convenience. It means that you don't have to spend a lot of time waiting on changes to thousands of pages when you update your Master Page. Simply make the update to your Master Page, save that change, and when you browse any Content Page that uses the Master Page, the new change will be present.

The other major advantage to using Master Pages (and perhaps the greatest advantage of all) is that Master Pages are ASP.NET pages and can run ASP.NET code. Even if you aren't currently writing any ASP.NET code for your site, it's still a wise move to be prepared to add that functionality to your site with the least amount of hassle. If your architecture is already running on ASP.NET, enhancing it with code at a later date will be a simple task.

If you aren't yet convinced that ASP.NET is a technology you should be using now, you will be after you finish the upcoming chapters. There's no reason you should not focus your site development on this technology, and using Master Pages is a big step in the right direction.



caution

If you change the content in a Content control and then later choose to revert to the content from the Master Page, all the custom content for the Content control will be lost. Even if you then switch back to display the custom content, the previously entered content will be gone.

Make absolutely sure that you want to revert to the Master Page's content before you confirm that choice. When in doubt, save the content to a temporary page first so you can recover it later if you change your mind.

Developing a Master Page Site

Now that you have a good overview of how Master Pages work, let's create a simple site using Master Pages.

Creating the Master Page

The Master Page will contain a table that controls the layout of the site. To create the Master Page, follow these steps:

1. Create a new Master Page by selecting File, New, Page and selecting Master Page from the list of available page types.
2. Click OK to create the Master Page.



note

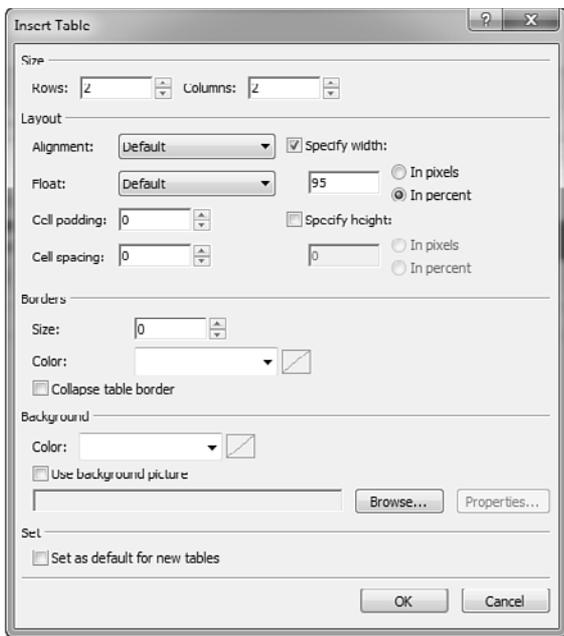
For this exercise, you will use the files downloadable from the website that accompanies this book at informit.com. You'll also find the completed Master Page site at that location.

3. Click the ContentPlaceHolder control to select it; then delete it from the page. If you don't see the ContentPlaceHolder control, select View, Visual Aids, Show to turn on Visual Aids.
4. Click Table, Insert Table.
5. In the Insert Table dialog, configure the table as follows:
 - Set both the Rows and Columns to 2.
 - Set the Width to 95 and select the In Percent radio button.
 - Set both cellspacing and cellpadding to 0.
 - Set the border size to 0.

The Insert Table dialog should now look like Figure 27.9. Click OK to insert the table.

**tip**

The default ContentPlaceHolder control is deleted from the page at this point because anything that exists within a ContentPlaceHolder control can be customized by the Content Page. When you are creating a Master Page, it's often preferable to delete the default ContentPlaceHolder control and then insert new ContentPlaceHolder controls after you have defined where you want custom content to exist. That's the method we'll use in this chapter.

**Figure 27.9**

The table you are configuring here will define the layout for the Master Page.

➔ For more information on inserting and configuring tables, see Chapter 5, “Using Tables.”

The table you just inserted will hold the common user interface elements for the site. We need to add a logo and change the width of the left column.

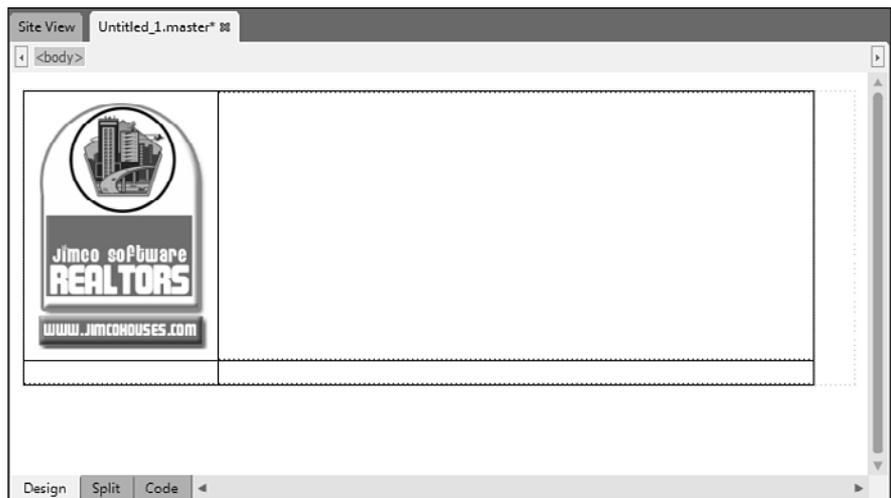
Copy the file `logo.jpg` from this book's website to your hard drive and insert it into the upper-left cell. Drag the right border of the column containing the logo so that the column is 160px wide.

Your table should now look like Figure 27.10. Note that I've formatted the table in Figure 27.10 with a black border so that you can more easily see the table cells. Your table will not have a border.

note

CSS is actually preferred over tables for controlling layout. However, since we're more interested in learning to use Master Pages in this chapter, I've used a table in the example to make it more simple.

Figure 27.10
This table will be used to lay out your site's structure.



➔ *For more information on working with images in Expression Web, see Chapter 9, “Using Graphics and Multimedia.”*

At this point, save the page as `master.master`. When you do, Expression Web warns you that no content regions exist. Click **Yes** to save the page anyway. We'll add a content region shortly.

We have a couple of other areas to configure for the Master Page. Next, we'll add a menu down the left edge of the page and an ASP.NET AdRotator control in the top area. Do the following:

1. Place the insertion point in the cell below the logo.
2. Insert the image `menu_top.jpg` into the cell.
3. Press **Shift+Enter** on your keyboard to insert a new line break after the image.

note

We won't add a menu to the page in this example. Instead, we'll use the word `MENU` as a placeholder. In a real-world application, you would add a menu—or possibly an ASP.NET Menu control—to the left side of this page.

4. Insert a new table with a width of 100%, 1 column and 1 row, a border of 0, and cellpadding and cellspacing values of 0.
5. Type **MENU** into the new table.
6. Insert the image `menu_bottom.jpg` under the table you just created.
7. Right-click in the cell with the menu images and select Cell Properties from the menu.
8. Click the drop-down next to Background Color and select More Colors from the menu.
9. Click the Select button in the More Colors dialog, as shown in Figure 27.11.

 **note**

The table you inserted into the menu cell prevents any menu text or control from extending beyond the width of the cell. Without the table in the cell, a menu could cause the cell to automatically widen. Because we've set a background color for the cell, having it extend beyond the width of the graphics would cause an unprofessional appearance.

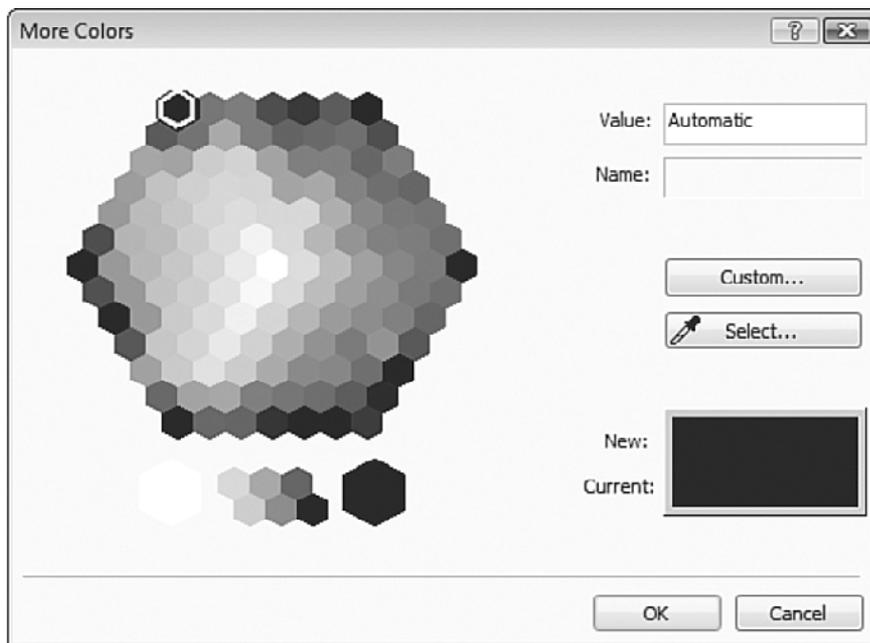


Figure 27.11
The More Colors dialog provides additional tools for selecting colors. In this case, the eye dropper is used by clicking the Select button.

10. Place the color dropper over the blue color on the logo and click to select that color.
11. Click OK in the More Colors dialog to finish selecting the color.
12. Click OK in the Cell Properties dialog to apply the new color.

Your page should now look like Figure 27.12.

Figure 27.12

The column for the menu has been formatted with some images and a background color for an attractive look.



We're now ready to configure the top row of the table. Remember that we want to add an ASP.NET AdRotator control to this page. We'll use the standard banner size of 468 pixels wide by 60 pixels high. Let's add a new table now to contain the AdRotator control. Follow these steps:

1. Right-click inside the upper-right cell in the table and select Cell Properties.
2. Change the Vertical Alignment to Top and click OK.
3. Select Table, Insert Table.
4. Change Rows and Columns to 1, change the width to 468, and select the In Pixels radio button.
5. Change both Cell Spacing and Cell Padding to 0 and set the Alignment to Center.
6. Click OK to insert the table.

Your page should now look like the one shown in Figure 27.13.

➔ *For more information on using the ASP.NET AdRotator control, see Chapter 25, "Using Standard ASP.NET Controls."*

**Figure 27.13**

The page now contains a table suitable for inserting an AdRotator control to show an ad banner.

We now need to define the areas of the Master Page that we will allow to be customized by the content in the Content Page. In this set of steps, we create two areas of customizable content: the main content area and an area along the right edge of the page:

1. Right-click in the cell just to the right of the menu cell and select Cell Properties from the menu.
2. Change the vertical alignment of the cell to Top and click OK.
3. From the Standard ASP.NET controls section of the Toolbox, add a new ContentPlaceHolder control to the cell.
4. If the Tag Properties panel is not visible, select Task Panes, Tag Properties to activate it.
5. If the ContentPlaceHolder control is not selected, click it to select it.
6. In the Tag Properties panel, change the ID of the ContentPlaceHolder control to MainContent.
7. Right-click the cell containing the MainContent ContentPlaceHolder control and select Insert, Column to the Right.

**tip**

We won't add any default content to the ContentPlaceHolder controls on the Master Page. However, in a real application, you might want to add content (perhaps something like advertisements) to the right edge of the Master Page. That way, page developers using your Master Page can decide to leave your default content in place or define their own content.

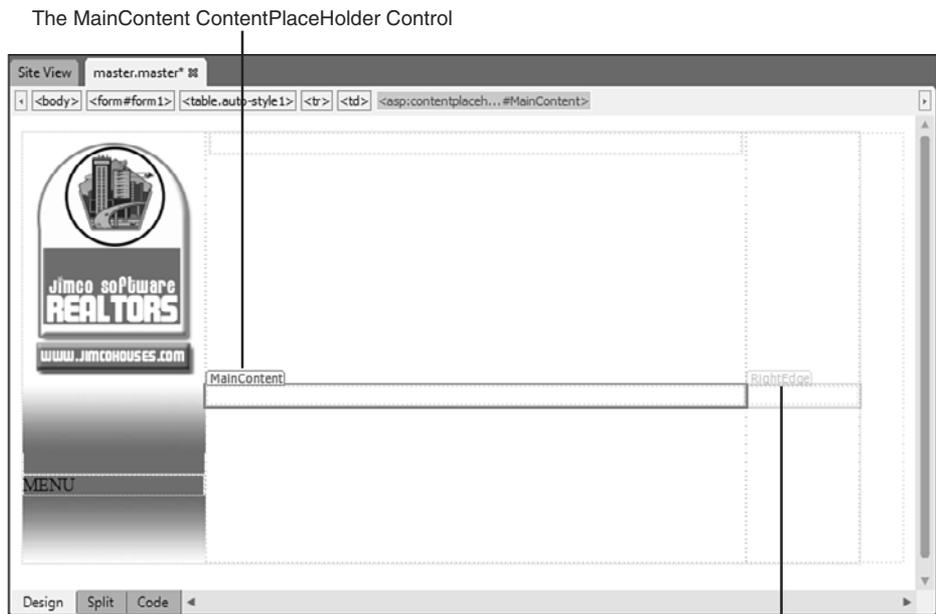
8. Right-click inside the bottom cell of the newly created column and select Cell Properties.
9. Change the vertical alignment of the cell to Top.
10. Change the horizontal alignment of the cell to Right.
11. Change the width of the cell to 100px.
12. Click OK.
13. Add a new ContentPlaceHolder control to the cell.
14. In the Tag Properties panel, change the ID of the ContentPlaceHolder control to RightEdge.

Your Master Page should now look like Figure 27.14. Save the Master Page.

note

Even though Expression Web labels this area as MainContent, you are not actually editing the MainContent ContentPlaceHolder control. Instead, you are adding content into a Content control that is mapped to the MainContent ContentPlaceHolder control on the master.master page.

Figure 27.14
The Master Page is now complete with two ContentPlaceHolder controls for custom content.



The RightEdge ContentPlaceHolder Control

Creating the Content Page

We're now ready to create a Content Page that uses the Master Page we just created. Here's how:

1. Select File, New, Create from Master Page.

2. Click Browse and select the `master.master` page you just finished creating.
3. Click Open to select the master page and click OK to create the new Content Page.

Save the new Content Page now as `default.aspx`.

If you look at the Code View for the new Content Page, you'll see that there is almost no code in it. At this point, the Master Page is defining all the content for this page. The only things that actually exist on the Content Page are the two Content controls that map to the ContentPlaceHolder controls on the Master Page.

Using the following steps, we define some custom content for the new Content Page:

1. Click inside the `MainContent` area to select it.
2. Click the button at the upper right of the control and select Create Custom Content from the Content Tasks pop-up.
3. Enter some text inside the `MainContent` area. As shown in Figure 27.15, the Content control resizes accordingly as you enter content.



Figure 27.15
The Content control changes size and shape depending on the content added to it.

Notice that you are unable to edit any portion of the Content Page that falls outside a Content control. Master Pages are a powerful method of enforcing a common user interface while still allowing developers the flexibility to design content the way they want.

Save the Content Page and browse it using the Microsoft Expression Development Server. Notice that the content you entered into the Content control is displayed along with the interface elements from the Master Page.

If you open the Master Page and make any changes to the layout or content, those changes take effect immediately in all Content Pages that use that Master Page. Unlike Dynamic Web Templates, Master Pages do not require an explicit update of attached pages.

Gaining an understanding of Master Pages is valuable because it positions you to not only design your own sites using Master Pages, but also to contribute to a site development team that uses Master Pages. As previously mentioned, one of the primary reasons to use Master Pages is so that programmers can work independently from designers. With a firm knowledge of how Master Pages work, you can efficiently use Expression Web to take advantage of this powerful feature.

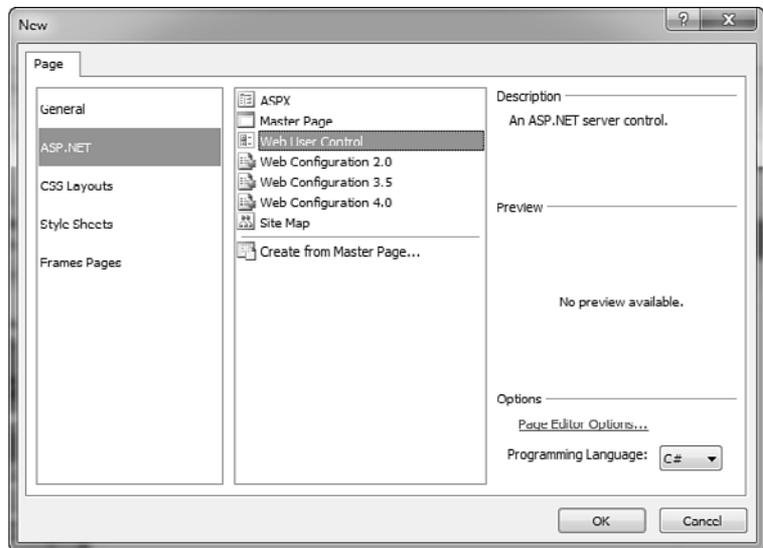
Extend Reusability with ASP.NET User Controls

Master Pages are a convenient way to provide consistent content across multiple pages. However, in some cases, you might want to reuse only a portion of a page. For example, in this chapter, you created a space at the top of the Master Page for an `AdRotator` control. Instead of using one `AdRotator` control for all of your pages, you might want to have several different `AdRotator` controls configured for a specific type of ad. In such a scenario, creating a few ASP.NET user controls with `AdRotator` controls on them makes it easy.

An ASP.NET user control is created in the same way that you create an ASP.NET Web Form. Once you've created a user control, you can add it to a page by simply dropping it on the page from the Expression Web Folder List.

To create a user control, select File, New Page and select the ASP.NET section in the New dialog. Select Web User Control as shown in Figure 27.16 and click OK.

Figure 27.16
Expression Web makes it easy to create a new ASP.NET user control.



You can now add content to your user control just as you would an ASP.NET Web Form. Once you've completed adding your content, save the user control. Expression Web saves the user control with a .ascx file extension.

User controls are not intended to be browsed directly. In fact, if you try to browse directly to a user control, ASP.NET generates an error message. For that reason, Expression Web does not let you preview a user control in the browser. To view your user control, you must first add it to a page.

To add a user control to a page, drag the .ascx file from the Folder List to the point in your page where you want the user control to appear. You can then edit the user control by selecting Edit UserControl from the UserControl Tasks pop-up that appears as shown in Figure 27.17.

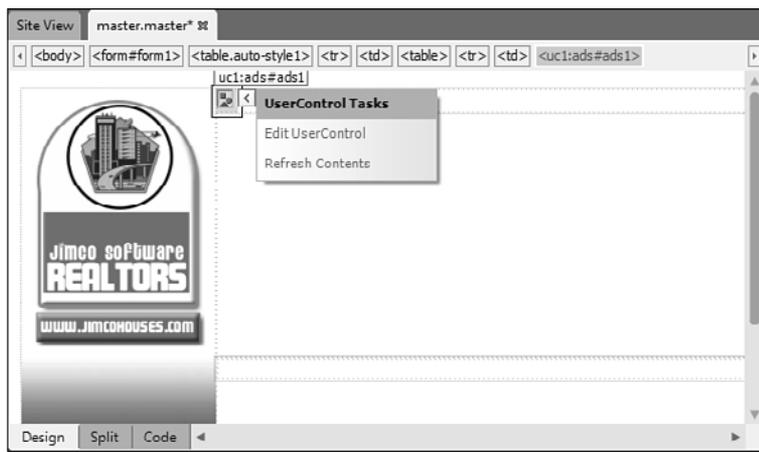


Figure 27.17

After you've inserted a user control, you can edit it by selecting Edit UserControl from the UserControl Tasks pop-up.

After you've created a user control, you can reuse it on any page by simply dropping it on the page. This capability makes user controls another powerful tool for reusing page content.

DEVELOPING A LOGIN SYSTEM USING ASP.NET

Website Login Systems

There are generally two types of sites. One type offers the same information to everyone, and the other type offers information that is somehow specific to the person using the site. It's becoming more and more common to see sites that fall into the latter category, and because of that, it's becoming more necessary for web developers to be able to design a method of allowing users to log in to a site so they can be uniquely identified.

There are many ways to implement a login system, but almost all of them require a significant amount of code. Not only that, but writing secure code for a login system is difficult because it requires a full understanding of security issues.

Fortunately, ASP.NET has a full suite of login controls that provide an amazing level of functionality without having to write a single line of code. You're probably picturing a simple system with a basic username and password implementation. Think again! ASP.NET login controls provide you with all the features needed for most applications.

By default, ASP.NET uses SQL Server Express Edition for storing login information. Before you begin the process of developing a login system, keep in mind that most hosting companies don't support SQL Server Express Edition. Therefore, you should use the exercises in this chapter to familiarize yourself with how everything works and then contact your hosting company to find out what it offers for storing ASP.NET membership data.



note

Many hosting companies do offer tools for moving ASP.NET login information from a SQL Server Express Edition database to a full-blown SQL Server database. Check with your hosting company.

Overview of ASP.NET Login Controls

Expression Web provides access to all seven ASP.NET login controls via the Login section of the ASP.NET controls Toolbox.

➔ *For more information on the Microsoft Expression Development Server, see Chapter 33, “Using the Microsoft Expression Development Server.”*

The Login Control

The Login control, like most of the other ASP.NET login controls, is simple in appearance but robust in functionality. In its simplest terms, the Login control provides users with text boxes for entering a username and a password, as shown in Figure 28.1. A Remember Me check box is also provided so users don't have to log in on each subsequent visit to your site.

caution

Security should be among your highest priorities when designing a web application. You should be aware that, by adding features such as login functionality, you are exposing yourself to security risks.

If you are going to use ASP.NET's login controls in your site, I highly recommend that you visit Microsoft's page on securing the login controls. You can access this page by browsing to <http://msdn2.microsoft.com/en-us/library/ms178346.aspx>.

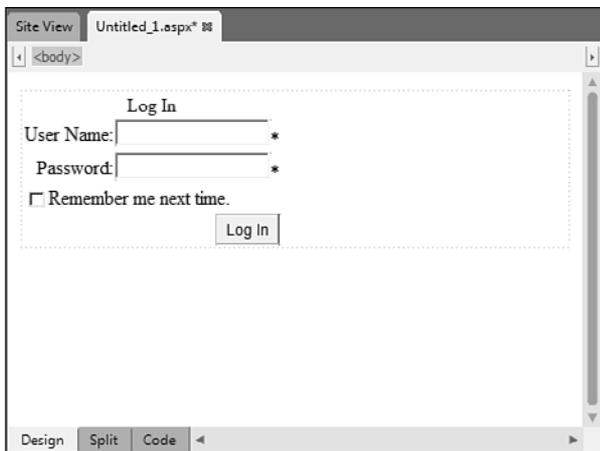


Figure 28.1

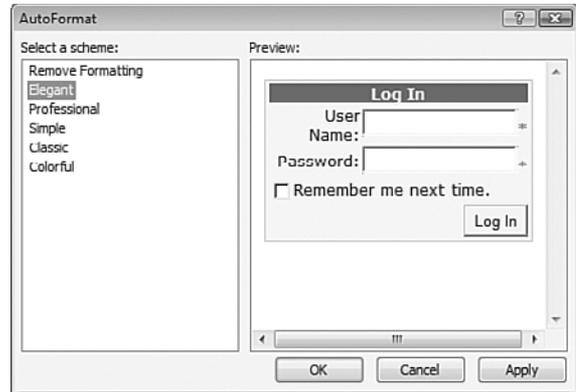
The Login control is simple in appearance, but there's a lot of functionality under the hood.

The Login control also implements form validation so users cannot submit a form without supplying both a username and a password.

Figure 28.1 shows the default appearance of the Login control. You can reformat it using CSS styles or by using the AutoFormat option on the Login Tasks pop-up. By choosing one of the six available formatting options, as shown in Figure 28.2, you can quickly change the appearance of the entire control.

Figure 28.2

You can quickly give the Login control a new look using the AutoFormat dialog.



Because the Login control is a single control, you cannot move the elements in the control by default. If you want to rearrange the controls that make up the Login control, you'll need to convert the control to a *template*, which is referred to as a *templated* control. By converting the control to a template, you have access to all the constituent controls that make up the Login control. All the functionality remains the same.

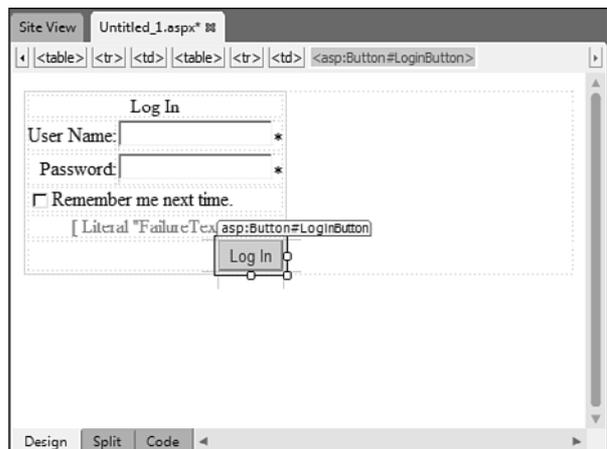
**note**

The ASP.NET membership system uses Microsoft SQL Server Express Edition by default to store users for your site. Therefore, to walk through the examples in this chapter, you need to install Microsoft SQL Server Express Edition.

The easiest way to get everything you need is to install Microsoft Visual Web Developer Express Edition. You can find Visual Web Developer Express Edition at www.microsoft.com/express/vwd. However, you can also just install SQL Server Express Edition from <http://www.microsoft.com/express/Database/>.

Figure 28.3

Converting the Login control to a template allows access to the controls that comprise it. Notice here that the Button control is selected. Accessing the button is impossible before converting the control to a template.



To convert the Login control to a regular control instead of a template, click the button on the control to access the Login Tasks pop-up and click the Reset option. Any changes you made to the control while it was in a templated state are reset, and the control takes on its default appearance.

Converting a control to a template is more powerful than it might seem at first. As you will see shortly, many properties for the Login control enable several other user interface elements. For example, the control can display links for help text or user registration. By converting the control to a template, you have full control over the placement of all parts of the control.

The built-in functionality of the Login control isn't limited to the appearance of the control. When the Log In button on the control is clicked, ASP.NET automatically does a lookup on the user and authenticates him. If the user presents incorrect credentials, the control notifies the user. If the credentials are correct, the control logs in the user and redirects him to the page specified in the `DestinationPageUrl` property of the control.

Many other useful properties of the Login control can be accessed via the Tag Properties panel:

- `CreateUserIconUrl`—Specifies a URL to a graphic file that acts as a link for new users who are not yet registered. When clicked, users are taken to a registration page for the site.
- `CreateUserText`—Text that is displayed that acts as a link for new users who are not yet registered. This is most often used in place of the `CreateUserIconUrl`.
- `DestinationPageUrl`—The URL to which users are redirected upon a successful login.
- `DisplayRememberMe`—If true, displays the Remember Me check box. This value is true by default.
- `FailureAction`—This property can be one of two values: `Refresh` or `RedirectToLoginPage`. It controls what happens if a user fails to log in successfully. The `RedirectToLoginPage` is used in situations where the Login control is placed on a page other than the login page.
- `FailureText`—Configures the text that is displayed if a user fails the login attempt. It is recommended that this text be chosen carefully so as not to cause a security risk. For example, notifying a user that a password is incorrect can imply that the username is valid, leading to a potential security risk.
- `HelpPageIconUrl`—Sets the URL of a graphic that can be displayed as a help icon. When the user clicks this graphic, she is redirected to a help page for your site.
- `HelpPageText`—Configures the text that is displayed as a link to the help page for the site.
- `HelpPageUrl`—Specifies the URL of the help page for the site.

**tip**

Those of you using an operating system without a web server are not excluded from testing the concepts described in this chapter. The Microsoft Expression Development Server provides a fully supported environment for ASP.NET.

**tip**

To convert the Login control to a template, click the button at the upper right of the control to access the Login Tasks pop-up and click the Convert to Template link. After you do this, the controls within the Login control immediately become accessible, as shown in Figure 28.3.

- **InstructionText**—Sets the text that appears to offer instructions to users.
- **LoginButtonImageUrl**—Specifies the URL of an image to use for the login button. The **LoginButtonType** property should be set to **Image** when using this property.
- **LoginButtonText**—Allows you to change the text that appears on the button. By default, the button displays the text **Log In**.
- **LoginButtonType**—This property can be one of three values: **Button**, **Image**, or **Link**. **Button** is the default. When set to **Link**, the button is converted to a regular link. When set to **Image**, the image specified by the **LoginButtonImageUrl** property is displayed.
- **Orientation**—Sets the orientation of the control to either **Horizontal** or **Vertical**.
- **PasswordLabelText**—Sets the text that labels the Password text box.
- **PasswordRecoveryIcon**—Used to specify an image to display for a link to a password recovery page. A user can click this when she forgets her password.
- **PasswordRecoveryText**—Sets the text to display for the password recovery link.
- **PasswordRecoveryUrl**—Sets the URL of the password recovery page.
- **PasswordRequiredText**—The text entered here is displayed in the **ValidationSummary** control on the Login control when no password is entered.
- **RememberMeSet**—Sets this property to **true** to check the Remember Me check box by default.
- **RememberMeText**—Sets the text for the Remember Me check box.
- **TextLayout**—Can be set to **TextOnLeft** or **TextOnTop**. This controls the position of the text for the text boxes.
- **TitleText**—The text that appears in the title bar of the control.
- **UserName**—Sets the initial text of the Username text box.
- **UserNameLabelText**—Sets the text that labels the Username text box.
- **UserNameRequiredError**—Sets the text that is displayed in the **ValidationSummary** control on the Login control when no username is entered.
- **VisibleWhenLoggedIn**—When true, the Login control is visible even when the user is logged in. This property is normally used only when the Login control appears on a page other than the login page.

As you can see from that rather long list of properties, the Login control is more powerful than it seems at first.

**note**

The Tasks pop-ups for the login controls have an **Administer Website** link. If you click this link, you are informed that administering the site is not supported. If you want to administer the site, you need to use the tools that ASP.NET provides.

LoginStatus Control

The LoginStatus control has two views: one view for when the user is logged in and one for when the user is logged out. As shown in Figure 28.4, the view of the control can be configured using the drop-down available in the LoginStatus Tasks pop-up.

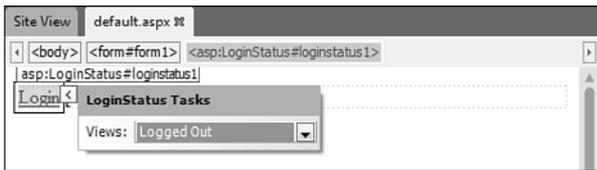


Figure 28.4

ASP.NET automatically chooses the correct view for the LoginStatus control based on whether the user is authenticated, but you can select the view yourself if you want.

Some of the useful properties of the LoginStatus control are as follows:

- **LoginImageUrl**—Specifies the URL of an image to display for the login image. When a URL is specified in this property, the link in the control changes from text to the image specified.
- **LoginText**—Specifies the text that appears for the login link.
- **LogoutAction**—This property can be set to one of three values: Refresh, Redirect, or RedirectToLoginPage. When set to Redirect, users are redirected to the URL in the LogoutPageUrl property when logging out.
- **LogoutImageUrl**—Specifies the URL of an image to display for the logout image. When a URL is specified in this property, the link in the control changes from text to the image specified.
- **LogoutPageUrl**—The URL to which users are redirected upon logging out when the LogoutAction is set to Redirect.
- **LogoutText**—The text that is displayed for the logout link.

LoginName Control

The LoginName control is perhaps the simplest of the login controls. It displays the name of the logged-in user.

By default, the control displays only the username. By altering the FormatString property, you can alter the text that is displayed. For example, if a user named Jim is logged in to the site and the FormatString property is set to Logged in as {0}, the LoginName control would display the text “Logged in as Jim.”

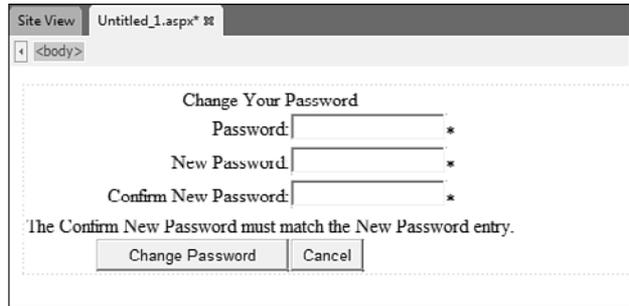
ChangePassword Control

The ChangePassword control allows a user to change his password (see Figure 28.5). In the default configuration, users are required to be logged in to change their passwords. However, the ChangePassword control can be configured to also prompt for the username. In those situations, a user can change his password without logging in. In fact, when the username field is displayed on

the control, a user can change the password of a different user as long as he knows the old password of the user whose password is being changed.

Figure 28.5

The `ChangePassword` control lets a user easily change his password. In some configurations, a user can even change someone else's password.

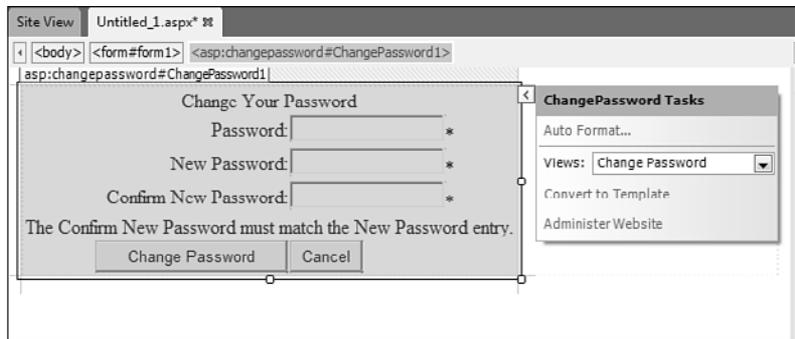


The screenshot shows a web browser window with the title "Site View" and "Untitled_1.aspx". The main content area displays a form titled "Change Your Password". The form contains three input fields: "Password:", "New Password:", and "Confirm New Password:", each followed by an asterisk. Below the fields is a message: "The Confirm New Password must match the New Password entry." At the bottom of the form are two buttons: "Change Password" and "Cancel".

Similar to the `Login` control, the `ChangePassword` Tasks pop-up shown in Figure 28.6 provides you with an `AutoFormat` link for easy formatting of the control and a link to convert the control to a template for more precision over the layout of the control.

Figure 28.6

The `ChangePassword` Tasks pop-up offers options similar to the `Login` control with the addition of a `Views` drop-down for switching between the two views provided by the control.



The screenshot shows the same "Change Your Password" form as in Figure 28.5, but with a "ChangePassword Tasks" pop-up menu open on the right side. The pop-up menu has the following options: "Auto Format...", "Views: Change Password" (with a dropdown arrow), "Convert to Template", and "Administer Website". The browser's developer tools are visible at the top, showing the HTML structure of the form.

The `ChangePassword` control also includes a `Views` drop-down that allows you to switch between the `Change Password` view and the `Success` view. The `Success` view is displayed when a user has successfully changed his password. Figure 28.6 shows the `Change Password` view of the control, and Figure 28.7 shows the `Success` view.

The `ChangePassword` control has many properties that enable you to customize the control's appearance and behavior. Because the list is long, we won't go over the properties that are similar to corresponding properties of the `Login` control. Refer to that section of this chapter to refresh your memory if necessary.

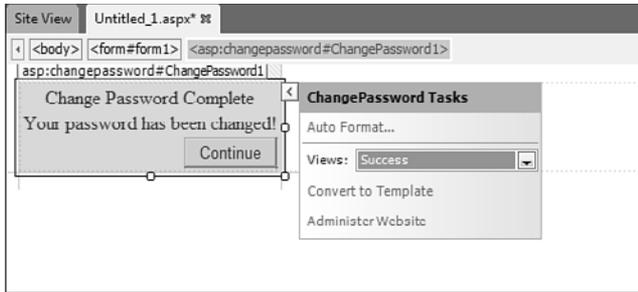


Figure 28.7
When a user has successfully changed his password, the Success view is displayed.

DisplayUserName Property

The `DisplayUserName` property controls whether a text box for entry of a username exists on the control. If the `DisplayUserName` property is `true` and a user is already logged in, the user will be able to change the password of another user assuming she knows the other user's password. The `ChangePassword` control shown in Figure 28.8 has a `DisplayUserName` property of `true`.

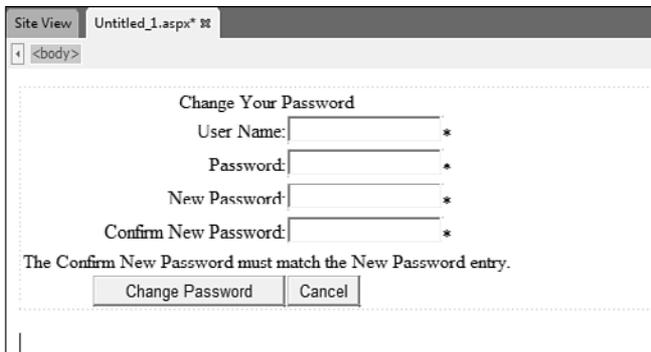


Figure 28.8
You can display a text box for a user's username by setting the `DisplayUserName` property to `true`.

By placing the `ChangePassword` control on a page that can be browsed without logging in to the site (a page that can be browsed anonymously), a user can change her password without logging in to the site, provided the `DisplayUserName` property is set to `true`.

MailDefinition Property

The `MailDefinition` property provides the capability to send users an email after a successful password change. The `MailDefinition` property is actually a reference to a special object in the .NET Framework called `MailDefinition`. As shown in Figure 28.0, the `MailDefinition` property expands into several different fields:

- **BodyFileName**—The `BodyFileName` field enables you to specify a file that will be used for the body of the email sent by the `ChangePassword` control. Any occurrence of `<%UserName%>` in this file will automatically be replaced by the username of the user, and any occurrence of `<%Password%>` will automatically be replaced by that user's new password.
- **CC**—Email address(es) that should be copied on the email. If you'd like to enter more than one address, separate the email addresses with commas.
- **EmbeddedObjects**—The `EmbeddedObjects` property is a collection of images or graphics to be embedded into the email. When you click the button next to this property, the `EmbeddedMailObject` Collection Editor appears, as shown in Figure 28.10. The code to insert these objects into the email is included in the file specified by the `BodyFileName` property.

The following code inserts the embedded object shown in Figure 28.10:

```

```

- **From**—The `From` property specifies the originating email address for the email.
- **IsBodyHtml**—This property specifies whether the file specified by the `BodyFileName` property contains HTML code or plain text. If you have set the `EmbeddedObjects` property, you should set the `IsBodyHtml` property to `true`.
- **Priority**—The priority at which the email will be sent. Valid values are `Normal`, `Low`, and `High`.
- **Subject**—The subject of the email.

PasswordHintText Property

The text specified for the `PasswordHintText` property appears above the `Password` text box. It is used to communicate your password requirements to users.

The default requirements of passwords specify that they be at least seven characters in length with at least one nonalphanumeric character.

note

We're delving into some basic programmatic concepts as we discuss the `MailDefinition` property. Because this isn't a programming book, we won't go into great detail on this property. I'll explain how it's used, but if you want the down-and-dirty details, your best resource is the MSDN site at <http://msdn2.microsoft.com>.

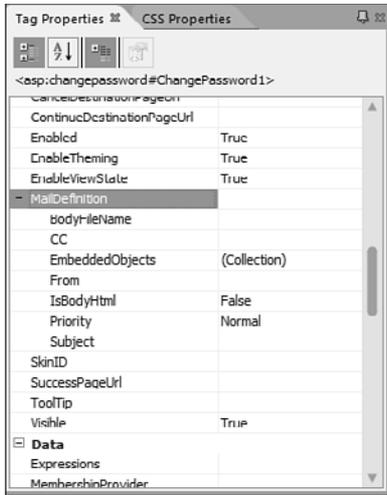
The details on the `MailDefinition` class can be found by browsing to <http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.maildefinition.aspx>.

tip

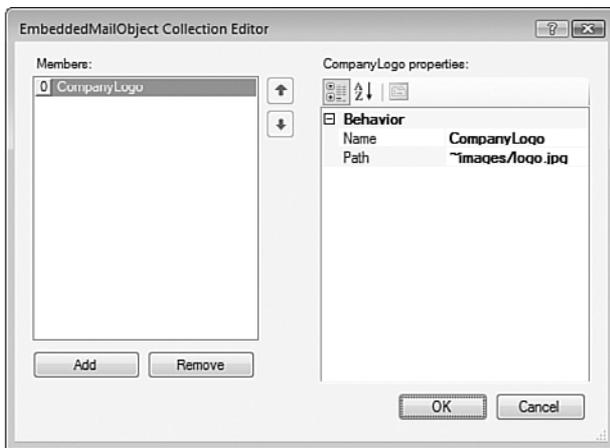
To send email, the computer on which the site resides must have the SMTP service configured on it. You can get information on how to configure SMTP by browsing to <http://msdn.microsoft.com/en-US/library/8b83ac7t.aspx>.

caution

When using the `MailDefinition` property, keep in mind that email can be intercepted fairly easily by someone other than the intended recipient. For that reason, you may want to consider using SSL for pages that use sensitive information.

**Figure 28.9**

The MailDefinition property is actually a collection of many fields that make defining an email to send to users upon a password change easy.

**Figure 28.10**

Embedding images in your email is accomplished via the EmbeddedMailObject Collection Editor. In this case, a company logo is being added.

PasswordRecoveryUrl Property

The PasswordRecoveryUrl property points to the page containing a PasswordRecovery control so that a user can recover or reset his existing password. (We'll cover the PasswordRecovery control later in this chapter.)

This property uses the PasswordRecoveryText or PasswordRecoveryIconUrl property to create the link.



note

We won't go into the specifics of configuring ASP.NET's behavior when storing passwords. If you'd like more information, Microsoft has excellent documentation on the MSDN site at <http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.passwordrecovery.membershipprovider.aspx>.

SuccessPageUrl Property

The `SuccessPageUrl` property is used when you want a user to be redirected to a specific page when her password has been successfully changed.

If this value is not specified, the `ChangePassword` control switches to the `Success` view upon a successful password change. If the `SuccessPageUrl` property is specified, the `Success` view of the `ChangePassword` control is not displayed at all.

PasswordRecovery Control

The `PasswordRecovery` control is designed to allow a user to gain access to your site even in cases of a forgotten password. The name of the control is a bit misleading. By default, a user cannot recover his password. Instead, a new one is assigned and sent in an email because ASP.NET uses an irreversible hashing algorithm before storing a password by default. You can change this behavior and store passwords encrypted. When a password is encrypted, it can be decrypted and recovered for the user.

As with the other controls we've reviewed, the `PasswordRecovery` control can be auto-formatted for a more pleasing appearance using the `AutoFormat` dialog shown in Figure 28.11. This dialog is accessible via the `PasswordRecovery` Tasks pop-up. The `PasswordRecovery` control is also a templated control, so you can rearrange the layout if you want.



tip

The view that is initially displayed for the `PasswordRecovery` control is the view selected in Expression Web when the page is saved. Therefore, you'll want to be sure you select the `UserName` view before saving your page.

Figure 28.11

The `PasswordRecovery` control can be formatted for a more pleasing appearance using the `AutoFormat` dialog.



When a user changes his password, the process actually occurs in a series of three steps. In the first step, the user is asked for his username. After the `Submit` button is clicked, the user is asked for the answer to his secret question provided when he registered for the site. Assuming the answer is correct, the user is presented with a message indicating that the password is being emailed.

Each of these steps is accessible in Expression Web using the Views drop-down in the PasswordRecovery Tasks pop-up, as shown in Figure 28.12. Because the control is templated, you can convert one or more of these steps to a template and control the layout and other functionality of each step.

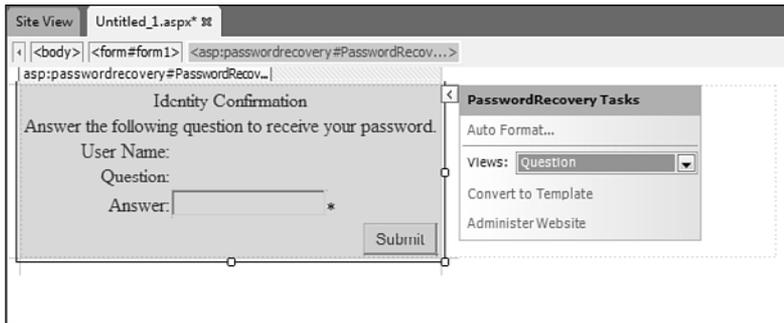


Figure 28.12

The PasswordRecovery control consists of three views to allow a user to retrieve or reset a password. The Question view is shown here.

The properties of the PasswordRecovery control are similar to the properties for the other login controls we've covered, so there's no need to cover the details here.

CreateUserWizard Control

The CreateUserWizard control is a wizard-based control that lets new users of your site easily create an account so they can log in (see Figure 28.13). The CreateUserWizard control is actually a customized ASP.NET Wizard control, so the configuration options available are the same options used for the Wizard control.

➔ *For more information on the Wizard control, see Chapter 25, "Using Standard ASP.NET Controls."*

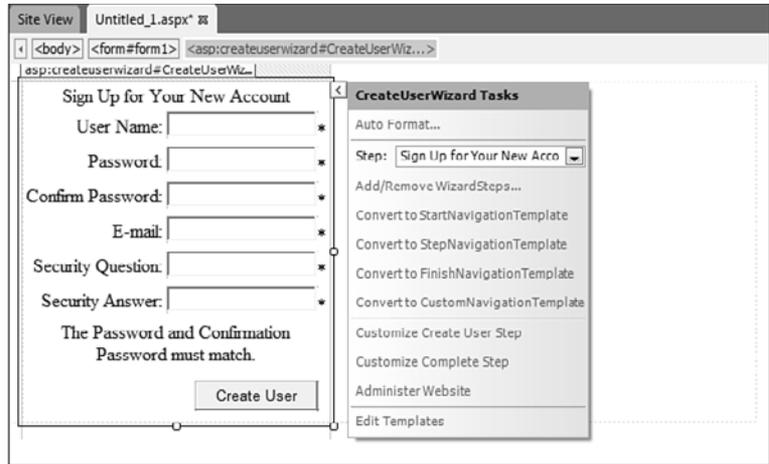
Several properties are unique to the CreateUserWizard control. The following are a few that affect the behavior of the control:

- **DisableCreatedUser**—By default, after a user creates a new account, she can immediately log in to the site. By setting the DisableCreatedUser property to false, the user will be prevented from logging in. This is commonly used when an administrator or moderator is required to approve an account before it is granted access.
- **LoginCreatedUser**—When a user creates a new account using the CreateUserWizard control, she is immediately logged in to the site upon completion of the wizard. If the LoginCreatedUser property is set to false, the user will have to explicitly log in instead of being logged in automatically.

If the DisableCreatedUser property is set to true, you should set the LoginCreatedUser to false so that the CreateUserWizard control doesn't attempt to automatically log in a disabled user account.

Figure 28.13

The `CreateUserWizard` control is a customized Wizard control designed to enable users to easily create new accounts for your site.



- `PasswordRegularExpression`—Allows you to use a regular expression to apply specific restrictions to password requirements for your site. This regular expression is applied in addition to the password requirements specified by ASP.NET.

The `CreateUserWizard` control exposes a `MailDefinition` object so you can send the new user her login information via email if you choose. We discussed the requirements for sending email in the overview of the `ChangePassword` control earlier in this chapter.

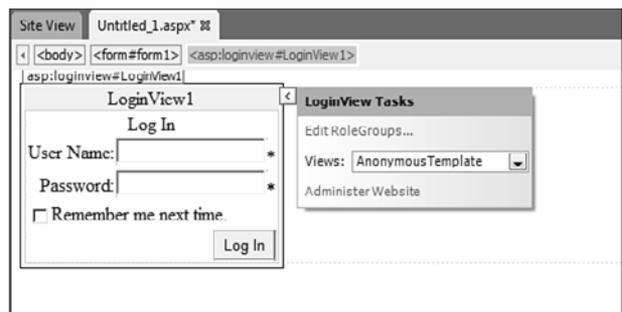
LoginView Control

So far, we've covered controls that have a single purpose. The `LoginView` control is a break from that pattern. The `LoginView` control is a container control for other controls and provides you with, by default, two views: the `AnonymousTemplate` view and the `LoggedInTemplate` view.

Controls that you place in the `AnonymousTemplate` view, as shown in Figure 28.14, are visible only to users who have not yet logged in to the site. In Figure 28.14, I have inserted a `Login` control into the `AnonymousTemplate` view so that users can log in if they haven't done so.

Figure 28.14

The `AnonymousTemplate` view is visible only to users who are browsing anonymously. After a user logs in, she no longer sees the `AnonymousTemplate` view.



On the other hand, Figure 28.15 shows the `LoggedInTemplate` view, where I have inserted `LoginStatus` and `LoginName` controls. I changed the `LoginStatus` control's view to the `LoggedIn` view so it will provide a link for the user to log out.

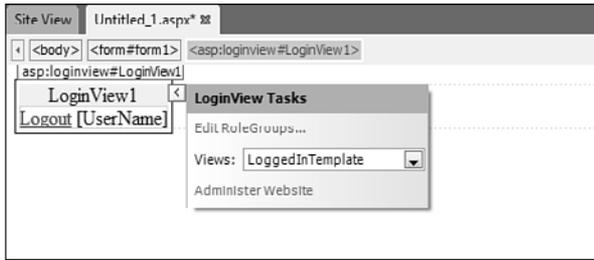


Figure 28.15

The `LoggedInTemplate` is usually visible to users who have logged in to the site. As you'll soon see, the user might see a different view instead.

The `LoginView` control is commonly used on pages where anonymous content is present, but you also want to provide a login interface in one part of the page so that users can log in if desired. A forum page is an excellent example of a page suitable for this kind of control. Users can browse the forums anonymously. If they want to post on the forum, they can log in right on the page they are currently viewing without having to redirect to a login page and lose their place.

As shown in Figures 28.14 and 28.15, the `LoginView Tasks` pop-up contains a link for editing `RoleGroups`. This enables you to define a template for different ASP.NET roles to which a user might belong using the `RoleGroup Collection Editor` shown in Figure 28.16. In Figure 28.16, you can see that I've defined a role called `Administrators`. If the logged-in user is a member of the `Administrators` role, he sees the view associated with that role instead of the `LoggedInTemplate` view. Figure 28.17 shows the new role selected in the `LoginView Tasks` pop-up.



tip

A discussion of ASP.NET roles is outside the scope of this book. If you'd like more information on ASP.NET roles, an excellent resource is available on the MSDN site at <http://msdn2.microsoft.com/en-us/library/5k85ozwb.aspx>.

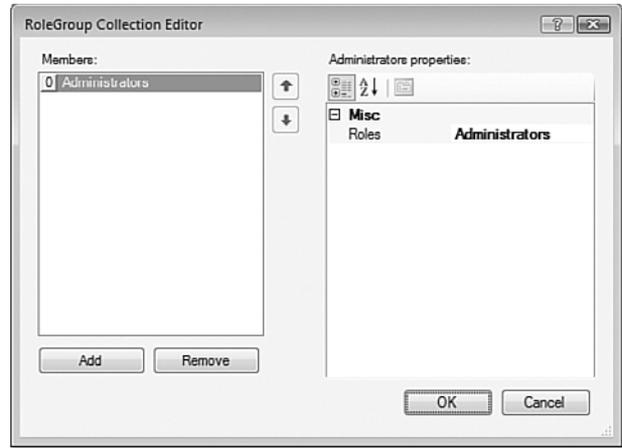


tip

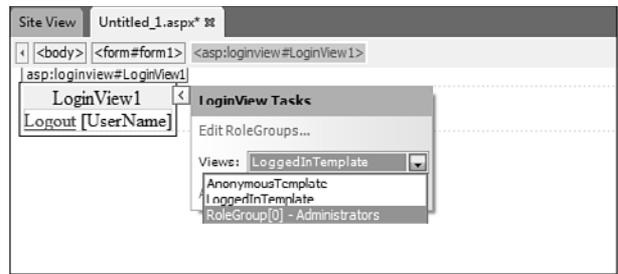
When the page containing a `LoginView` control is browsed, ASP.NET looks for a role that applies, starting at the top of the list of roles. As soon as it locates a role to which the current user belongs, it displays that view to the user. Therefore, if a user is a member of more than one role, he always sees the view for the first role in the list of `RoleGroups`.

Figure 28.16

The RoleGroup Collection Editor allows for the creation of views for specific ASP.NET roles that have been previously created.

**Figure 28.17**

When a new RoleGroup has been added, you have a new view to select in the Views drop-down. Users who are in the Administrators role see the new Administrators view.



Creating a Login Solution

Now that you've seen an overview of the various login controls that are available, let's design a simple membership site. Create a site either as a disk-based site or on your local IIS instance.

The first step in creating our membership site is to configure the site for ASP.NET Forms authentication. Forms authentication is a security mechanism enforced by ASP.NET. If a user tries to browse to a page that is not available anonymously, ASP.NET automatically sends her to a login page instead, where she can enter a username and password to log in to the site.

Use the appropriate steps that follow in order to configure the website based on whether you're using IIS or the Microsoft Expression Development Server.

note

If you are using the Microsoft Expression Development Server, refer to the section "Configuring the Website (Microsoft Expression Development Server)" later in this chapter.

Configuring the Website (IIS 5 or IIS 6)

In ASP.NET 1.0 and 1.1, configuring Forms authentication required the manual editing of configuration files. ASP.NET 2.0 and later make the job significantly easier by providing a Windows interface into the configuration options.

1. Create a new site at `http://<server>/membership`, where `<server>` is a web server running IIS 5 or greater with the .NET Framework 4 installed.
2. Open the Windows Control Panel and double-click Administrative Tools.
3. Double-click Internet Information Services.
4. Expand the computer name node in Internet Information Services.
5. Expand the Web Sites node.
6. Expand the site on which the site in step 1 was created.
7. Right-click the membership web application and select Properties.
8. Click the ASP.NET tab.
9. Make sure that 4.0.30319 is selected in the ASP.NET version drop-down.
10. Click the Edit Configuration button, as shown in Figure 28.18.
11. Click the Authentication tab.
12. Set the Authentication mode drop-down to Forms, as shown in Figure 28.19.
13. Make sure the Login URL is set to `login.aspx`. We will create this page soon.

➔ *For more information on creating sites in Expression Web, see Chapter 2, “Creating, Opening, and Importing Sites.”*

note

The following steps require IIS 5 or IIS 6. The Microsoft Expression Development Server does not allow for configuration of a site from within a Windows interface.

note

Configuration of sites is dramatically different in IIS 7, so we'll cover it in the next section.

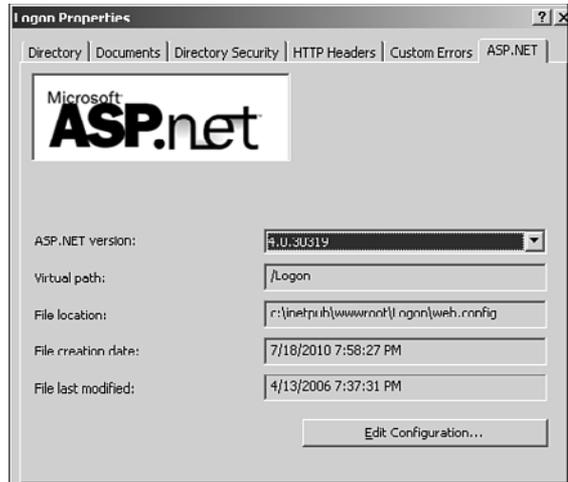
If you're running Windows XP Professional, you have access to IIS 5. If you're running Windows Server 2003, you have access to IIS 6. If you're running a later version of Windows, you have access to IIS 7 or IIS 7.5.

tip

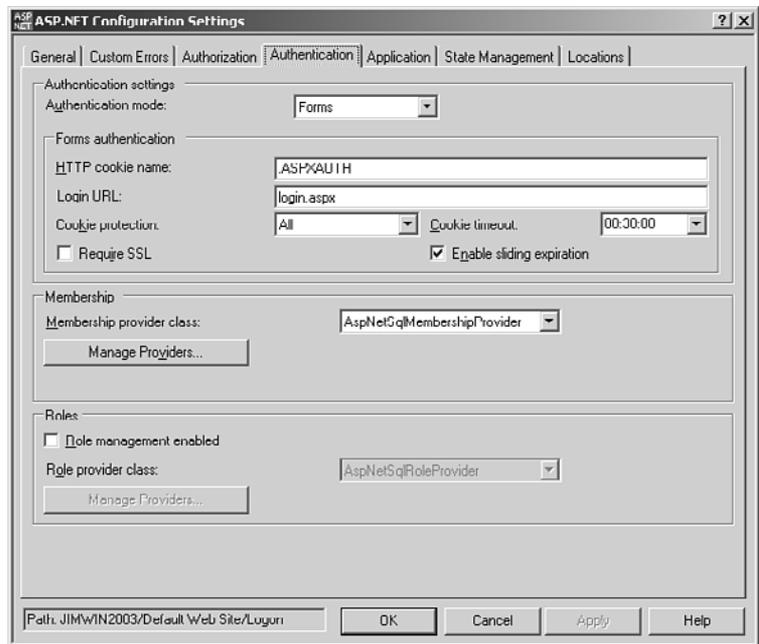
If you cannot select a version from the dropdown, make sure that the site you expanded in step 6 is configured for ASP.NET 4.0.30319.

Figure 28.18

Editing the configuration of an ASP.NET site is easy thanks to the new user interface built in to IIS.

**Figure 28.19**

We need to configure the new application for Forms authentication so we can use the ASP.NET login controls.





ASP.NET Tab Is Missing

If you've opened the properties for your site but there isn't an ASP.NET tab, the most common reason is missing Registry information. Open a command prompt and run the following command:

```
Regsvr32.exe %windir%\microsoft.net\framework\v4.0.30319\mmcaspext.dll
```

This adds the Registry information for the file that creates the tab and hopefully corrects the problem. If that doesn't correct your problem, open a command prompt and run the following commands:

```
cd %windir%\microsoft.net\framework\v4.0.30319  
aspnet_regiis -i
```

Now we need to configure the application so that no one can browse it anonymously. We'll use ASP.NET's authorization features to implement this restriction. Here's how:

1. Click the Authorization tab in the ASP.NET Configuration Settings dialog.
2. Click the Add button.
3. Select the Deny radio button in the Rule Type section.
4. Select the Anonymous Users radio button in the Users and Roles section, as shown in Figure 28.20.
5. Click OK in the Edit Rule dialog.
6. Click OK in the ASP.NET Configuration Settings dialog.
7. Click OK in the Properties dialog for your web application.

You have just configured an ASP.NET Forms authentication site and set the authorization of the site so anonymous users are not allowed.



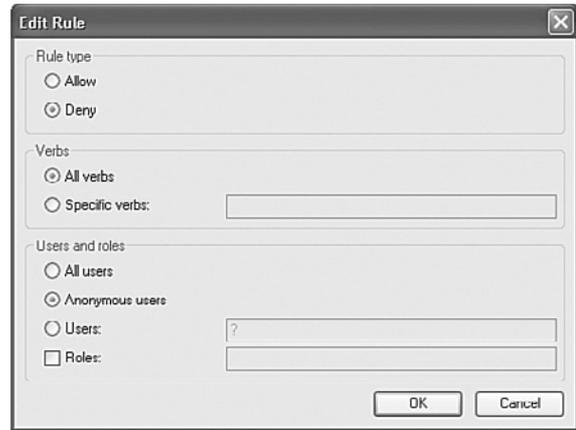
Notice that we didn't tell ASP.NET to allow anonymous users to access the login page. ASP.NET always allows anonymous access to the login page, so even though we've configured the site so no anonymous users are allowed, everyone will be able to access the login page.

Configuring the Website (IIS 7.x)

The default installation of IIS 7.x contains only the components required for the simplest site. To run an ASP.NET membership site on IIS 7.x, you must first install ASP.NET support. The steps necessary to install ASP.NET support differ based on the operating system you are running.

Figure 28.20

The Edit Rule dialog is an easy way to configure authorization settings for your ASP.NET application. You can manually edit configuration files to do the same thing, but the Windows interface is much easier to use.



Configuring ASP.NET Support for IIS 7.x

Follow these steps to install ASP.NET support on IIS 7.x running on Windows Vista or on Windows 7:

1. Select Programs and Features in Control Panel.
2. Click the Turn Windows Features On and Off link.
3. Expand the Internet Information Services node.
4. Expand the World Wide Web Services node.
5. Expand the Application Development Features node.
6. Check the ASP.NET check box, as shown in Figure 28.21.
7. Click OK.

Follow these steps to install ASP.NET support on IIS 7.x running on Windows Server 2008:

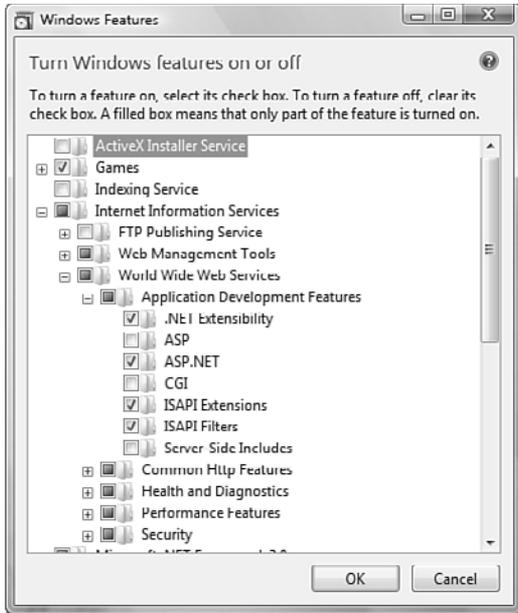
1. Open Server Manager.
2. In the Roles Summary section, click the Web Server (IIS) link, as shown in Figure 28.22.

**tip**

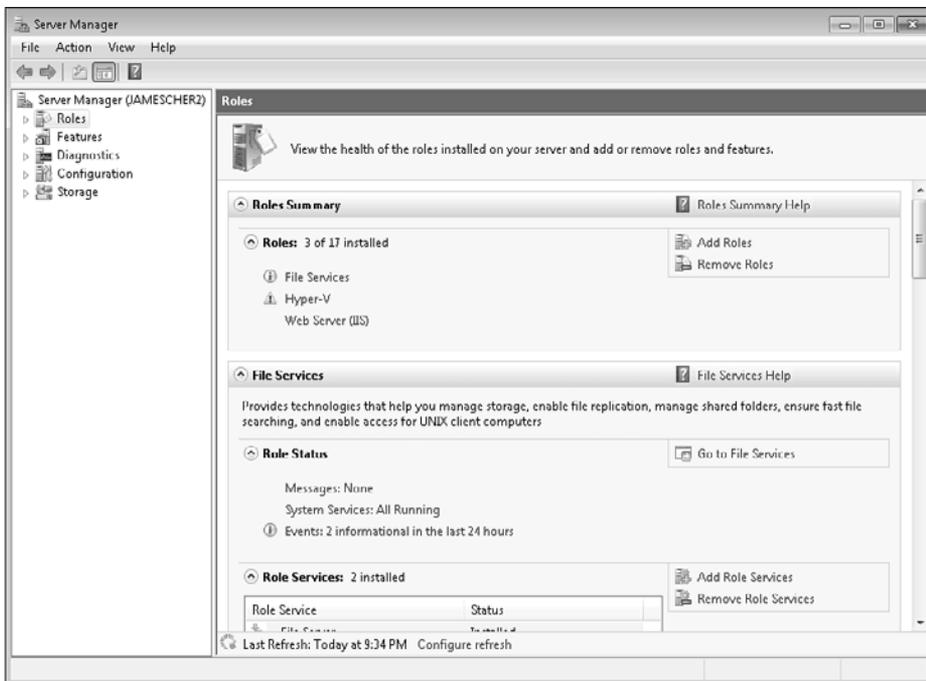
When you select ASP.NET, the .NET Extensibility, ISAPI Extensions, and ISAPI Filters nodes will be selected automatically.

**tip**

You can add .NET roles and assign your users to roles using the .NET Roles icon.

**Figure 28.21**

Check the ASP.NET check box under the Application Development Features node to install ASP.NET support on IIS 7.x.

**Figure 28.22**

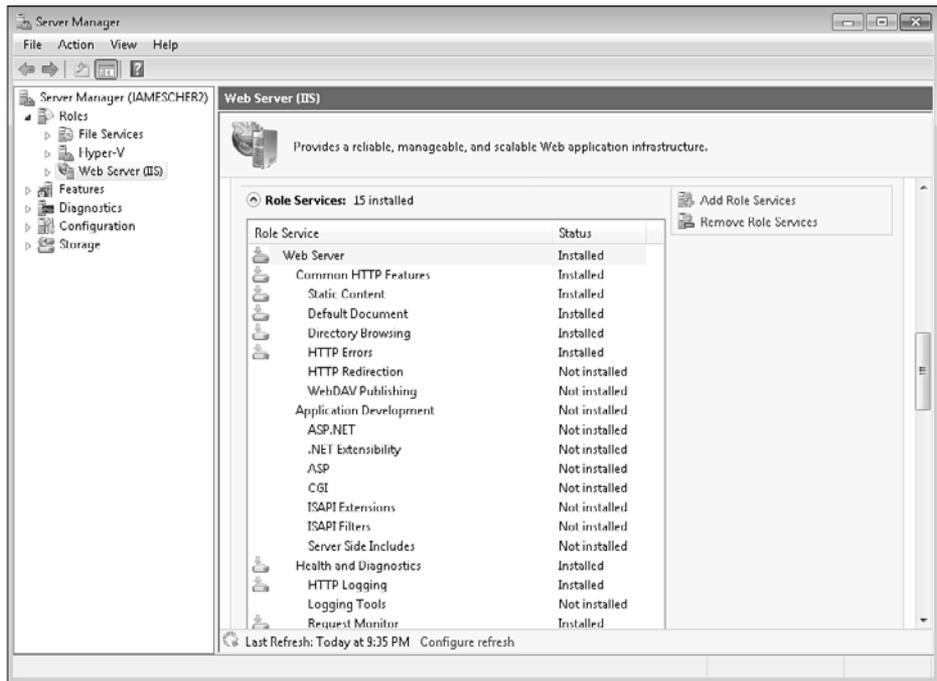
The Roles Summary section of Server Manager contains a Web Server (IIS) link after IIS 7 is installed.

3. In the Role Services section, click the Add Role Services link, as shown in Figure 28.23.
4. Check the ASP.NET check box.
5. Click the Add Required Role Services button in the Add Role Services dialog.

Note

If either Windows authentication or Basic authentication is installed, you'll need to ensure that they are disabled when you enable Forms authentication.

Figure 28.23
Click the Add Role Services link to add a new role service to IIS.



6. Click Next, and then click Install to complete the process.

After ASP.NET support has been installed, the steps for configuring ASP.NET membership are identical on Windows Server 2008, Windows Vista, and Windows 7.

Enabling Forms Authentication

The first step in configuring ASP.NET membership in IIS 7 is to enable Forms authentication for your application.

Follow these steps to enable Forms authentication:

1. Open Internet Information Services (IIS) Manager.
2. In the Connections pane, navigate to your application and click to select it.
3. Double-click Authentication in the IIS section of Features View, as shown in Figure 28.24.
4. Right-click Forms Authentication and select Enable, as shown in Figure 28.25.

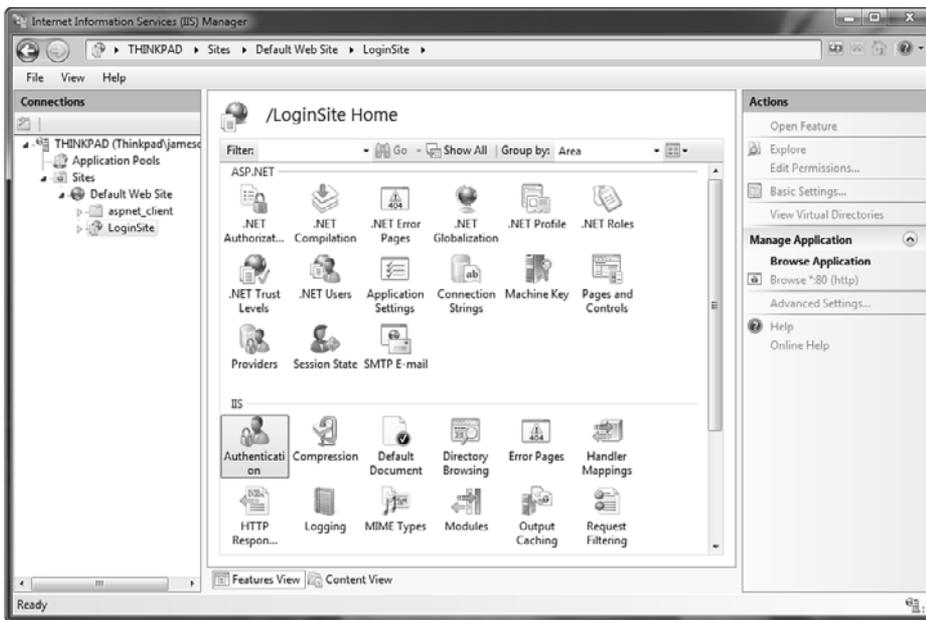


Figure 28.24
The Authentication icon in the IIS section enables you to easily configure the authentication method used for your application or site.

Adding Users

After Forms authentication is enabled, you can add users for your membership site using the .NET Users icon in the ASP.NET section, as shown in Figure 28.26.

Figure 28.25
To enable Forms authentication, right-click Forms Authentication and select Enable from the menu.

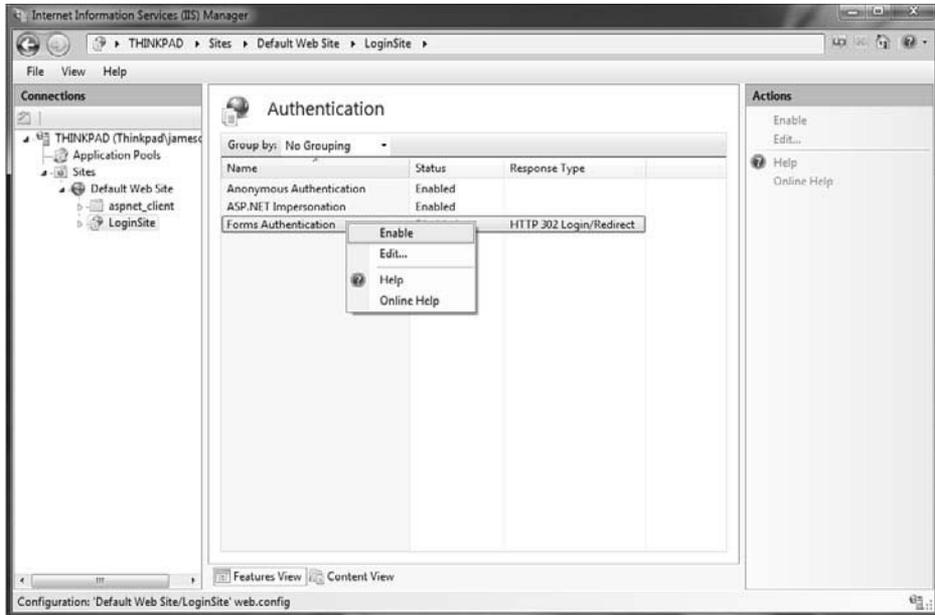
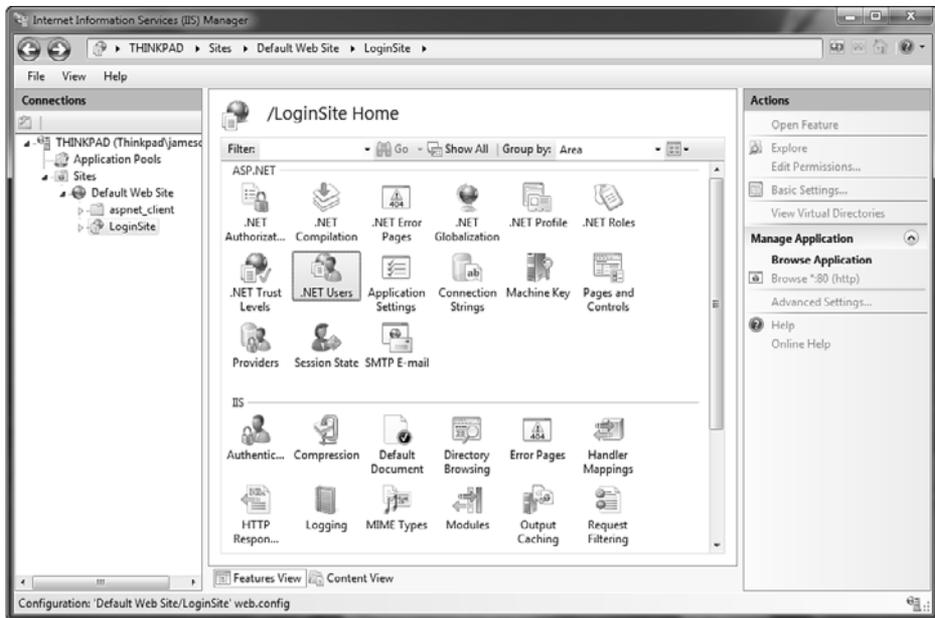


Figure 28.26
You add ASP.NET using the .NET Users icon.



After you double-click .NET Users, right-click and select Add (shown in Figure 28.27) to access the Add .NET User dialog shown in Figure 28.28.



Figure 28.27
Add ASP.NET users by selecting Add from the context menu.

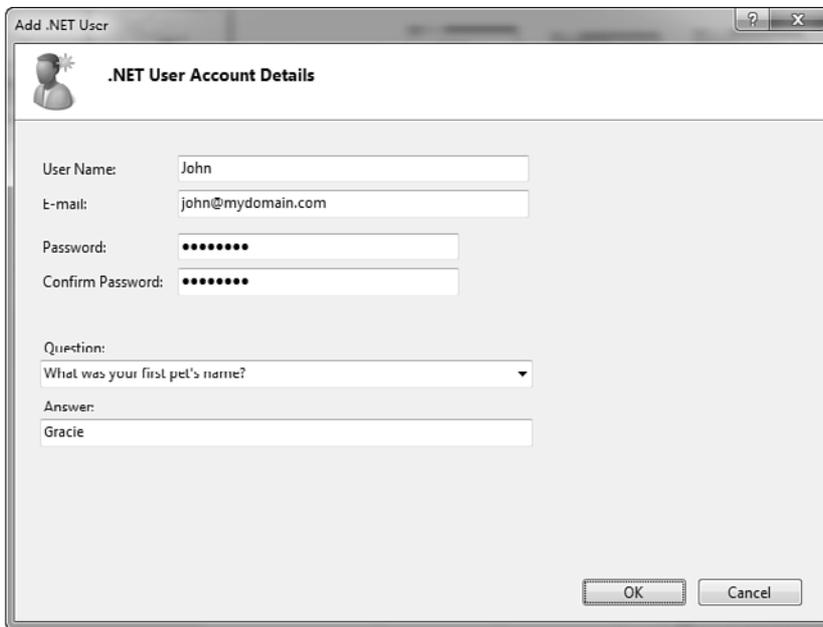


Figure 28.28
Using the Add .NET User dialog is a convenient way to add ASP.NET users to your membership site.

Fill out the Add .NET User dialog; then click OK to add your user.

Configuring Authorization Rules

After you've created users for your membership site, use the .NET Authorization Rules settings to configure access (see Figure 28.29). Doing so configures URL authorization for IIS, so authorization rules that you apply here will apply to all requests, ASP.NET pages, and other pages and files as well.

Figure 28.29
To configure authorization rules, double-click the .NET Authorization Rules icon.



After double-clicking the .NET Authorization Rules icon, you can add a new Allow rule to specify content that is allowed and a Deny rule to specify content that is denied, as shown in Figure 28.30.

note

The Authorization Rules icon is available only when URL Authentication is installed in IIS. If you don't see the Authorization Rules icon, install URL Authentication in the Security section of IIS setup.

**Figure 28.30**

IIS URL authorization rules are similar to ASP.NET authorization rules except that they apply to all requests and not just to ASP.NET pages.

Configuring the Website (Microsoft Expression Development Server)

The Microsoft Expression Development Server does not have a Windows interface for modifying the configuration of your site, so you'll need to create a configuration file for that purpose. To make matters even more confusing, Expression Web does not have IntelliSense for ASP.NET configuration files.

➔ *For information on controlling the color-coding of files within Expression Web, see Chapter 11, "Configuring Page Editor Options."*

To control the configuration of a specific web application, ASP.NET uses a special configuration file in the root of the site called `web.config`. The `web.config` file contains XML code that controls many aspects of the site.

note

For details on how IIS 7 authorization rules differ from ASP.NET authorization rules, see www.iis.net/articles/view.aspx/IIS7/Managing-IIS7/Configuring-Security/URL-Authorization/Understanding-IIS7-URL-Authorization?Page=5.

note

Because ASP.NET configuration files are XML files, Expression Web provides color-coding for the files.

To configure our site for ASP.NET membership, we need to do the following:

- Configure the site for ASP.NET Forms authentication.
- Disable the ability for unauthenticated users to browse the site.
- Turn on ASP.NET Impersonation.

Begin by selecting File, New, Page and selecting XML from the list of file types. Place your cursor after the XML code that Expression Web adds by default and enter the following code:

```
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
  <system.web>
    <authentication mode="Forms" />
    <authorization>
      <deny users="?" />
    </authorization>
    <identity impersonate="true" />
  </system.web>
</configuration>
```

Now save the file as `web.config`. Make sure to select All Files (*.*) from the Save as Type drop-down; otherwise, Expression Web automatically adds an `.xml` file extension to your file.

Note that the outermost element in the `web.config` file is the `<configuration>` section. Within that section is a `<system.web>` section. This is where you make changes to ASP.NET configuration.

In this case, the first element inside the `<system.web>` section is the `<authentication>` element. We have enabled ASP.NET Forms authentication by setting the mode of the `<authentication>` element to Forms.

The next element defines the `<authorization>` section. The `<authorization>` section enables you to control who has access to the site. In this case, we want to disallow anyone from browsing the site unless they have been authenticated. The `<deny>` tag allows us to do that. By specifying a `users` attribute with a value of `?`, we are telling ASP.NET that any user who is not authenticated should be denied access.

The next element is the `<identity>` element. We use this element to turn on ASP.NET Impersonation. The ASP.NET application would normally execute under the identity of the process in which it runs. By enabling ASP.NET Impersonation, we're forcing our page to run under the identity of the user who is authenticated to our application.



caution

ASP.NET configuration files are case-sensitive. If you enter information in the wrong case, you'll see errors in your application.



note

ASP.NET Impersonation is a feature that allows you to run ASP.NET code under a user identity other than the default identity.



tip

If you want a better method of creating a configuration file (and many other ASP.NET tasks), I encourage you to download Microsoft Visual Web Developer Express Edition from <http://www.microsoft.com/express/vwd/Default.aspx>.

Creating the Web Pages

Our membership site requires only two pages: the login page and a content page. In a real application, you'd probably want to have a separate login page and new user page, but in this sample, we use one page for both functions.

1. Create a new ASP.NET Web Form and save it as `login.aspx`.
2. Create a second ASP.NET Web Form and save it as `default.aspx`.
3. Add a Login control and a CreateUserWizard control to the `login.aspx` page. The `login.aspx` page should now look like Figure 28.31.
4. Add a new LoginStatus control to the `default.aspx` page and set the view to LoggedIn.
5. Place the insertion point just to the right of the LoginStatus control and press the spacebar on your keyboard.
6. Insert a new LoginName control.

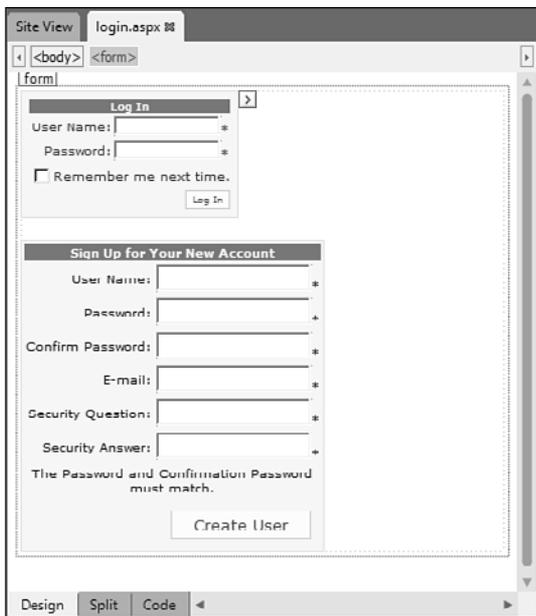


Figure 28.31

The login page serves two purposes: It allows users to log in or create a new account if they've never visited the site before.

Your `default.aspx` page should now look like Figure 28.32. When a user logs in and sees this content, the controls allow him to log out easily.

Figure 28.32

The content page uses ASP.NET login controls to allow a user to easily log out.



Now you're ready to try the site. Try browsing to the `default.aspx` page. You should be redirected automatically to the `login.aspx` page because you have not yet logged in.



Metabase Access Fails Browsing ASP.NET Page

If you're browsing your ASP.NET page and see an error that says `Failed to access IIS metabase`, it means you have a permissions issue. The account that is used to run ASP.NET needs access to a special database IIS uses to keep track of its settings.

To correct this problem, open a command prompt and switch to the following directory:

```
%windir%\microsoft.net\framework\v4.0.30319
```

At the command line, run the following command if you're using Windows 2000 or Windows XP:

```
aspnet_regiis -ga ASPNET
```

If you're using Windows 2003, run the following command:

```
aspnet_regiis -ga IISWPG
```



Failed to Start Monitoring Changes Error

If you're browsing your ASP.NET page and get an error that says `Failed to start monitoring file changes` or `Failed to start monitoring directory changes`, this error means that the ASP.NET process account does not have permission to access the site's content. Here's how to fix it:

1. Right-click the root folder of the site (`c:\inetpub\wwwroot` by default).
2. Select Properties from the menu.

continued...

3. Select the Security tab.
4. Click Add.
5. Enter **ASPNET** in the box if you are using Windows 2000 or Windows XP and **IISWPG** if you are using Windows 2003.
6. Click the Advanced button.
7. Place a check in the box Replace Permission Entries on All Child Objects with Entries Shown Here That Apply to Child Objects.
8. Click OK.
9. Click OK again.

Fill out the form to sign up for a new account, as shown in Figure 28.33; then click the Create User button to create your account. You'll be presented with a message letting you know that the account has been successfully created, as shown in Figure 28.34.



Access Denied Error

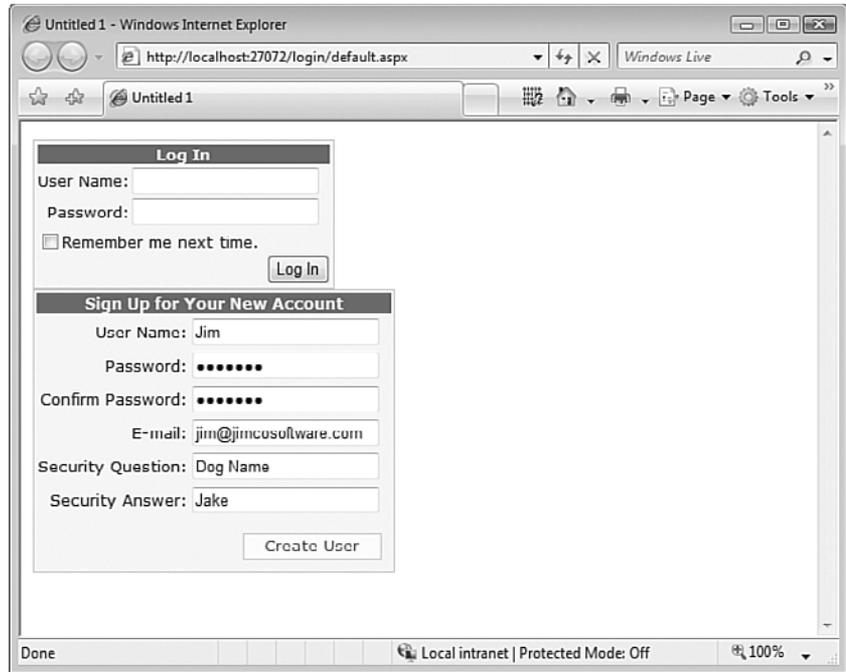
If you're browsing your membership site and get an error that says `Access denied creating App_Data` subdirectory, the steps to resolve this are the same as the steps to resolve the `Failed to Start Monitoring Changes` Error in the previous troubleshooting note.

When you create the first user in an ASP.NET membership application, the SQL Server 2005 Express Edition database is created for you automatically.

As you've seen in this chapter, creating a powerful membership system in ASP.NET and Expression Web is as simple as just dropping the controls on the page. We created an entire application without writing a single line of code.

Figure 28.33
The

CreateUserWizard control not only collects all the information for a new user, but also creates the user database when the first user is created.



The screenshot shows a Windows Internet Explorer browser window titled "Untitled 1 - Windows Internet Explorer". The address bar displays "http://localhost:27072/login/default.aspx". The page content is divided into two sections. The top section, titled "Log In", contains a "User Name:" text box, a "Password:" text box, a checkbox labeled "Remember me next time.", and a "Log In" button. The bottom section, titled "Sign Up for Your New Account", contains a "User Name:" text box with the value "Jim", a "Password:" text box with masked characters, a "Confirm Password:" text box with masked characters, an "E-mail:" text box with the value "jim@jimcosoftware.com", a "Security Question:" text box with the value "Dog Name", and a "Security Answer:" text box with the value "Jake". A "Create User" button is located at the bottom of this section. The browser's status bar at the bottom indicates "Done", "Local intranet | Protected Mode: Off", and "100%".

Figure 28.34

Your new user account has been created and you can now log in using your new user credentials.



The screenshot shows a web browser window displaying a "Log In" form and a "Complete" message. The "Log In" form includes a "User Name:" text box, a "Password:" text box, a checkbox labeled "Remember me next time.", and a "Log In" button. Below the form is a "Complete" message box with the text "Your account has been successfully created." and a "Continue" button.

Using Web Deploy to Publish a Membership Database

ASP.NET's membership features make it simple to create a login system for your site. However, many people run into trouble once they move the site to a hosting company. Most hosting companies don't support SQL Server Express Edition databases, which can be a problem given that ASP.NET uses SQL Server Express Edition by default for storing membership information.

As I mentioned earlier, many hosts provide tools for moving users created in a SQL Server Express database into a SQL Server database running on the host's server. However, if your host doesn't offer such a feature, you're still in luck. Microsoft's Web Deploy tool can take a local copy of your application and deploy it to a remote server. When it does, it can also change the connection information for your database and migrate the information from your local SQL Server Express database to the remote SQL Server database.

By far, the easiest way to use the Web Deploy tool is to use it within Visual Studio 2010. As shown in Figure 28.35, even the free Express edition offers a nice user interface for not only configuring how a site is packaged for deployment to a remote server, but also for easily publishing the files to your host.

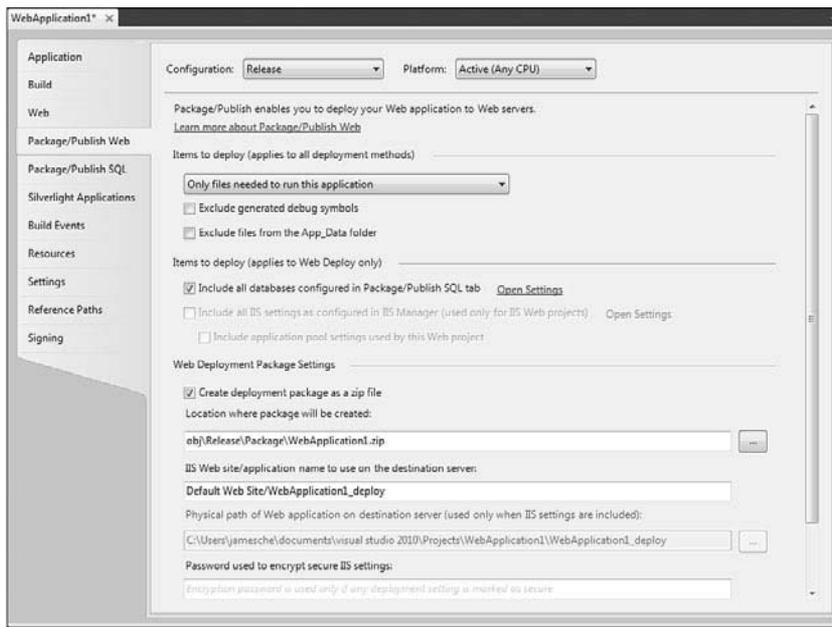
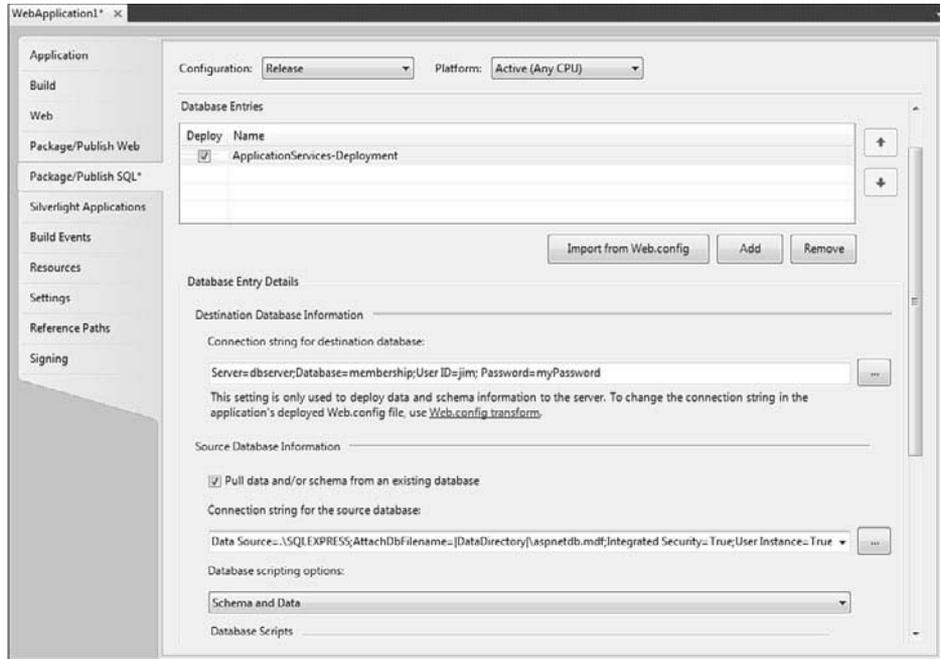


Figure 28.35
The Package/Publish Web features in Visual Studio and Visual Web Developer Express provide a powerful and flexible deployment toolset.

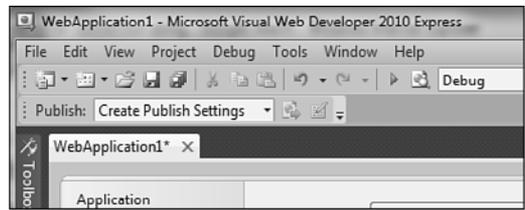
Where the deployment tools in Visual Studio really shine is in copying data from your local SQL Server Express membership database to your hosting company's SQL Server instance. Figure 28.36 shows the Package/Publish SQL settings in Visual Web Developer 2010 Express. My ApplicationServices database entry (which is stored in the `web.config` file of my application) appears in the Database Entries section. Below that is the connection string for the destination database. When Visual Studio copies my application to my hosting company, it automatically changes the connection string for my application, and because I've selected Schema and Data in the Database Scripting Options drop-down, it also copies my ASP.NET membership database as well as any users and roles that I've added to it from my local machine.

Figure 28.36
The Package/
Publish SQL
features in
Visual Studio
2010 make it
easy to deploy
your ASP.NET
membership
database.

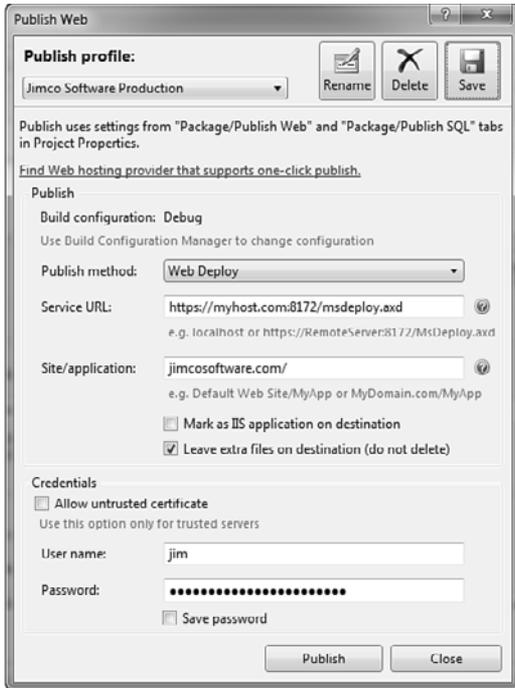


Once you've configured all of these settings, you can use the Publish toolbar in Visual Studio to deploy your application (see Figure 28.37). Selecting <New> from the drop-down on the toolbar launches the Publish Web dialog shown in Figure 28.38. From here, you can create a Publish profile that will deploy your application.

Figure 28.37
The Publish toolbar is new to Visual Studio 2010 and allows you to create Publish profiles for publishing your application.



When you use Publish profiles in Visual Studio, you can publish your application with a single click using the Publish toolbar. This feature is called One-Click Publish, and it's a great way to deploy your ASP.NET application. If you choose to use these features in Visual Studio, I strongly suggest that you find a host that supports One-Click Publish. There's a link in the Publish Web dialog (shown previously, in Figure 28.38) that takes you to a site where you can locate a host. You may also want to check out my host, DiscountASP.NET (www.discountasp.net). I've been using them for many years, and they've always been a top-notch hosting company.

**Figure 28.38**

The Publish Web dialog is where you'll configure Publish profiles that specify how your application should be published. I'm using Web Deploy in this figure.

Because this is a lengthy topic for discussion, I've only touched on these features in this section. If you want to learn more about how you can take advantage of these powerful deployment features, check out <http://msdn.microsoft.com/en-us/library/dd394698.aspx>.

FORM VALIDATION USING ASP.NET

The Need for Form Validation

You're no doubt familiar with the saying "Garbage in, garbage out." Web applications that require data entry by users provide the perfect opportunity to put that old saying into practice. Try as you might, you can never predict the actions of users of your site. That's why you must take every opportunity to prevent any user from shooting himself or herself in the foot, or worse yet, corrupting your data with bad input.

Web developers have always struggled with implementing robust form validation. Many web developers are not experts at client-side scripting, and because the most efficient form validation is client-based, that puts them at a distinct disadvantage. Perhaps you are one of those JavaScript-challenged developers yourself. Well, fear not. ASP.NET can solve the problem of form validation for you.

The ASP.NET Validation Controls

To access the ASP.NET validation controls, click the plus sign next to Validation in the ASP.NET Controls section of the Toolbox. Six validation controls are available. Before we dig too deep into how to use the validation controls provided by ASP.NET, let's first have a look at the various controls that are available.

CompareValidator

`CompareValidator` compares the value in one form field to the value in a second form field to see if they are equal. For example, when asking a user to create a password, it's common practice to present the user with one text box for the password and a second text box to confirm the password. `CompareValidator` can be used to easily ensure that both text boxes contain the same password.

RangeValidator

`RangeValidator` is used to ensure that the value entered in a form field falls within a specified range. When using this control, you specify minimum and maximum values for a particular form field. Validation fails if a value entered is outside that range.

RegularExpressionValidator

`RegularExpressionValidator` is one of the most versatile validation controls. It enables you to validate a form field based on a regular expression. Several common regular expressions are provided for you.

RequiredFieldValidator

`RequiredFieldValidator` enables you to specify that a particular form field is required. The user will then be required to enter a value or validation will fail.

CustomValidator

`CustomValidator` enables you to implement form field validation when no other control is suitable for your needs.

When you use the `CustomValidator` control, you can specify the name of a client-side function that is used to validate your form field.

ValidationSummary

The `ValidationSummary` control displays a summary of all validation failures in one location on your page. This control is most useful when you have a large form and want validation problems to appear in one place to make it easier on your users.



tip

The ASP.NET validation controls are powerful, but some developers want even more, especially in the area of cross-browser compatibility.

If you're interested in enhancing the ASP.NET validation controls, you might want to check out Peter Blum's "Professional Validation and More" suite of controls at www.peterblum.com/VAM/Home.aspx. These controls add significant functionality to the validation controls that ship with ASP.NET. Keep in mind, however, that you won't be able to add these controls to the Expression Web Toolbox. If you want to use them from the Toolbox, you'll need to use Visual Web Developer Express or Visual Studio.



tip

Regular expressions are difficult to learn. If you want to learn more about regular expressions, read *Sams Teach Yourself Regular Expressions in 10 Minutes* from Sams Publishing. The title obviously is not realistic, but this is an excellent book in downloadable format (from www.sampublishing.com).

I highly recommend this book to anyone who is interested in learning how to take advantage of regular expressions. I discovered it a couple of years ago, bought it immediately, and it has become an important reference for me.

Common Properties

Several validation controls have common properties. The `ValidationSummary` control, however, is an exception. After you learn how to use these properties for one of the controls, you'll be able to easily use the same techniques in any of the other validation controls.

ErrorMessage

The `ErrorMessage` text property is displayed when a validation error occurs. Suppose you have a `RequiredFieldValidator` attached to a text box for the user's name and the user attempts to submit a form without entering her name in the required field. In that scenario, your `ErrorMessage` property might be something like "Name is a required field. Please enter your name."

ControlToValidate

The `ControlToValidate` text property defines the ASP.NET control on the page to which the validation control is connected. Expression Web provides a drop-down list in the Tag Properties pane that lists all the valid controls on the page. To set this property, simply select the name of the control from the list.

Display

The `Display` property can be set to `None`, `Static`, or `Dynamic`. `Static` is the default value.

This property controls the CSS code that is generated by ASP.NET for the validation control. Because a validation control is displayed on a page only when validation fails, the appearance of the control can sometimes cause other page elements to shift. When that happens, the shifting of the other page elements can result in poor appearance.

When the `Display` property is set to `Static`, the validation control renders with a visibility style of `hidden`. Therefore, the control takes up its normal amount of space on the page but is not visible. This setting prevents other page elements from shifting when the validation control becomes visible.

When the `Display` property is set to `Dynamic`, the control renders with a display attribute of `none`. This causes the browser to not render the control at all until it is required. This can result in the shifting of other page elements when the validation control is displayed.

If you are not interested in displaying the validation error message next to controls (such as when you are using a `ValidationSummary` control), set the `Display` property to `None` to prevent the rendering of the controls in all cases.



caution

Form field validation is vital to the security of your site. In a real-world application, you should always implement form field validation on both the client and the server.

ASP.NET validation controls will always validate on the server as well, but if you implement your own validation instead of using the validation controls, make sure you validate user input in your server-side code.



tip

You can change the color of the text by changing the `ForeColor` property using the Tag Properties pane.



Validation Controls Don't Work

If your validation controls don't work at all—they don't show up and the page acts like they aren't there—make sure you don't have the `Visible` property set to `false` for the validation controls. Some developers are thrown off by the fact that the error message for a validation control always shows up in the designer, so they change the `Visible` property thinking that will make it work correctly. In fact, any control with a `Visible` property of `false` is not rendered by ASP.NET at all.

EnableClientScript

By default, ASP.NET validation controls generate client-side JavaScript to validate form fields. If validation fails, users are notified immediately without having to wait for a return trip to the web server. If you want to prevent ASP.NET from generating the client script used for validation, you can set the `EnableClientScript` property to `false`.

You might be wondering why you would want to disable client scripts. Suppose you are using validation controls for server-side validation only. You have some custom scripts that you or another developer wrote that are handling all the client-side validation. In that case, you would want to disable ASP.NET's generated code. You also might want to disable the client-side script if you are using numerous validation controls and you determine that the generated code makes the page too large to download quickly.



Remember that ASP.NET validation controls will always validate on the server as well. Therefore, even if you set the `EnableClientScript` property to `false`, your validation will still occur when the user submits the form.

SetFocusOnError

The `SetFocusOnError` property specifies whether focus should be given to a form field when validation fails. For example, suppose you have a Text box control for a user's phone number and you have added a `RequiredFieldValidator` to that text box with the `SetFocusOnError` property set to `true`. If the user fails to enter a value and attempts to submit the form, the text in the `ErrorMessage` property will be displayed and the cursor will be placed inside the text box so the user can correct the error.

ValidationGroup

The `ValidationGroup` property enables you to assign one or more validation controls to a group. This property makes validating multiple forms on a page without having them interfere with each other easy. You'll read more on validation groups later in this chapter.

Creating a Validated Form

Now that we've covered the basic properties and concepts behind validation controls, let's build a form that utilizes ASP.NET validation controls.

➔ *For more information on using the Microsoft Expression Development Server, see Chapter 33, "Using the Microsoft Expression Development Server."*

Create a new disk-based site and add an ASP.NET Web Form. We'll create a contact form with the following requirements:

- The user's first name and last name are required fields.
- The user will be required to enter his phone number in the format ###-###-####.
- The user will be required to provide a valid email address.
- The user must repeat his email address for verification.
- The user can optionally provide a street address and city.
- The user can provide a U.S. ZIP code, but if provided, it must be in the proper five-digit format.

➔ *For more information on creating sites, see Chapter 2, "Creating, Opening, and Importing Sites."*

➔ *For more information on creating ASP.NET Web Forms, see Chapter 25, "Using Standard ASP.NET Controls."*

note

I've asked you to create a disk-based site so you can use the Microsoft Expression Development Server to test it. If you have a local copy of IIS installed and you want to use it, please do so. Just be sure that ASP.NET is working properly on it before you begin.

Creating the Form

Before we add validation controls to the page, let's create the form:

1. Position the insertion point inside the Web Form and create a new table that is 2 rows by 2 columns. Be sure to set the border width to 0 and the width of the table to 550px.
2. Drag the vertical separator between columns to the left so the left column is approximately 150px in width.

➔ *For more information on creating tables in Expression Web, see Chapter 5, "Using Tables."*

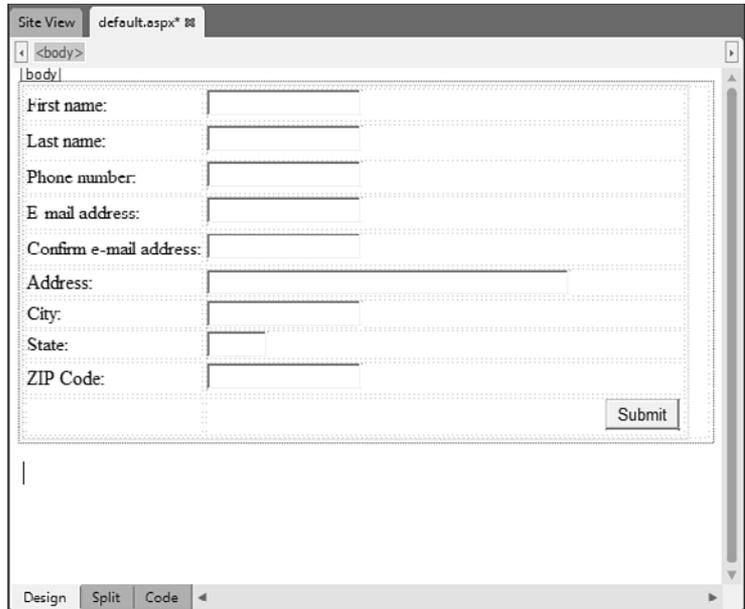
3. In the upper-left cell, enter the text **First name:**.
4. In the upper-right cell, insert an ASP.NET `TextBox` control.
5. In the left cell of the second row, enter the text **Last name:**.
6. In the right cell of the second row, insert an ASP.NET `TextBox` control.
7. Click to the right of the text box you've just added so it is no longer selected.

8. Press Tab to insert a new table row.
9. Enter the text **Phone number:** in the left cell of the new row.
10. Add an ASP.NET TextBox control to the right cell of the new row.
11. Create a new table row.
12. Enter the text **E-mail address:** in the left cell of the new row.
13. Add an ASP.NET TextBox control to the right cell of the new row.
14. Create a new table row.
15. Enter the text **Confirm e-mail address:** in the left cell of the new row.
16. Add an ASP.NET TextBox control to the right cell of the new row.
17. Create a new table row.
18. Enter the text **Address:** in the left cell of the new row.
19. Add an ASP.NET TextBox control to the right cell of the new row. Change the width of the new text box to 300px.
20. Create a new table row.
21. Enter the text **City:** in the left cell of the new row.
22. Add an ASP.NET TextBox control to the right cell of the new row.
23. Create a new table row.
24. Enter the text **State:** in the left cell of the new row.
25. Add an ASP.NET TextBox control to the right cell of the new row. Change the width of the new text box to 50px.
26. Create a new table row.
27. Enter the text **ZIP Code:** in the left cell of the new row.
28. Add an ASP.NET TextBox control to the right cell of the new row.
29. Create a new table row.
30. Add a new ASP.NET Button control to the right cell of the new row.
31. Change the align property of the right cell to right.

Your table should look like Figure 29.1.

Figure 29.1

Using a table to lay out your form will make for a more appealing form because it enables you to precisely align your form elements.



You've now created a form that can collect all the information you need. However, at this stage, your users have control over the data. They can literally enter anything they want into your form. Adding some validation controls can put you back in control of the data you are collecting.

Adding Form Validation

Before we add validation controls, we should change the name of the form fields on the form. Doing so makes configuring the validation controls much easier.

To change the names of the form fields, we use the ID property of each `TextBox` control. Configure the ID of each text box as described in Table 29.1.



tip

ASP.NET automatically prevents some dangerous data in form fields. For example, suppose someone enters some JavaScript into one of your form fields. Even without any validation controls, ASP.NET will recognize this as dangerous data and will disallow it.

Table 29.1 Form Field IDs

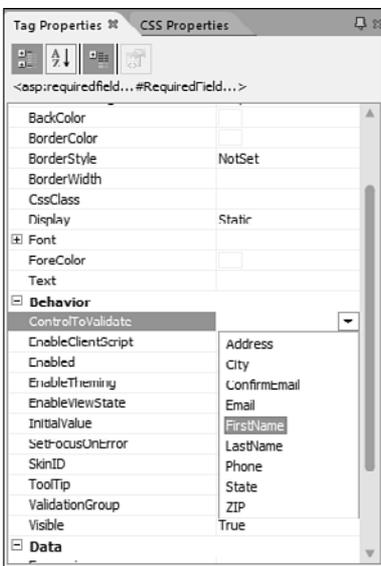
Text Box Data	Text Box ID
First Name	FirstName
Last Name	LastName
Phone Number	Phone

Table 29.1 Continued

Text Box Data	Text Box ID
Email Address	Email
Confirm Email	ConfirmEmail
Address	Address
City	City
State	State
ZIP Code	ZIP

Now we're ready to add the validation controls and finish the form:

1. Add a new `RequiredFieldValidator` immediately to the right of the `FirstName` `TextBox` control.
2. Change the `ErrorMessage` property of the `RequiredFieldValidator` to `Required Field`. This is what appears if the user fails to complete this required field.
3. Click the drop-down for the `ControlToValidate` property of the `RequiredFieldValidator` and select the `FirstName` `TextBox` control, as shown in Figure 29.2.

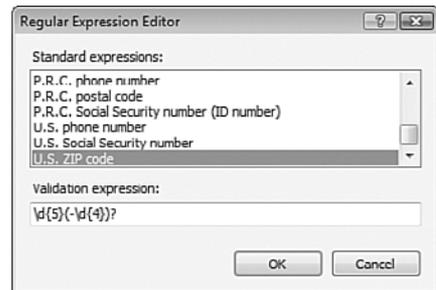
**Figure 29.2**

The `ControlToValidate` property can be set easily using the drop-down in the Tag Properties pane.

4. Add another `RequiredFieldValidator` next to the `LastName` `TextBox` control and configure it the same way that you did the first `RequiredFieldValidator`, but this time, set the `ControlToValidate` property to the `LastName` `TextBox` control. (You can copy and paste the first `RequiredFieldValidator` to make this faster.)
5. Repeat the same process to add a `RequiredFieldValidator` for the phone number and email address `TextBox` controls. Configure the `ControlToValidate` property so it validates the `Phone` and `Email` text boxes, respectively.
6. Add a new `CompareValidator` to the right of the `ConfirmEmail` text box.
7. Change the `ErrorMessage` property to `Addresses must match`.
8. Set the `ControlToCompare` property to the `Email` `TextBox` control.
9. Set the `ControlToValidate` property to the `ConfirmEmail` `TextBox` control.
10. Make sure the `Operator` property is set to `Equal`. This forces validation to fail if the values of the controls specified in the `ControlToValidate` and `ControlToCompare` properties are not the same.
11. Add a new `RegularExpressionValidator` to the right of the `ZIP` text box.
12. Change the `ErrorMessage` property to `ZIP Code is Invalid`.
13. Set the `ControlToValidate` property to the `ZIP` text box.
14. Click the ... button next to the `ValidationExpression` property in the `Tag Properties` pane to display the `Regular Expression Editor` (see Figure 29.3).
15. Select `U.S. ZIP Code` from the list of regular expressions.
16. Click `OK`.

Figure 29.3

If you don't know how to write regular expressions, you can still take advantage of them using the `Regular Expression Editor`.



We now have a functioning form, but you can enter an invalid phone number or email address. Let's add a couple more `RegularExpressionValidators` to fix that:

1. Add a new `RegularExpressionValidator` to the right of the `RequiredFieldValidator` for the Phone `TextBox` control.
2. Set the `ControlToValidate` property to the Phone text box.
3. Set the `ErrorMessage` property to Invalid Phone Number.
4. Set the `ValidationExpression` to U.S. Phone Number.
5. Add a new `RegularExpressionValidator` to the right of the `RequiredFieldValidator` for the Email `TextBox` control.
6. Set the `ControlToValidate` property to the Email text box.
7. Set the `ErrorMessage` property to Invalid Email.
8. Set the `ValidationExpression` to Internet email address.

You can now save and browse the form. Notice that when you do, the spacing on the Phone `TextBox` control is thrown off a bit by the validation controls, as shown in Figure 29.4. You can easily resolve that using the `Display` property.

Extra space caused by validation control

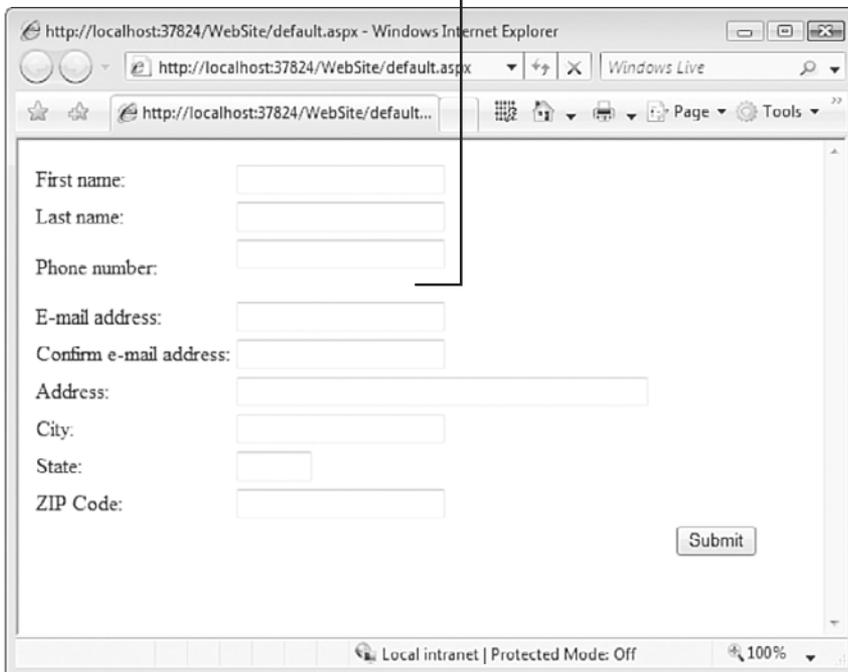


Figure 29.4
Because the Phone text box has two validation controls associated with it, the space below it is thrown off by the space reserved for the validation controls.

Change the `Display` property for both validation controls associated with the Phone text box to `Dynamic`. Now save and browse the form again and you'll see that the spacing between all the `TextBox` controls is consistent.

Testing the Form

To test the form, enter your name and then enter 1 - 555 - 222 - 1212 for the phone number. Click another field (not the Submit button) or press `Tab` to go to the next field. You will see a validation failure immediately on the phone number field because the number is in an incorrect format. Change the phone number value to 555 - 222 - 1212 and it will pass validation. Test the other validation controls as well to ensure that we've met the requirements outlined earlier.



Submit Button Does Nothing

If you have completed all the fields on your form correctly but nothing happens when you click Submit, be sure you aren't using any client-side script that could be interfering with the submission of the form other than script generated by ASP.NET. If you mix your own client-side validation code with ASP.NET's code, it can sometimes cause this behavior. Test the page without your custom script and see whether that corrects the problem.



tip

As mentioned previously, ASP.NET also validates your form on the server. If you want to test this, disable JavaScript in your browser and test your form. You'll see that your validation controls still display the error messages when they fail, but it requires a round trip to the server to validate.

Validation Groups

You've seen the power and versatility of ASP.NET validation controls, but what happens when you have two or more different areas of a page that act as separate forms? For example, suppose you have three different mailing lists that users of your site can sign up for. You have an email `TextBox` control and a button for each mailing list. You also hook up a `RegularExpressionValidator` and `RequiredFieldValidator` control to each text box so you get good email addresses.

Imagine the frustration of your users when they realize that to sign up for any mailing list, they must enter a valid email address into the `TextBox` controls of all three mailing lists! Validation groups are a new feature to ASP.NET, and they were created to solve just this kind of problem.

Each of your `TextBox` controls and each of your validation controls will have a `ValidationGroup` property. (The `ValidationGroup` property is not specific to text boxes, but our example is.) By



note

As you can see, ASP.NET validation controls are a powerful way to help improve the security of your site. However, they are just one step in protecting your data. If you want a more comprehensive approach to securing your application, the Microsoft Developer Network (MSDN) site offers many great resources, including a comprehensive white paper at <http://msdn.microsoft.com/msdnmag/issues/05/11/SecureWebApps/default.aspx>.

entering the same value for the `ValidationGroup` property of a `TextBox` control and a validation control, you create a relationship between those controls. Validation controls will pay attention only to controls that are within the same validation group.

You can easily solve the problem outlined previously by creating three validation groups, one for each mailing list.

USING ASP.NET WEB PARTS

An Introduction to Web Parts

The term *Web Parts* can actually mean two different things: Web Parts as a technology and Web Parts as the controls that reside on an ASP.NET web page.

Web Parts technology consists of a series of ASP.NET controls that let you easily develop portal-like pages that can be customized and arranged within a browser while viewing a page. Web Parts technology also supports ASP.NET *personalization*, a feature that allows individual users of a site to customize the appearance and layout of a page and to have that customization applied automatically each time they visit the site.

Web Parts controls are the controls inserted onto a Web Parts page. They can be either custom controls that you or other developers create or the Web Parts controls that exist within the Expression Web Toolbox.

Web Parts controls are added to a page inside an area called a *Web Parts zone*. When you design your page, you first add one or more `WebPartZone` controls to the page; then you add other controls to the Web Parts zone. Depending on the configuration of the page, users can drag and drop Web Parts between Web Parts zones.

As mentioned previously, you can create your own custom Web Parts for use in your pages. The most common method of creating a custom Web Parts control is to create a special kind of ASP.NET page called a *user control*.

I first covered user controls in Chapter 27, “Using ASP.NET Master Pages and User Controls.” In this chapter, we create a user control that we can use in a Web Parts page later in this chapter.

Creating ASP.NET User Controls

As mentioned previously, an ASP.NET user control is a special kind of ASP.NET page. You can think of a user control as a sort of “page within a page.”

When you are designing a user control, you work with it just as you work with a regular ASP.NET page. You can add controls to it, use Expression Web behaviors, create layers, and so on. However, the user control cannot be browsed directly. Instead, you must place it onto an ASP.NET page and then browse to the ASP.NET page to use the user control.

As you might have surmised, ASP.NET user controls are useful for creating reusable components that can be inserted onto an ASP.NET page. You’ll have the chance to create a couple of user controls as you progress through this chapter. As a matter of fact, let’s create a simple user control now.

To create an ASP.NET user control, follow these steps:

1. Select File, New, Page.
2. Select ASP.NET from the list of page types on the left.
3. Select Web User Control from the list of ASP.NET page types.
4. Select the language of your choice, and click OK to create the user control.

After you’ve created your user control, drag a few controls onto it:

1. Drag an ASP.NET Label control onto the user control.
 - ➡ *For more information on adding ASP.NET controls to a page, see Chapter 25, “Using Standard ASP.NET Controls.”*
 - ➡ *For more information on using ASP.NET controls in the Toolbox, see Chapter 25.*
2. Position the insertion point just to the right of the Label control and press Enter.
3. Drag an ASP.NET TextBox control onto the user control.
4. Position the insertion point just to the right of the TextBox control, and press the spacebar.
5. Drag an ASP.NET Button control onto the user control.

note

ASP.NET personalization features rely on a database to store data on individual users. Before you go through the examples in this chapter, you should download and install Visual Web Developer Express from Microsoft by browsing to <http://www.microsoft.com/express/Web/>. Visual Web Developer Express comes with a version of SQL Server that you can use for this purpose.

note

ASP.NET user controls can also contain ASP.NET code. When a user control is placed on an ASP.NET page, the code it contains is also available.

note

Now is a good time to create a new site you can use to work through the samples in this chapter. If you don’t want to create your own site, you can find all the finished files for this chapter’s site in the `Examples\Chapter30\Files\Website` folder on the website that accompanies this book at www.informit.com/register.

The sample site was developed in C#, but code samples in this chapter are provided in both C# and VB.

6. Switch to Code View and change the first line of code in the control from this:

```
<%@ Control Language="C#" ClassName="WebUserControl1" %>
```

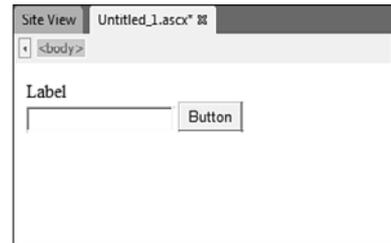
to this:

```
<%@ Control Language="C#" ClassName="PropControl" %>
```

Your user control should now look like Figure 30.1. Save the user control as `prop.ascx`. Later in the chapter, we use this user control to create a Web Parts control.

Figure 30.1

Your finished user control contains three simple ASP.NET controls.



Web Parts Controls in the Toolbox

ASP.NET comes with several Web Parts controls that are used to create Web Parts pages. These controls are located in the ASP.NET section of the Expression Web Toolbox.

We don't have room to go into the details of every Web Parts control in the Toolbox, but we will discuss the following controls as we progress through this chapter:

- **WebPartManager**—The `WebPartManager` is a nonvisual control that manages all the Web Parts controls and zones on a page. Any page that uses Web Parts controls must have exactly one `WebPartManager` on the page.
- **WebPartZone**—A `WebPartZone` is a container for Web Parts controls. Each `WebPartZone` control is identified with a unique ID. Depending on the current configuration of the `WebPartManager` control, users might be able to drag and drop Web Parts between zones in the browser. Users might also be able to add new Web Parts controls to the zone of their choice.
- **CatalogZone**—The `CatalogZone` is a special Web Parts zone that allows users to browse available Web Parts controls and add them to the Web Parts zone of their choice.

note

In this chapter, we won't go into detail on every feature concerning Web Parts, but we will cover enough of the features so that you'll feel comfortable experimenting with them and using them in your own site.

note

Although all the Web Parts controls used in this chapter are made up of user controls and existing ASP.NET controls, a developer can also create custom Web Parts controls from scratch. Doing so is not trivial and is far outside the scope of this book.

If you're interested in more advanced material on the topic of custom Web Parts controls, Microsoft has some great documentation available at <http://quickstarts.asp.net/QuickStartv20/aspnet/doc/webparts/custom.aspx>.

- **EditorZone**—The `EditorZone` control adds editing functionality to a Web Parts page. It is used in conjunction with an editing control such as the `AppearanceEditorPart` control.
- **AppearanceEditorPart**—The `AppearanceEditorPart` provides a user interface with which a user can modify the appearance of Web Parts controls on a page.

Creating a Web Parts Page

You now have enough information regarding Web Parts controls to create a Web Parts page. We'll start simple and then build on the page as we progress through the rest of the chapter. Do the following:

1. Create a new ASP.NET page.
2. Drag a `WebPartManager` control onto the page.
3. Place the insertion point to the right of the `WebPartManager` control, and press `Enter` to insert a new line.
4. Insert a new table. Set the number of rows to 1, the number of columns to 2, and the table width to 500 pixels. Leave all other settings at the default values.
5. Insert one `WebPartZone` control in each table cell.
6. Select the first `WebPartZone` control and click the arrow button to display the `WebPartZone` Tasks pop-up as shown in Figure 30.2.
7. Click the `AutoFormat` link, select the Professional scheme, and click `OK`.
8. Select the second `WebPartZone` control and click the arrow button to display the `WebPartZone` Tasks pop-up.
9. Click the `AutoFormat` link, select the Colorful scheme, and click `OK`.
10. Drag a `Calendar` control from the Standard ASP.NET Controls section of the Toolbox and drop it into the first `WebPartZone` control.
11. Drag a `Login` control from the Login ASP.NET Controls section of the Toolbox and drop it into the second `WebPartZone` control.
12. Save the page as `default.aspx`.

Your page should look like Figure 30.3. You can't tell from the black-and-white image in Figure 30.3, but the controls you dropped in each `WebPartZone` control take on the formatting of that zone.



tip

Because the `WebPartManager` is a nonvisual control, you might be prompted to enable the Visual Aid so you can see the `WebPartManager` control. You should enable the Visual Aid because it will give you a better understanding of the makeup of a Web Parts page. However, not enabling the Visual Aid will not prevent you from completing the example in this chapter.



tip

To place the insertion point to the right of the `WebPartManager` control, first select the `WebPartManager` control and then press the right arrow key on your keyboard.



caution

Be sure that you select `In Pixels` instead of `In Percent` when inserting the table; otherwise, the table will take up five times the width of the table.

Figure 30.2

WebPartZone controls can be easily configured using the WebPartZone Tasks pop-up.

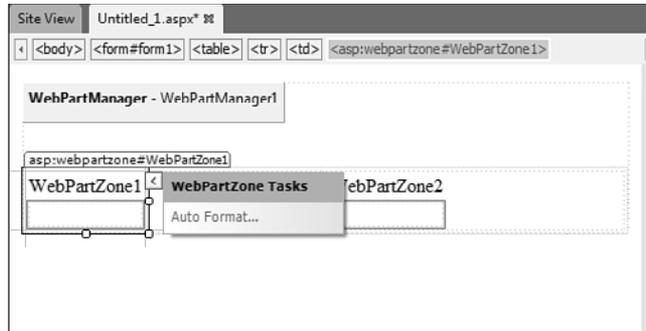
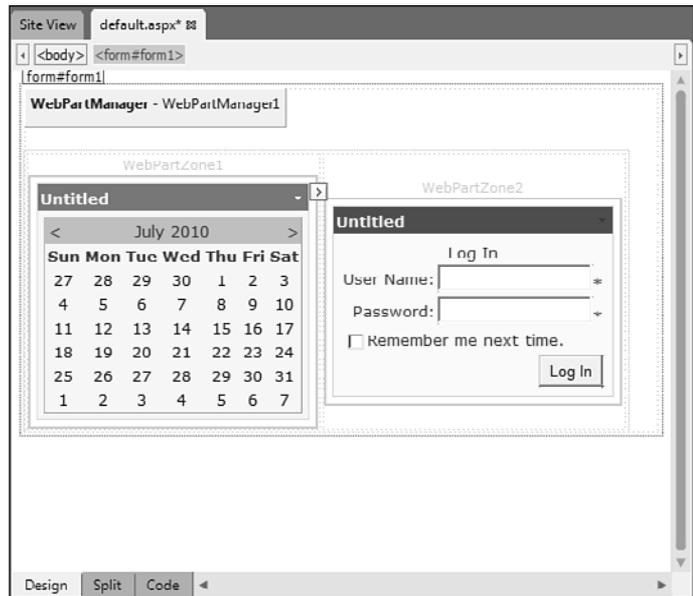


Figure 30.3

This Web Parts page contains two Web Part zones. It has limited functionality at this point.



Go ahead and browse the page. Each Web Parts control on the page has a small downward-pointing arrow button in the upper-right corner. If you click that button, a menu is displayed that enables you to either minimize or close the Web Parts control, as shown in Figure 30.4. As we add more features to this Web Parts page, you'll see new items appear on this menu.



WebPartManager Error When Browsing Web Parts Page

If you add a WebPartManager Web Parts control and some Web Parts zones to your page but, when you browse the page, see an error telling you that you need to add a WebPartManager control to enable Web Parts, note that the WebPartManager must be added before any other Web Parts controls. If you add the WebPartManager after other Web Parts controls, you can drag and drop the WebPartManager to the top of the page to get rid of this error.

Note that you might need to enable the ASP.NET nonvisual controls Visual Aid to accomplish this.



Error Connecting to SQL Server

ASP.NET Web Parts pages use SQL Server Express by default to store data about how a page is laid out for individual users. If you haven't installed SQL Server Express, you'll see an error indicating that ASP.NET could not connect to SQL Server.

You can get SQL Server Express free from Microsoft by browsing to <http://www.microsoft.com/express/Database/> or by installing it along with Visual Web Developer Express as previously mentioned in this chapter.

If you get an error connecting to SQL Server at this point, see the Troubleshooting note, “Error Connecting to SQL Server” earlier in this chapter.

If you click Minimize on the menu, the Web Parts control collapses so that only the title is displayed and the Minimize menu item changes to Restore. If you click Close on the menu, the Web Parts control is removed from the page.

At this stage, you can't customize the page or move Web Parts controls around. To do that, we'll need to change the display mode of the page.



note

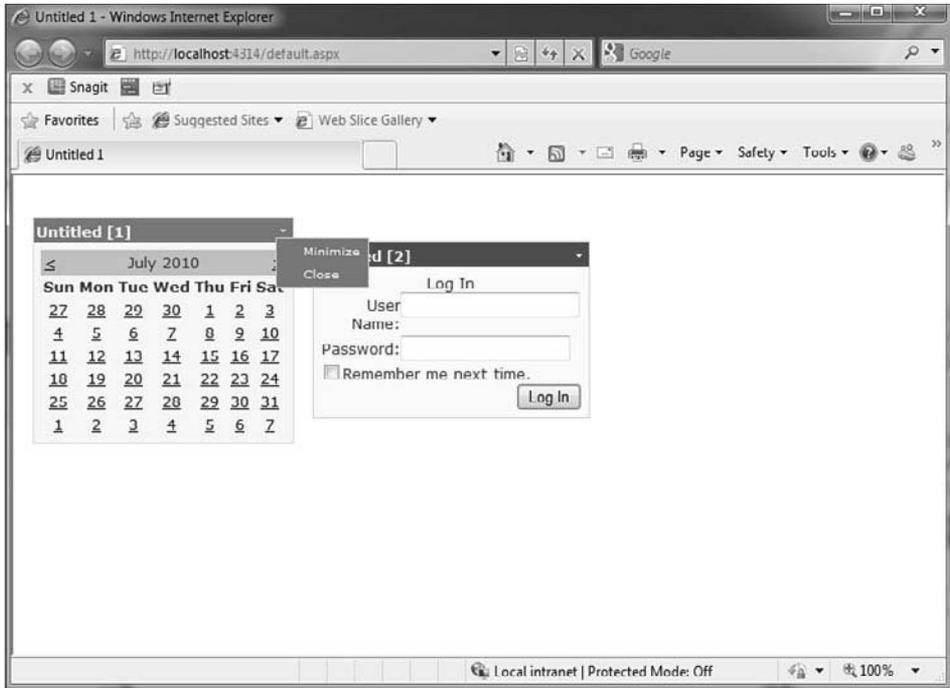
When you preview your Web Parts page, ASP.NET automatically creates a SQL Server Express database and adds it to the App_Data folder in your site. ASP.NET uses this database to store personalization information for the Web Parts page.



caution

Be sure you don't select Close from the menu of either Web Part at this point. If you close a Web Part, it will remove it from the page and there is no easy way to get it back until we add that functionality later.

Figure 30.4
Each Web Parts control contains a menu that enables you to interact with it. You can't do much at this stage.



Web Parts Page Display Modes

A Web Parts page can be displayed in one of five display modes:

- **BrowseDisplayMode**—This is the default view of the page. It allows minimal configuration of the page. The page you previewed earlier was in **BrowseDisplayMode**.
- **CatalogDisplayMode**—In this mode, the Web Parts Catalog is displayed so users can add or remove Web Parts controls from the page. Web Parts controls can also be dragged and dropped in this mode.
- **ConnectDisplayMode**—In this mode, the Web Parts page displays user interface elements that aid in connecting Web Parts together. (For example, you might have a ZIP Code lookup Web Part that communicates with a weather map Web Part.)
- **DesignDisplayMode**—In this mode, the Web Parts zones are visible and users can drag and drop Web Parts controls between zones and within zones.
- **EditDisplayMode**—In this mode, Web Parts controls can be edited. To allow editing of a Web Parts control, the page must contain an **EditorZone** control and at least one editor control such as the **AppearanceEditorPart** mentioned earlier.

To change the display mode of a Web Parts page, set the `DisplayMode` property of the `WebPartsManager` control. Unlike many of the other properties we've set on ASP.NET controls so far in this book, you cannot set the `DisplayMode` property declaratively in the HTML code for the control. Instead, you must do it using server-side ASP.NET code.

Creating a User Control That Sets the Display Mode

It's common practice to provide users with a user interface for changing the display mode of a Web Parts page. Because an interface that changes display modes is something you might want to use in all your Web Parts pages, creating the interface as a user control is the perfect choice.

Let's create a simple user control that can be used on any Web Parts page you create to change the display mode. Here's how:

1. Create a new user control.
2. Add a `DropDownList` ASP.NET control to the user control and change the `ID` property to `ddlDisplayMode`.
3. In the Common `DropDownList` Tasks pop-up, check the `Enable AutoPostBack` check box.
4. In the Common `DropDownList` Tasks pop-up, click the `Edit Items` link.
5. Click `Add` in the `List Item Collection Editor` dialog.
6. Change the `Text` property to `Browse`, as shown in Figure 30.5.



caution

Be sure you don't insert a `Drop-Down Box` control from the `Form Controls` section of the `Toolbox`. You should insert a `DropDownList` control from the `Standard ASP.NET` section of the `Toolbox`.

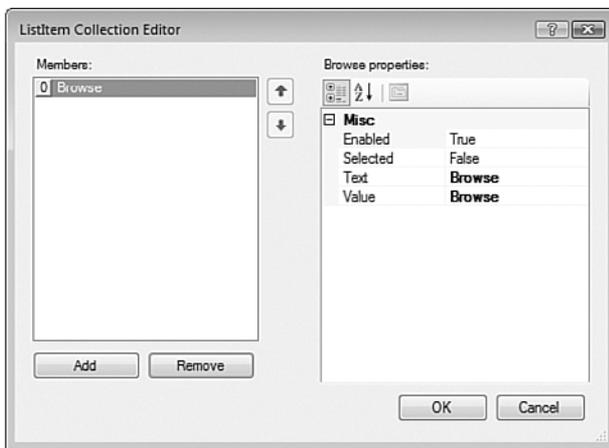


Figure 30.5

Items are added to the `DropDownList` control using the `List Item Collection Editor`.

7. Click Add to add a new ListItem.
8. Change the Text property to Edit.
9. Add two more ListItems. Change the text to Design for the first and Catalog for the second.
10. Click OK in the ListItem Collection Editor dialog.
11. Save the user control as `displaymodeui.ascx`.

At this point, the user control has no functionality. Let's add some code to change the display mode of the Web Parts page to which the user control will be added.

Adding Code to Change the Display Mode

Switch to Code View in the user control and add the following code after the first line of code that starts with `<%@ Control`. If you are using C#, use the code in Listing 30.1; if you are using VB, use the code in Listing 30.2.

Listing 30.1 C# Code for User Control

```
<script runat="server">
    // create a local WebPartManager
    WebPartManager _mgr;

    void ddlDisplayMode_SelectedIndexChanged(object sender, EventArgs e)
    {
        WebPartDisplayMode mode =
        _mgr.SupportedDisplayModes[ddlDisplayMode.Selected.Value];
        _mgr.DisplayMode = mode;
    }

    void Page_Load(object sender, EventArgs e)
    {
        _mgr = WebPartManager.GetCurrentWebPartManager(this.Page);
    }
</script>
```

Listing 30.2 VB Code for User Control

```
<script runat="server">
    'create a local WebPartManager
    Dim _mgr As WebPartManager

    Sub ddlDisplayMode_SelectedIndexChanged(ByVal sender As
    object, ByVal e As
    EventArgs)
```

note

You might have noticed that the server-side code includes code that runs when the page runs, but you are not required to hook up that code so it runs when the user control is loaded. That's because ASP.NET handles that automatically for you.

```
Dim mode As WebPartDisplayMode =
_mgr.SupportedDisplayModes(ddlDisplayMode.SelectedValue)
_mgr.DisplayMode = mode
End Sub

Sub Page_Load(ByVal sender As object, ByVal e As EventArgs)
_mgr = WebPartManager.GetCurrentWebPartManager(Me.Page)
End Sub

</script>
```

This code is pretty simple. It uses the `SupportedDisplayModes` method of the `WebPartDisplayMode` to determine whether the selected mode is supported. It then changes the `DisplayMode` property of the `WebPartManager` to the mode you selected.

The final code change in the user control is to hook up the server-side code you just added to the `DropDownList` control. Locate the code in the user control for the `DropDownList` control and change it from this:

```
<asp:DropDownList runat="server" id="ddlDisplayMode" AutoPostBack="True">
```

to this:

```
runat="server" id="ddlDisplayMode"
AutoPostBack="True" OnSelectedIndexChanged="ddlDisplayMode_SelectedIndexChanged">
```

Save the user control. The user control is now complete and is ready to be added to the Web Parts page:

1. Open the Web Parts page (`default.aspx`) and switch to Design View if it's not already visible.
2. From the Folder List in Expression Web, drag the `displaymodeui.ascx` user control and drop it right under the `WebPartManager` control in `default.aspx`.
3. Save `default.aspx`.

Browse `default.aspx` and you'll notice that the drop-down from the user control appears above the Web Parts zones, as shown in Figure 30.6.

If you select Design from the drop-down, you'll notice that the Web Parts zones become visible and you can drag and drop controls from one Web Parts zone to the other (see Figure 30.7).



tip

By dragging and dropping the user control onto the page, you have added the functionality provided by the user control to the page. In this case, you've just added a user interface for switching the display mode of the Web Parts page. To add the same user interface to any other Web Parts page, simply drop the same user control onto the Web Parts page.

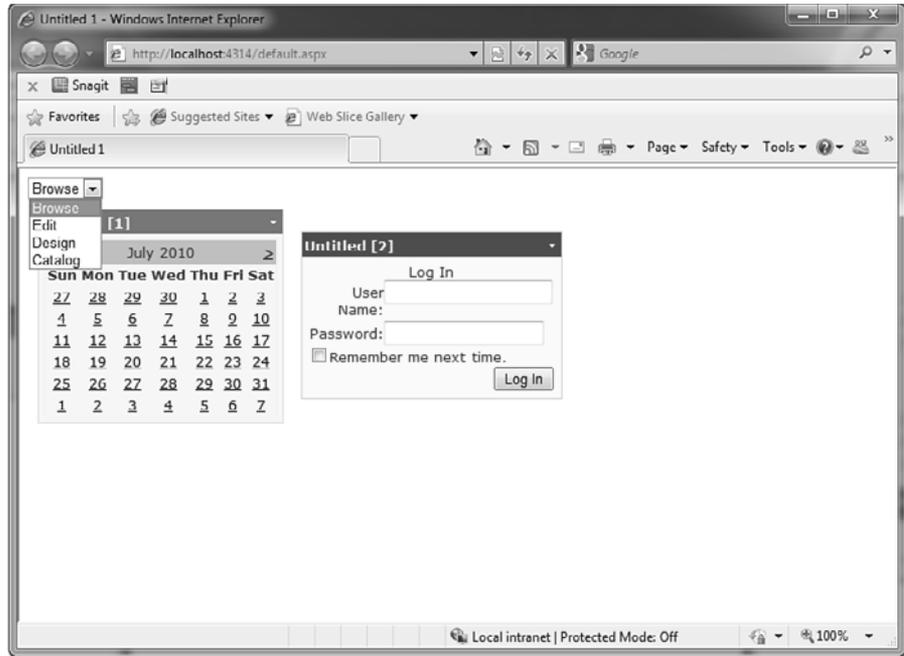


note

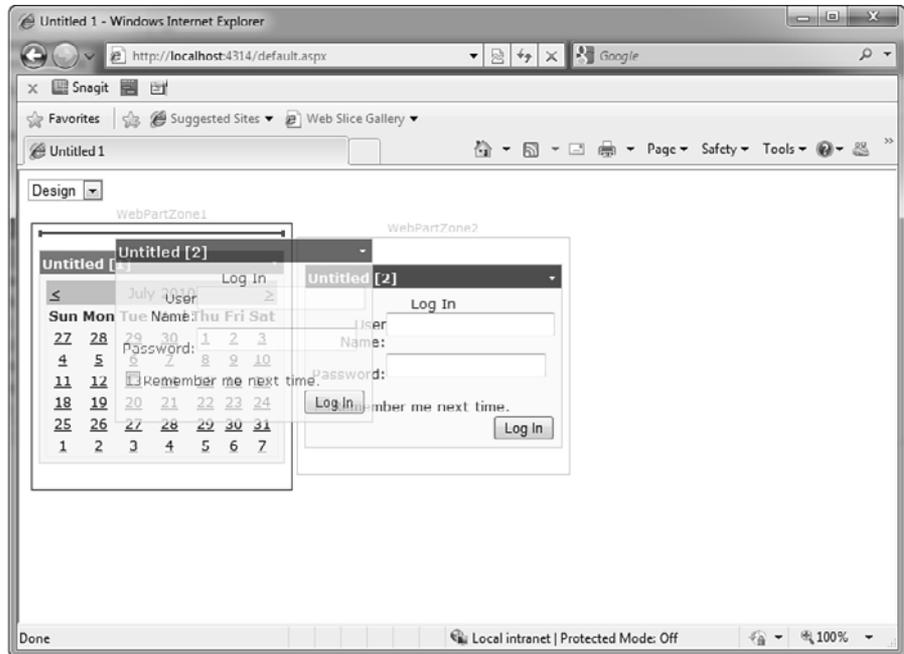
At this point, if you select anything other than Browse or Design, you will see an error in the browser. We add more functionality later in this chapter.

Figure 30.6

The user control we just added creates a user interface for changing the display mode of the Web Parts page.

**Figure 30.7**

In Design mode, you can drag and drop controls from one Web Parts zone to another. In this case, I'm dragging the Log In control from the zone on the right to the zone on the left.





Null Value Message When Browsing Web Parts Page

Suppose you are browsing your Web Parts page after adding the `displaymodeui.ascx` user control. When it loads in the browser, all you see is an error that says `Value cannot be null`, and you have no idea what that means.

The code that we added to the `displaymodeui.ascx` user control uses a value in a `DropDownList` control to set the display mode. ASP.NET knows which display modes are valid, and if you attempt to set one that isn't valid, you'll see this message.

There could be a couple of reasons for this. If you are selecting a mode that should be valid, it's likely that you simply misspelled one of the entries in the `DropDownList` control. Check the items in the `DropDownList` carefully and ensure that they are spelled correctly.

Another cause of this problem is selecting a mode that is not currently valid. For example, if you were to select `Edit` at this point, it would display the error you are seeing because `Edit` is not a valid mode until the page has been configured to allow for the editing of Web Parts controls.

In a real application, the code in the user control would intercept this error and perhaps just display a neatly formatted informational message to the user. However, that kind of complexity is outside the scope of this book.



note

You might not know it, but ASP.NET personalization features are kicking in at this point. If you move a control from one Web Parts zone to another and then close your browser, that move will still be in effect the next time you browse the page.

Because you haven't actually logged in to this site, ASP.NET personalization features are using client cookies to identify you. If you move to another computer and browse the same page, changes you made on the first computer will no longer apply.

The Web Parts Catalog

The Web Parts page you've created to this point contains only two Web Parts controls. To fully utilize the power of Web Parts, let's add the capability to add new Web Parts controls to the page.

The capability to add Web Parts controls to a page is provided by the `CatalogZone` and `DeclarativeCatalogPart` Web Parts controls.

1. Open `default.aspx` in Expression Web.
2. Create a new table column to the right of the existing table columns.



For more information on adding columns to tables, see Chapter 5, "Using Tables."

3. Drag a `CatalogZone` control from the Toolbox to the new table cell.
4. Click the `AutoFormat` link in the `Common Catalog Zone Tasks` pop-up and apply a formatting scheme of your choice.
5. Drag a `DeclarativeCatalogPart` control from the Toolbox, and drop it inside the `CatalogZone` control you added in step 3.
6. Click the `Edit Templates` link in the `Common DeclarativeCatalogPart Tasks` pop-up.
7. Drag the `prop.ascx` user control you created earlier into the `DeclarativeCatalogPart` control, as shown in Figure 30.8.
8. Save the page.

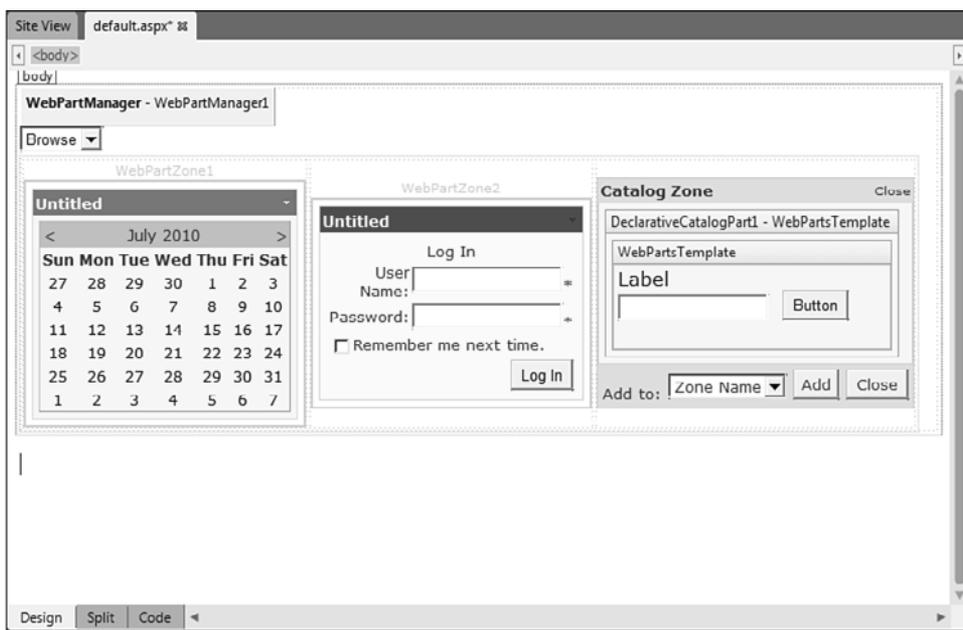


Figure 30.8

The `DeclarativeCatalogPart` is inserted inside the `Catalog Zone`. The `prop.ascx` user control is then added to the `DeclarativeCatalogPart`.

Browse the `default.aspx` page now to test the new Web Parts Catalog.



Error When Browsing Page with User Controls

ASP.NET creates a representation of every user control in memory and uses these “duplicates” to create the user controls on a page. Every instance in memory must be uniquely identified. ASP.NET does this using class names.

When you created the `prop.ascx` user control, you should have changed the `ClassName` attribute in Code View from `WebUserControl1` to `PropControl`. Otherwise, you will get an error that says that `WebUserControl1` exists in multiple places. If so, you likely forgot that step.

When the page is displayed, select `Catalog` from the drop-down and you will see the `Web Parts Catalog`, inside of which is a check box labeled `Untitled`. It would be nice to provide a more descriptive name. Let's do that now:

1. Open `default.aspx` in `Expression Web` and locate the code for the `prop.ascx` user control.
2. Change the code for the user control from this:

```
<uc2:prop id="prop1" runat="server" />
```

to this:

```
<uc2:prop title="Prop Control" id="prop1" runat="server" />
```

3. Save `default.aspx`.

Now browse `default.aspx` and switch to `Catalog` mode. The control now shows a descriptive name. Check the box for the `Prop Control` control, select a `Web Parts` zone, and click `Add` to add the control (see [Figure 30.9](#)).

As you progressed through this chapter, you created a powerful portal-like page that contains robust user interface features, personalization features, and customization features—and you did all of it with only a small amount of code. You also learned how to create user controls so you can build your own `Web Parts` controls for use in `Web Parts` pages.

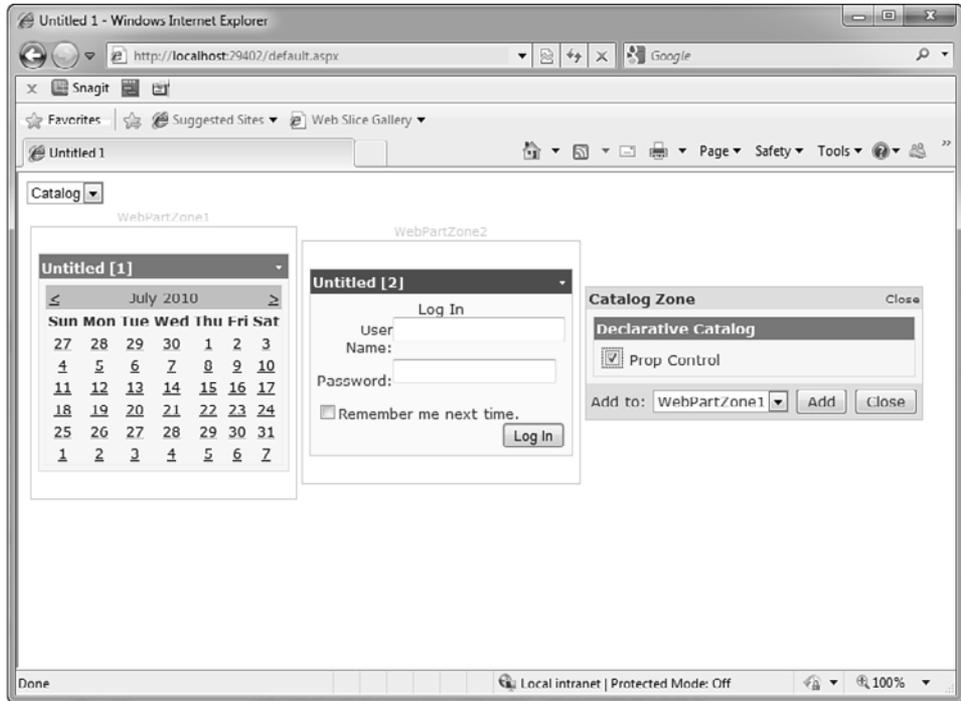
The best way to get comfortable with using `Web Parts` is to experiment with them. This chapter should have provided a solid foundation for doing just that.



tip

You can add `title` attributes to the `Calendar` and `Login` controls you added earlier if you want to give them more descriptive names. If you do, you'll notice that neither the `Calendar` nor the `Login` control recognizes `title` as a valid attribute and `Expression Web` identifies the code as invalid. You can safely ignore `Expression Web`'s complaint about the attribute.

Figure 30.9
You can now add the Prop Control to a Web Parts zone by switching to Catalog mode.



Editing Web Parts Controls

In addition to adding Web Parts controls and moving them between Web Parts zones, you can also add the capability of editing controls. Editing Web Parts controls is accomplished using one or more of the following Web Parts controls:

- `AppearanceEditorPart`—Allows editing of the appearance of a Web Parts control.
- `BehaviorEditorPart`—Allows editing of the behavior of a Web Parts control. For example, you can use the `BehaviorEditorPart` to enable users to specify whether a specific Web Parts control can be minimized or closed.
- `LayoutEditorPart`—Allows the editing of several layout elements of a Web Parts control. For example, you can use `LayoutEditorPart` to add the capability of configuring in which Web Parts zone a particular Web Parts control appears.
- `PropertyGridEditorPart`—Allows the editing of properties of a Web Parts control. The developer of the Web Parts control can specify which properties are editable by the `PropertyGridEditorPart`.

Let's add the capability to edit the appearance of Web Parts controls on the page you created in this chapter:

1. Open `default.aspx` and make sure you are in Design View.
2. Drag an `EditorZone` Web Parts control from the Toolbox, and drop it directly under the display mode drop-down.
3. Drag an `AppearanceEditorPart` control from the Toolbox, and drop it into the `EditorZone` Web Parts control.
4. Configure an `AutoFormat` scheme for the `EditorZone` Web Parts control if you want, and save the `default.aspx` page.

Your page should now look like Figure 30.10.

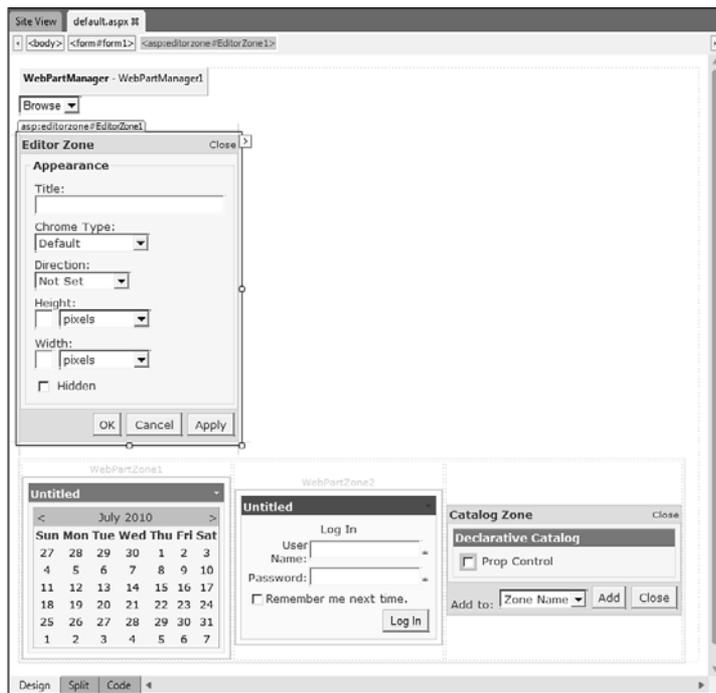


Figure 30.10

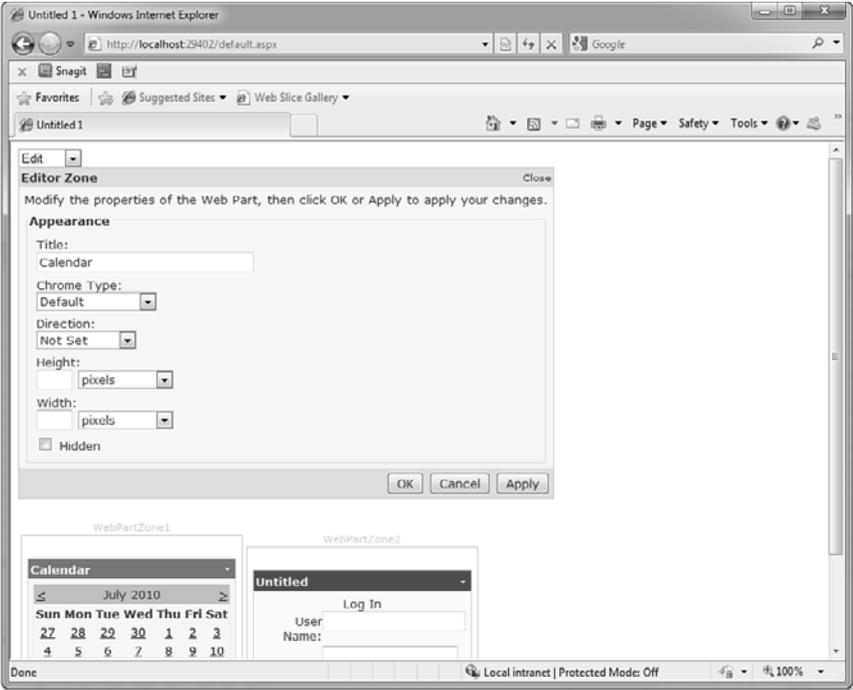
The Web Parts page now contains an `EditorZone` Web Parts control and an `AppearanceEditorPart` Web Parts control so you can edit the appearance of the controls on the page.

Browse the page and select `Edit` from the display mode drop-down. At first glance, selecting `Edit` seems to have no effect. However, if you click the arrow button at the upper right of any Web Parts control, you'll notice there is now an `Edit` menu item.

When you click `Edit`, the `AppearanceEditorPart` Web Parts control is displayed, as shown in Figure 30.11. Make a change to one of the Web Parts controls and click `OK` to apply it.

You've now completed the functionality of the Web Parts page. I think you'll agree that you can create some incredibly powerful functions with minimal effort. I hope the information in this chapter encourages you to experiment more with Web Parts controls and Web Parts pages.

Figure 30.11
You can now edit the appearance of your Web Parts controls using the Edit display mode.



This page intentionally left blank

USING ASP.NET AJAX

What Is Ajax?

The underlying architecture of web applications has always posed a problem with usability. Whenever a user performs an action on a page that requires a postback to the web server, data from all form fields is sent over the network, the page goes blank, and the browser sits in limbo waiting for the web server to send the response. In a typical ASP.NET scenario where a postback reloads the same page after the server processes the data, the speed of the application can be sacrificed because the entire page must reload each time it's posted back.

Ajax is well-suited to providing a user experience that doesn't suffer from such problems. When a page uses Ajax to post back to the server, only the applicable form data is sent to the server in a process known as a *partial postback*. After the server processes that data, it responds with data that the page uses to dynamically update the page as necessary. During this process, the page remains visible in the browser. The result is a user experience very much like a Windows application instead of a typical browser-based experience.

The technology that Ajax uses is called XMLHttpRequest. XMLHttpRequest is a standardized method of exchanging data between an application and a web server. Almost all browsers support XMLHttpRequest, and the latest Opera mobile browser allows for support of Ajax functionality from mobile devices and smart phones.

Microsoft's ASP.NET Ajax

Microsoft's implementation of Ajax is called ASP.NET Ajax. It includes the client-side Ajax (client-side libraries), server-side ASP.NET Ajax (the

server-side controls that are available in the Expression Web toolbox), and the ASP.NET Ajax Control Toolkit (a collection of Ajax controls complete with source code).

Client-Side Ajax

The Microsoft client-side Ajax consists of client scripts and libraries that can be used to add Ajax functionality to any page. The client-side Ajax is the perfect choice, for example, for PHP developers who want to add Ajax functionality to a PHP application using Microsoft's implementation. In fact, one of Microsoft's employees has developed PHP for Microsoft client-side Ajax that does just that, and you can download it from codeplex.com/phpm-sajax.

Server-Side Ajax

Server-side Ajax consists of a series of .NET Framework assemblies (DLL files) that include not only server-side functionality, but also ASP.NET controls that make using Ajax in an ASP.NET page a drag-and-drop endeavor. The following Ajax controls are included with Expression Web:

- **ScriptManager**—The `ScriptManager` control is required on any web form that uses Ajax. It provides the capability to dynamically include client-side scripts used in Ajax functionality.
- **ScriptManagerProxy**—The `ScriptManagerProxy` control is used most often in content pages that use ASP.NET master pages. We'll look at this control in more detail in the "Using Client-Side Ajax" section later in this chapter.
- **Timer**—The `Timer` control enables you to easily perform postbacks of a page (both synchronous and asynchronous) at a regular interval.
- **UpdatePanel**—The `UpdatePanel` control is a container for any control on a web form that takes part in an Ajax partial postback. We'll use an `UpdatePanel` later in this chapter.
- **UpdateProgress**—The `UpdateProgress` control is designed to display a message or some other indicator when an `UpdatePanel` is performing a partial postback.



tip

Ajax is an acronym for Asynchronous JavaScript and XML.



tip

Microsoft is actually trending toward support of jQuery for Ajax functionality instead of using the Microsoft libraries exclusively. You can find more information on jQuery at jquery.com.



note

We'll use the client-side Ajax in the "Using Client-Side Ajax" section of this chapter, but I won't go into great detail about it in this book. If you're interested in reading more about it, read *ASP.NET Ajax in Action* from Manning. It's an excellent book on Ajax.



tip

Any web form that uses Ajax must have exactly one `ScriptManager` control, and it must appear above any control that uses Ajax; otherwise, an error will occur.

Microsoft Ajax Control Toolkit

The Ajax Control Toolkit (which can be viewed and downloaded from <http://www.asp.net/ajaxlibrary/act.ashx>) is a set of Ajax controls that includes full source code. Microsoft released these controls to encourage developers to create their own Ajax controls using Microsoft's server-side Ajax, and some of the coolest Ajax controls are included in the Toolkit.

Developing new Ajax controls and extenders is an advanced programming topic, and Microsoft has provided plenty of excellent documentation on the Ajax Control Toolkit site.



note

Unfortunately, you cannot add controls from the Ajax Control Toolkit to the toolbox in Expression Web.

Adding Ajax Functionality to a Web Form

Adding Ajax functionality to your ASP.NET page is much easier than you might think. Let's create a simple ASP.NET page, and then modify the page to add Ajax functionality to it.

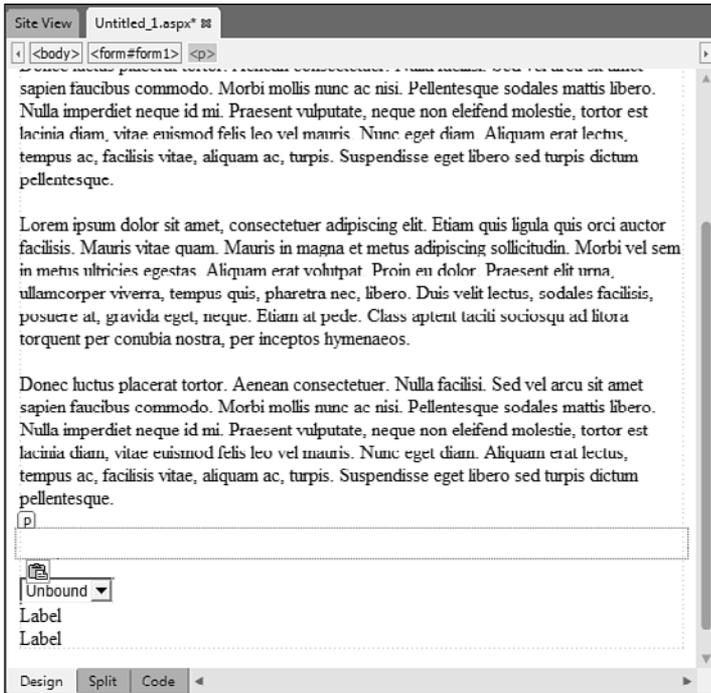
Creating a Site and Page

Let's create a new site and page, and then add Ajax functionality to the page using ASP.NET Ajax:

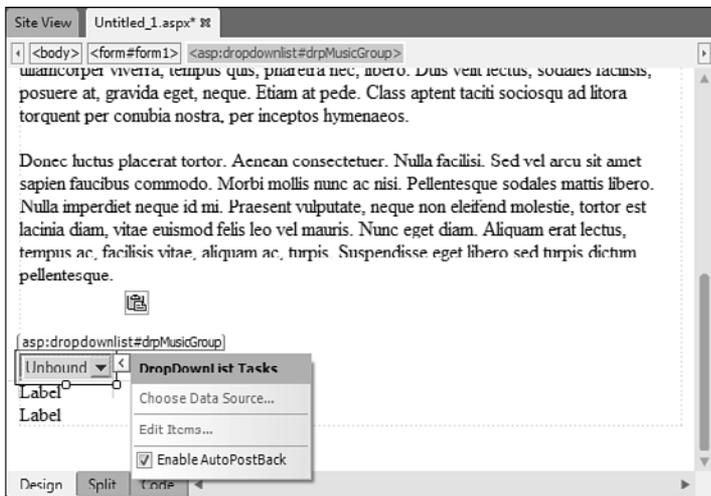
1. Create a new one-page site at a file location on the local computer so you can preview pages using the Microsoft Expression Development Server.
2. Select File, New, Page and select ASPX from the list of page types.
3. Select C# as the language and click OK to create the page.
4. Make sure you are in Design View.
5. Add about 15 lines of text. Any text will suffice, such as text generated by the Lorem Ipsum generator at www.lipsum.com.
6. From the ASP.NET section of the toolbox, add a DropDownList control under the text you added previously. Change the ID property of the DropDownList control to `drpMusicGroup`.
7. Change the `AutoPostBack` property of the DropDownList control to `True`.
8. Add a new line under the DropDownList control and add an ASP.NET Label control. Change the ID property of the Label control to `lblDisplayGroup`.
9. Add a new line and add a new Label control. Change the ID of the new Label control to `lblTime`.
10. Add 20 or so new lines so the text and controls scroll off the design surface.

Your page should now look like the one shown in Figure 31.1. Save the page as `ajax.aspx`.

Next, you need to add some items to the DropDownList control. Click the arrow button at the top of the DropDownList control to display the DropDownList Tasks dialog; then click the Edit Items link as shown in Figure 31.2.

**Figure 31.1**

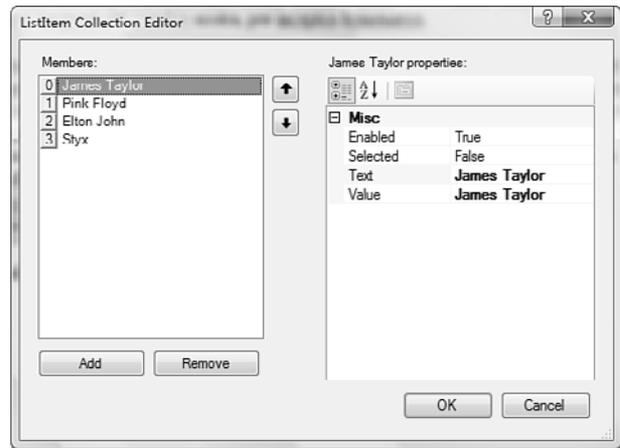
After you've finished creating the Web Form, it should look similar to the one shown here.

**Figure 31.2**

The Edit Items link enables you to easily add and edit items in a DropDownList control.

Click Add and add a few of your favorite music groups, as shown in Figure 31.3; then click OK to add the items.

Figure 31.3
Add a few of your favorite music groups to the DropDownList control.



Adding Server-Side Code

Now that you've created a simple page, you need to add some server-side code to process the page when a new item is selected in the DropDownList control. Switch to Code View and enter the code in Listing 31.1 directly above the opening `<html>` tag.

Listing 31.1 Server-Side Code for Ajax Page

```
<script runat="server">

    protected void Page_Load(object sender, System.EventArgs e)
    {
        lblTime.Text = "It is now " + System.DateTime.Now.ToString();
    }
    protected void drpMusicGroup_SelectedIndexChanged(object sender,
        System.EventArgs e)
    {
        string Group = drpMusicGroup.SelectedValue.ToString();
        lblDisplayGroup.Text = "You selected " + Group;
    }
</script>
```

This code is very simple. When the page loads, it displays the current date and time in the `lblTime` Label control. It also includes code that will run when a new value is selected in the `drpMusicGroup` DropDownList control. That code changes the `lblDisplayGroup` Label control so that it displays the group name you've selected.

note

We haven't begun adding Ajax capability to this page. The purpose of this exercise is to show you how easily you can build a page and then add Ajax functionality to it.

To make this page work, you need to change the code for the `drpMusicGroup` control to hook it up to the code you added to the page. To do that, be sure you're in Code View and add the following to the opening `<asp:DropDownList>` tag on the page:

```
OnSelectedIndexChanged="drpMusicGroup_SelectedIndexChanged"
```

After you've added that, the opening `<asp:DropDownList>` tag should look like this:

```
<asp:DropDownList id="drpMusicGroup" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="drpMusicGroup_SelectedIndexChanged">
```

You're now ready to test the page without Ajax functionality. Save the page and preview it in your browser. When the page loads, scroll down so the drop-down is at the top of the page; then select a new group. When you do, you'll notice a couple of things happening. First, the page will scroll back to the top when you select a new group; then, the date and time change in the `lblTime` Label control. The reason for this is that the entire page is redrawn every time you select a different group.



Error When Previewing Page

If you preview the page and get an error that tells you your control must be placed in a form with a `runat` attribute set to `server`, switch to Code view and ensure that the closing `</form>` tag in your page appears after all ASP.NET controls on the page. To be safe, you can place the closing `</form>` tag right before the closing `</body>` tag on the page.

Ajax is the perfect solution for a situation like this. By adding Ajax functionality to this page, you can update the `lblDisplayGroup` Label control when a new group is selected without changing anything else on the page.

Only two steps are involved in making this page Ajax-enabled:

1. Add a `ScriptManager` control to the page.
2. Add an `UpdatePanel` control to the page and place the controls you want Ajax-enabled into the `UpdatePanel`.

Adding a ScriptManager Control

As I mentioned earlier, any page that uses Microsoft's server-side Ajax must have exactly one `ScriptManager` control on the page. The `ScriptManager` control also must be added to the page above any control that uses server-side Ajax; otherwise, an error will occur.



tip

You can add your own scripts to the `ScriptManager` control as well. We'll cover the concept of adding scripts to the `ScriptManager` in the "Using Client-Side Ajax" section of this chapter.

Ajax uses the `ScriptManager` control to load the necessary client scripts to support *partial post-backs*, the process of posting only certain data back to the web server.

Let's add a `ScriptManager` control to the `ajax.aspx` page:

1. Open the `ajax.aspx` page if it's not already open.
2. Add a new `ScriptManager` control on the page directly above the `drpMusicGroup` control. The `ScriptManager` control will likely show a warning icon as shown in Figure 31.4. If you don't already have a `web.config` file that contains configuration information for the 3.5 .NET Framework, you need to add one before you can use ASP.NET Ajax.

Figure 31.4

If you don't already have a `web.config` file that contains .NET Framework 3.5 configuration information, Expression Web displays a warning icon on the new `ScriptManager` control.

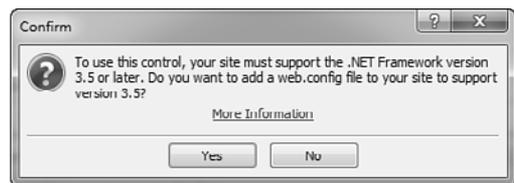


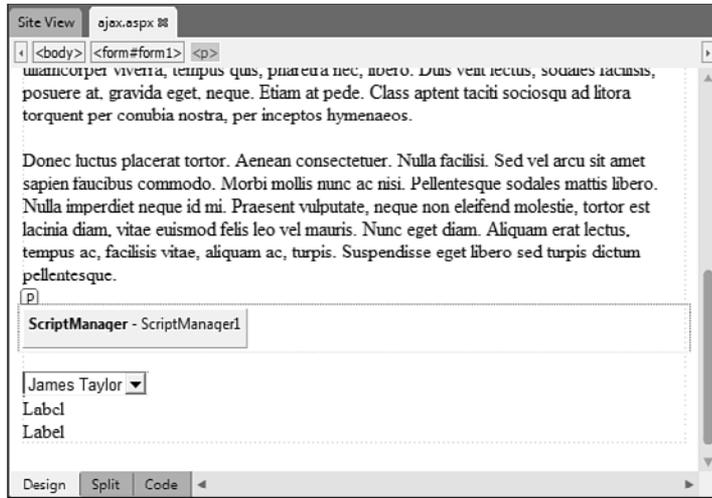
3. To add the necessary configuration information, click the `asp:scriptmanager` link on the `ScriptManager` control shown previously in Figure 31.4. Expression Web displays a message informing you that you need to update or add a `web.config` file to support the .NET Framework 3.5, as shown in Figure 31.5. Click Yes to add the necessary information to your site.
4. Save the page.

Your page should now look similar to the one shown in Figure 31.6.

Figure 31.5

Expression Web can add the necessary information to your site to support the .NET Framework 3.5.



**Figure 31.6**

The ASP.NET page now has a `ScriptManager` control on it and can take advantage of server-side Ajax.

Adding an `UpdatePanel` Control

The `UpdatePanel` control is extremely powerful. Any control placed within an `UpdatePanel` control automatically becomes an Ajax enabled control. In the case of the `ajax.aspx` page, we want the `drpMusicGroup` and `lblDisplayGroup` controls to be Ajax enabled so that when an item is selected in the drop-down, the label is updated to reflect the selected item via Ajax. Accomplishing this is going to be a lot easier than you might think.

You need to add a new `UpdatePanel` control that contains both controls. To do that, add the `UpdatePanel`; then cut and paste the controls into it:

1. Click to the right of the `ScriptManager` control, and press Enter to insert a new line above the `drpMusicGroup` control.
2. Add a new `UpdatePanel` control from the AJAX section of the ASP.NET Toolbox.
3. Select the `drpMusicGroup` and the `lblDisplayGroup` controls and select Edit, Cut to remove them and place them on the Windows Clipboard.
4. Click inside the new `UpdatePanel` control and select Edit, Paste to paste the controls inside the `UpdatePanel` control.

Believe it or not, you have just finished Ajax-enabling the page. If you preview the page in your browser and select an item in the `drpMusicGroup` drop-down, the `lblDisplayGroup` label will be populated with the appropriate group using ASP.NET Ajax. When that happens, you'll notice that the page will no longer scroll to the top and the time that appears in the `lblTime` Label control reflects the time that the page was first loaded and not when you selected the group in the drop-down.



tip

If you don't see the `UpdatePanel` control after inserting it, select View, Visual Aids, Show to turn on Visual Aids.

Using Client-Side Ajax

Client-side Ajax enables you to write client-side scripts that interact with your ASP.NET application. Suppose the code that runs when a new group is selected in the `drpMusicGroup` drop-down might take several seconds to run and return a value. In such a scenario, displaying a message to the user is a good practice, and client-side Ajax lets you easily do that.

In this section, we use client-side Ajax to display a status message while the Ajax request is executing.



note

You can use the `UpdateProgress` control to display a message, but it's a good idea to know how to take advantage of client-side Ajax in case you want a more customized solution.

Adding a `<div>` to the Web Form

You'll need to add a `<div>` to the `ajax.aspx` page. The `<div>` will be used to display the status message.

Follow these steps to add the `<div>`:

1. Open the `ajax.aspx` page if it's not already open.
2. Click to the right of the `drpMusicGroup` `DropDownList` control, and press `Enter` to add a new line.
3. Add a new `<div>` control to the new line from the `Tags` section in the `HTML` section of the `Toolbox`.
4. In the `Tag Properties` panel, change the `id` property of the `<div>` to `WaitIndicator`.

Creating the Client Library

The client library is a JavaScript file used to interact with the page during an Ajax partial postback. Follow these steps to create the client library:

1. Select `File, New, Page` to add a new file to the site.
2. Select `JavaScript` from the list of file types and click `OK`.
3. Select `File, Save` and save the JavaScript file as `ClientLibrary.js`.

Now you must add the JavaScript that will update the `<div>` you added earlier with a status message. Add the following JavaScript code to the `ClientLibrary.js` file:

```
Sys.Net.WebRequestManager.add_invokingRequest(App_InvokeRequest);
Sys.Net.WebRequestManager.add_completedRequest(App_CompleteRequest);

function App_InvokeRequest(sender, eventArgs)
{
    var waitDiv = $get("WaitIndicator");
```

```
        waitDiv.innerHTML = "Please wait...";  
  
    }  
  
    function App_CompleteRequest(sender, eventArgs)  
    {  
        var waitDiv = $get("WaitIndicator");  
        waitDiv.innerHTML = "";  
    }  
}
```

**note**

A detailed discussion of client-side Ajax is outside the scope of this book. For detailed information on using client-side Ajax, see <http://msdn.microsoft.com/en-us/library/bb397536.aspx>.

The first few lines of code add the event handlers that fire when the Ajax request begins (the `invokingRequest` event) and ends (the `completedRequest` event). These lines use the `Sys.Net.WebRequestManager` object, which is provided by client-side Ajax.

The `App_InvokeRequest` and `App_CompletedRequest` functions set the `innerHTML` property of the `<div>` you added earlier using the `$get` method the client-side Ajax provides. The `$get` method returns a reference to the control whose ID is passed to it.

After you've added the code to the `ClientLibrary.js` file, save the file.

Adding the Client Script to the ScriptManager Control

The final step is to add the `ClientLibrary.js` file to the `ScriptManager`'s `Scripts` collection. As I mentioned earlier in this chapter, you can add your own client scripts to the `ScriptManager` control. When you do, `AjaxASP.NET Ajax` loads your scripts dynamically.

Follow these steps to add the `ClientLibrary.js` script to the `ScriptManager`'s `Scripts` collection:

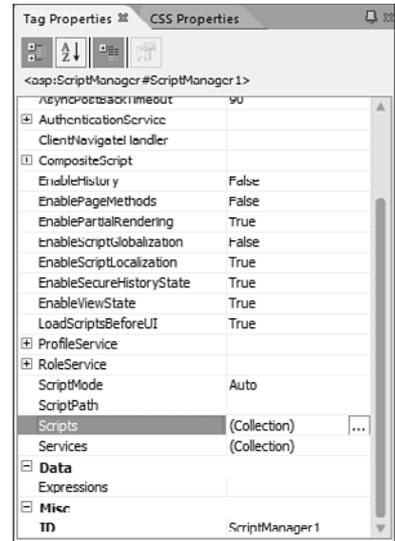
1. Select the `ScriptManager` control you added to the `ajax.aspx` page earlier.
2. Click the ellipse button for the `Scripts` property in the `Tag Properties` panel as shown in Figure 31.7.
3. Click the `Add` button in the `ScriptReference Collection Editor`.
4. Change the `Path` property of the new `ScriptReference` to `ClientLibrary.js` (see Figure 31.8).
5. Click `OK` and save the page.

When the `ajax.aspx` page is browsed, the `ScriptManager` loads the `ClientLibrary.js` script.

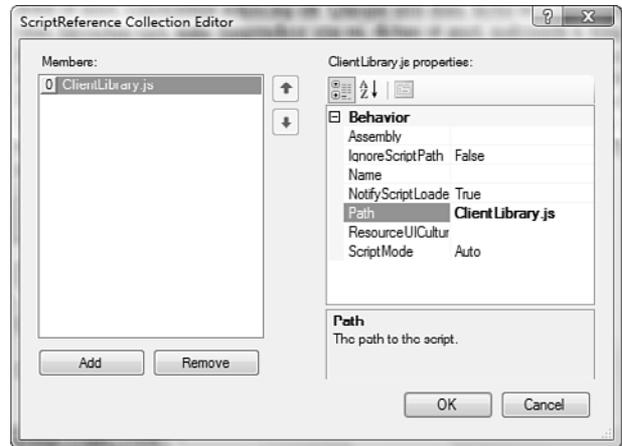
If you browse to the page right now, the script we added displays a message that says `Please wait...` while the Ajax request is being processed. However, because the Ajax request is almost instantaneous in our test page, we need to build in a delay so we can see the effect.

Figure 31.7

You can use the Scripts property in the Tag Properties panel to add your own scripts to the ScriptManager control.

**Figure 31.8**

The Path property points to the script you want to add.



Open the `ajax.aspx` page and add the following code to the `Page_Load` event you added earlier:

```
if (IsPostBack) System.Threading.Thread.Sleep(7000);
```

After you add that code, the `Page_Load` event looks like the following code:

```
protected void Page_Load(object sender, System.EventArgs e)
{
    if (IsPostBack) System.Threading.Thread.Sleep(7000);
    lblTime.Text = "It is now " + System.DateTime.Now.ToString();
}
```

This code makes the page wait seven seconds when the Ajax partial postback takes place, enabling you to see the `Please wait...` message. You can now save the page and preview it to see the status indicator displayed.

USING PHP

An Introduction to PHP

Although ASP.NET is a popular standard when it comes to server-side application development, it's certainly not the only one. Scripting technologies such as ColdFusion, Java Server Pages, and even the old stalwart Perl have all been used to perform similar work on the server to generate dynamic content. One of the most popular server-side scripting technologies, PHP, is so ubiquitous that Microsoft has decided to provide support for it in Expression Web.

PHP (a recursive acronym that stands for PHP: Hypertext Preprocessor) was designed as an alternative to Perl because it is better suited for generating dynamic web content. Whereas Perl requires all content to be generated by the Perl script, PHP's model is much more like that of classic ASP code. PHP code is embedded directly into pages, but only content that is embedded inside special characters called PHP delimiters is interpreted as PHP code. All other content on the page is sent as is. This enables most of a page to be written as standard HTML, while only the portions of the page that need to be generated dynamically utilize the PHP code.

PHP Syntax

Whether you are actually coding in PHP or simply using other people's PHP code in your pages, it is helpful to understand a bit about PHP syntax and how PHP programs are structured. Like ASP code, PHP is embedded directly into a page and requires special tags (called *delimiters*) to let PHP know what can be output directly and what needs to be handled by the PHP interpreter.

PHP code can be delimited in several ways. The first is through the use of the `<script>` tag:

```
<script language="php">
echo "Welcome to PHP!\n";
</script>
```

This method can cause problems with some HTML authoring tools because the content between the script tags is not recognizable as HTML content. Therefore, this method has been deprecated, although it can be found in older PHP pages.

The recommended syntax for PHP is to use `<?php ?>` instead. This syntax is referred to as *full tags*. For example:

```
<?php
echo "Welcome to PHP!\n";
?>
```

In some cases, you might find PHP documents using a shorter version of this syntax referred to as *short tags*. For example:

```
<? echo "Welcome to PHP!\n"; ?>
```

You might even find them using ASP-style tags:

```
<% echo "Welcome to PHP!\n"; %>
```

Use of short tags is optional and can be disabled on a particular web host. Sticking with full tags is the best option for code portability.

Similar to JavaScript and other languages that share a C-like syntax, PHP code is case sensitive. Statement lines are followed with a semicolon, as you can see with the various `echo` statements listed previously.

PHP Comments

PHP comments make code easier to understand for people not familiar with the language and for other PHP developers reading your code. Three styles of comments are supported in PHP: C-style, C++/Java-style, and Perl-style. Here is an example of each style of comment:

```
/* C style */
// C++/Java style
# Perl style
```

The C-style comments can occupy more than one line, and everything between the opening and closing identifier is treated as part of the comment. With C++ and Perl-style comments, text up to the next newline is treated as a comment.

**tip**

The `echo` statement seen here is used to output information to the browser window when the page is browsed.

PHP Variables

PHP variables are declared with a preceding dollar sign (\$) followed by the name. The name must consist of a leading letter (A–Z, a–z) or underscore, followed by any number of letters, underscores, or numbers. Variable names embedded in a string will be substituted by the value of the variable at runtime. Here is an example:

```
<?php
$first_name = 'John';
$last_name = 'Doe';
$age = 33;
echo "name: $last_name, $first_name age: $age\n";
?>
```

In this example, three variables are declared: `$first_name`, `$last_name`, and `$age`. When the page runs, the variable values are output to the page and the following text is displayed:

```
name: Doe, John age: 33
```

The type of a variable is not typically assigned by the programmer. Instead, the PHP interpreter sets the type based on the type of data that is first assigned to it.

Table 32.1 lists the variable types available in PHP.

Table 32.1 PHP Variable Types

Type	Description
Boolean	True or False
Integer	–2, 0, 100, 3500, 0xff (hex), and so on
Float	–6.0, 1.2, 0.0, 1.3e4, and so on
String	“Sample string text”
Array	A collection of objects indexed by a key
Object	A reference to an instance of a class
Resource	A handle to an external resource such as a file or a database connection
NULL	A variable with no value

PHP Program Flow

PHP supports language structures that are similar to those used by many other languages. It can use `if/else/elseif`, `while`, `do-while`, `for`, `foreach`, and more. Each of these structures can use either of two syntaxes. One will be more familiar to C++ or JavaScript programmers due to the use of braces to mark the beginning and end of a block. The alternative style will likely seem more

familiar to Visual Basic programmers because it uses keywords to mark the end of a block instead of braces, as shown here:

```
<?php
// JavaScript-style 'if' statement
if ($user_role == "administrator") {
    echo "administrator login successful.\n";
}
else {
    echo "user login successful.\n";
}

// Alternative style 'if' statement
If (%user_role == "administrator") :
    echo "administrator login successful.\n";
else :
    echo "user login successful.\n";
endif;
?>
```

The alternative style can also make the code more readable if the beginning and ending of the block are in different PHP script blocks:

```
<php? if (%user_role == "administrator") : ?>
  <br />
  <span>administrator login successful</span>
  <br />
<php? else : ?>
  <br />
  <span>user login successful</span>
  <br />
<php? endif; ?>
```

Functions

Blocks of PHP code can be inserted into functions, allowing the code to be called from the main program flow or from another function. Functions should be used for code that needs to be called from multiple locations. They can also be used to pull out long sections of code from the main program flow, making it easier to read and follow. In this example, the code in the initialize function is not executed until it is called from the main program flow:

```
<?php
function initialize ()
{
    echo "initializing...\n";
```

note

It is outside the scope of this book to go into thorough detail about the use of PHP. If you are interested in learning more about PHP and how to write dynamic web content with it, read *PHP 5 Unleashed* from Sams Publishing (ISBN 067232511X).

You can also learn about PHP, follow tutorials, and browse documentation by visiting the PHP group's site located at www.php.net.

```
}  
  
echo "calling initialize\n";  
initialize();  
echo "initialization complete\n";  
?>
```

Installing PHP

Installing PHP is straightforward enough, but until the release of IIS 7, getting it to work correctly with IIS was a fairly involved process. The Common Gateway Interface (CGI) for IIS can call PHP executable reliably and safely, but prior to IIS 7, this method was inefficient because it required the creation of a new Windows process for each request.

A more efficient method of calling PHP utilizes the Internet Server Application Program Interface (ISAPI) extension. With ISAPI, IIS calls into the PHP ISAPI DLL with its own thread. This is efficient because it bypasses the need to create a new process for each request, but a potential problem still exists with this method. The ISAPI interface demands that the code it calls be thread-safe, meaning multiple threads can safely access the same block of code in the same memory space. Although the core PHP is available in a thread-safe version, it doesn't perform as well as the nonthread-safe (NTS) version. A fair number of popular PHP libraries aren't thread-safe either, so if you intend to use them, you'd be forced into the standard CGI model.

The FastCGI protocol was created to address these problems. FastCGI provides a pool of processes to host CGI calls. When a call finishes, the process is returned to the pool instead of being terminated, so it can be reused. This greatly speeds up the process of calling CGI programs because it eliminates the process startup penalty. In addition, because CGI applications are running as their own process space with protected memory, thread safety issues don't apply.

To show its commitment to supporting the use of PHP on IIS, Microsoft has created FastCGI components for both IIS 6 and IIS 7. In this section, I show you how to get PHP up and running with IIS 7 using FastCGI. But first, I briefly cover how to install IIS itself.



tip

IIS 6 users can download the FastCGI extension for IIS 6 from www.iis.net/downloads/default.aspx?tabid=34&g=6&i=1521.

IIS 7 users can download FastCGI from www.iis.net/downloads/default.aspx?tabid=34&i=1299&g=6.

Installing IIS 7 and FastCGI

To use FastCGI, you must have IIS 7 for Windows 7, Windows Vista SP1 or later, or Windows Server 2008. Because IIS is not part of the default installation for the operating system, you will have to add it yourself.

Installing IIS 7 and FastCGI in Vista and Windows 7

You install IIS 7 in Vista and Windows 7 through the Control Panel. Follow these steps to do it:

1. Click Start, Control Panel.
2. Select Classic View, double-click Programs and Features, and then select Turn Windows Features On or Off.
3. Check the Internet Information Services box, as shown in Figure 32.1.

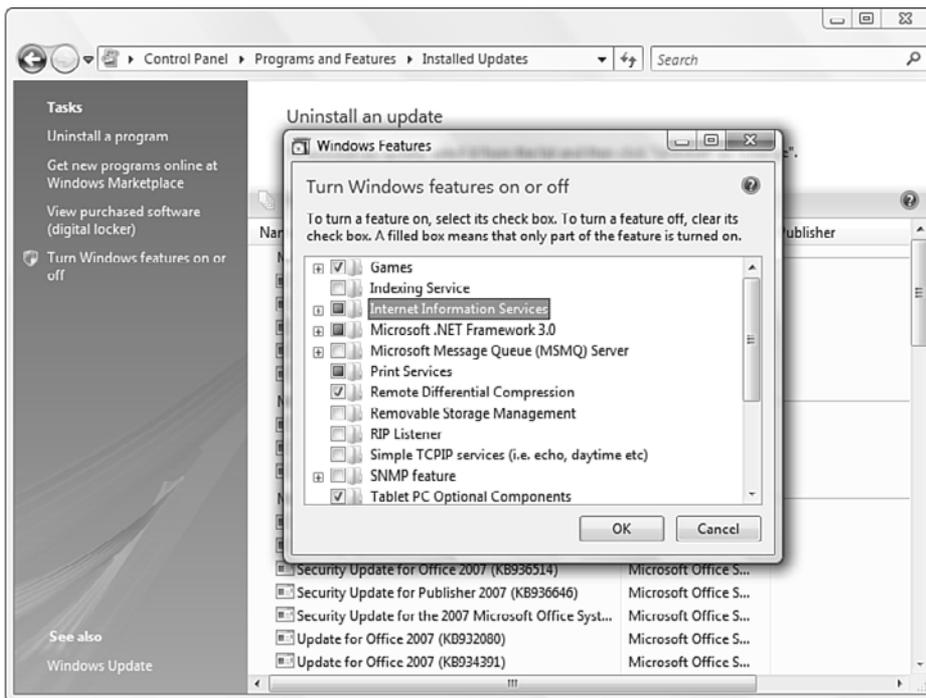


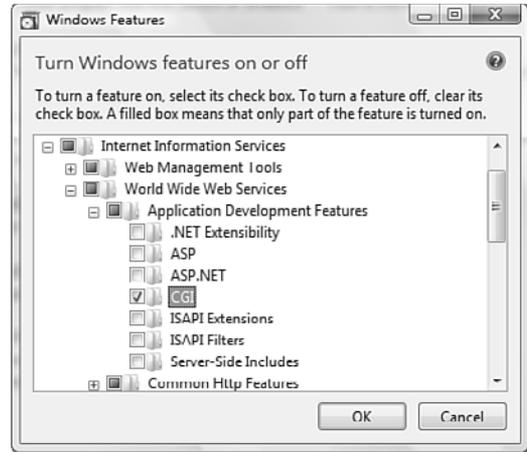
Figure 32.1
IIS is not installed in Windows Vista or Windows 7 by default.

To use FastCGI, you must now enable the CGI feature in IIS. Continuing in the Windows Features dialog box, do the following:

1. Expand the Internet Information Services tree node.
2. Expand World Wide Web Services.
3. Expand Application Development Features.
4. Check the CGI box, as shown in Figure 32.2.
5. Click OK when you are finished.

Figure 32.2

You must enable the CGI feature to use FastCGI.



Installing IIS and FastCGI in Windows Server 2008

Installing IIS 7 in Windows Server 2008 is a somewhat different process. You must use the Server Manager to enable the Web Server role for the server. Follow these steps to enable this role:

1. Click Start, Administrative Tools, Server Manager.
2. Select Add Roles.
3. On the Select Server Roles page, select the Web Server (IIS) check box.
4. Click Next.
5. Select any role services you want to install at the same time. For our purposes, be sure the CGI option is selected. This service is necessary for FastCGI to function.
6. Click Next.
7. Click Install.

Installing PHP

At the time of this writing, PHP version 5.4 is the current release of PHP, and it can be downloaded from www.php.net/downloads.php. This download page is shown in Figure 32.3. The NTS binaries are preferred for use with FastCGI and will perform better. Don't download the installer version because it is unnecessary when using FastCGI. Simply download the Windows binaries Zip file.

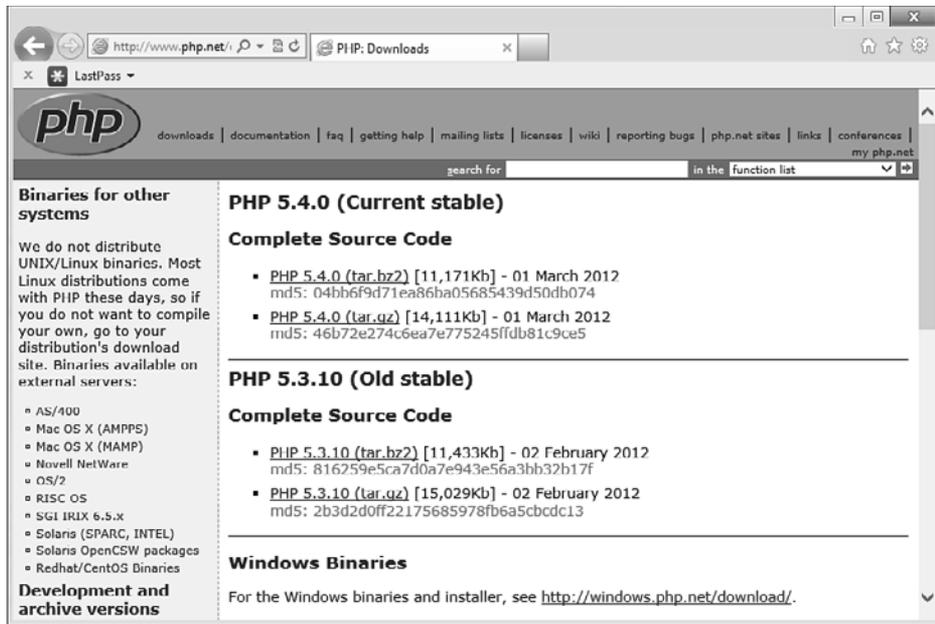


Figure 32.3
Download the most recent version of PHP directly from www.php.net.

If you need to use an older version of PHP for a particular application, you can use FastCGI with versions 4.4 and higher. Versions 5.2.3 and higher, however, include optimizations that enable better performance when running on Windows with FastCGI.

When the download is complete, extract the contents of the Zip file to C:\PHP. Then, in that folder, rename the file `php.ini-development` to `php.ini` when you are developing your site and the file `php.ini-production` to `php.ini` when you are ready to release.



tip

Because of problems in version 5.2.4 that cause performance degradation, it's recommended that you not use version 5.2.4. These problems have been fixed in version 5.2.5 and later.

Configuring the Microsoft Expression Development Server for PHP

After you've installed PHP, you can configure your preview settings in Expression Web so you can use the Microsoft Expression Development Server to test your PHP pages in cases where IIS is not available. To configure PHP for use with the Microsoft Expression Development Server, follow these steps:

1. Select Tools, Application Options.

2. In the PHP section of the General tab, click Browse and browse to the `php-cgi.exe` file you installed previously.
3. Click OK.

Alternatively, you can use the Preview tab in the Site Settings dialog to specify preview options for a particular site.

Enabling PHP for IIS Using FastCGI

After you have IIS 7 and PHP installed, you need to tell IIS where it can find the PHP installation and which file extensions to use for it. This is done in the IIS Manager; this configuration process is the same for Vista SP1, Windows 7, and Windows Server 2008. Follow these steps to enable PHP using FastCGI:

1. Click Start, type **INETMGR** in the search box, and press Enter.
2. Select the server name in the pane on the left, and look for the Handler Mappings item in the Features View (see Figure 32.4). Double-click this item.

Figure 32.4
The IIS Manager is used to configure PHP.



3. Click Add Module Mapping from the Action menu on the right. This opens the Edit Module Mapping dialog box, shown in Figure 32.5.
4. Type `*.php` in the Request Path.

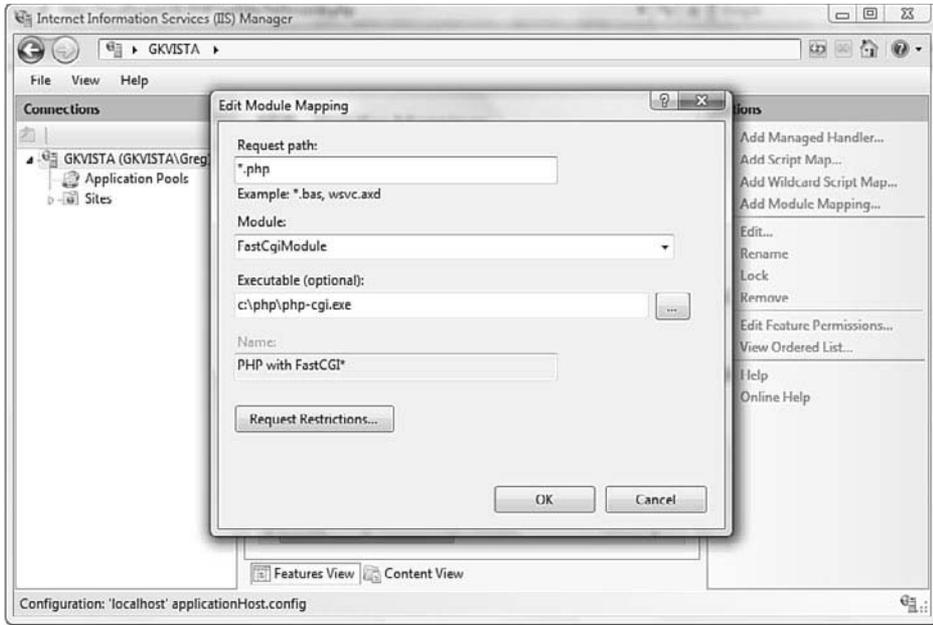


Figure 32.5
You must tell IIS to route PHP files to the PHP CGI executable.

5. Select FastCgiModule from the Module drop-down list.
6. Enter the path to your PHP CGI module under Executable. If you followed the instructions for installing PHP earlier, it will be `C:\php\php-cgi.exe`.
7. Type **PHP with FastCGI** in the Name field.
8. Click OK, and then click Yes when asked whether you want to create a FastCGI application for this executable.
9. Close the IIS Manager.



tip

If you want to use impersonation in your site, change the value of `fcgi.impersonate` in the `php.ini` file to 1.

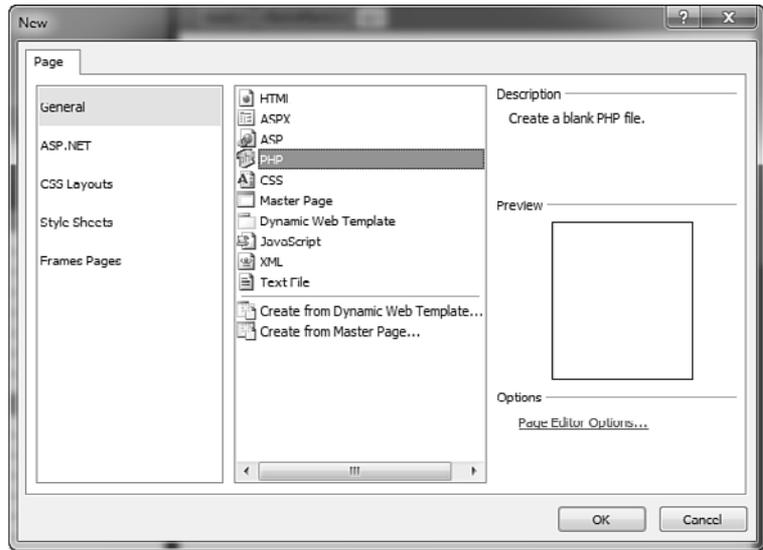
Creating PHP Pages

With IIS configured and PHP enabled, you can now create a PHP page in Expression Web to test the functionality of the FastCGI application. Follow these steps to create a sample PHP site for testing:

1. In Expression Web, select Site, New Site.
2. Under the General tab, select Empty Site.
3. Enter a name for the site, such as **PHPTestSite**. Click OK.

4. Now, create a PHP page. Select File, New, Page to open the New dialog box.
5. Select PHP as the file type in the General section as shown in Figure 32.6; then click OK.

Figure 32.6
You can create PHP pages as easily as any other type of page.



6. Select Code View. In the Body section, enter the following PHP code:

```
<?php  
echo "Hello World!\n";  
?>
```

7. Save the page as `helloworld.php`. Figure 32.7 shows the page in Code View.

Previewing the Page

Even if you have IIS7 installed and configured to use PHP, Expression Web will not use it by default when you preview in the browser. Instead, it will use the Microsoft Expression Development Server. You need to either configure the site you just created to use IIS for the preview or tell the Microsoft Expression Development Server where to find your PHP CGI executables. Fortunately, both of these configuration options are done in the same place in the Site Settings dialog. Follow these steps to configure the preview of PHP pages using the Microsoft Expression Development Server:

1. Select Site, Site Settings to open the dialog.
2. Click the Preview tab.



```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1
2 <html xmlns="http://www.w3.org/1999/xhtml">
3
4 <head>
5 <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
6 <title>Untitled 1</title>
7 </head>
8
9 <body>
10 <?php
11 echo "Hello World!\n";
12 ?>
13 </body>
14
15 </html>
16
```

Figure 32.7

Enter a simple PHP block in the body section of the sample page.

3. Make sure the Preview Using Website URL button and the Use Microsoft Expression Development Server check box are selected.
4. If you have not configured the PHP CGI executable path in the Application Settings dialog, or if you want to use a different version for this site, select the Use a PHP Executable for Only this Web Site option.
5. If necessary, enter the path to your PHP CGI executable or browse to it. It will be `C:\php\php - info . exe` if it was installed using the instructions in the previous section.
6. Click the Preview in Browser button. The page should be parsed by the PHP program, as shown in Figure 32.8.

If you'd rather configure the preview feature to use IIS running locally instead, follow these steps:

1. Configure a virtual directory in the IIS Manager that points to your development site path. Right-click Default Web Site, and select Add Virtual Directory.
2. In the Add Virtual Directory dialog, type **PHPTestSite** as the alias; then enter or browse to the path where your site project files are located.

**tip**

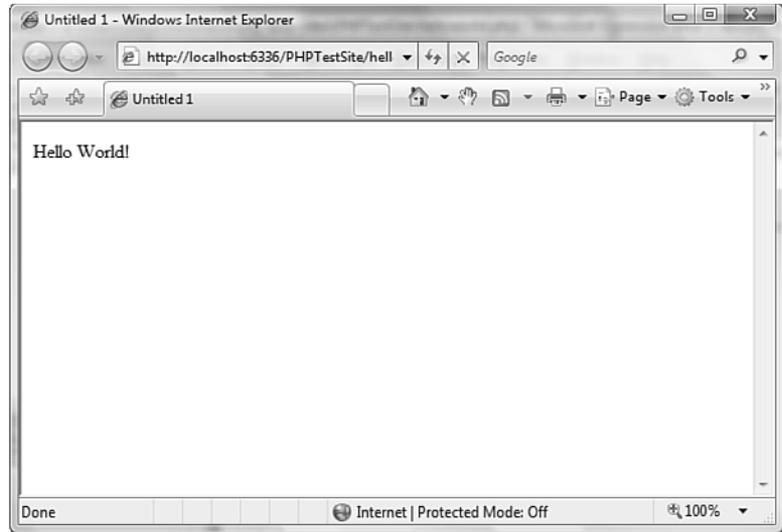
You can configure the location of the PHP executable in the Application Settings dialog as well. If you configure it in the Application Settings dialog, it will affect all sites you create with Expression Web.

**tip**

If the `PHP.ini` file is not correctly configured for displaying PHP, Expression Web displays a dialog asking whether you want the file to be configured. Make sure you answer Yes to this so that Expression Web can fix the `PHP.ini` file.

Figure 32.8

View the page in the browser. If all went well, you'll see the PHP-generated text.



3. Right-click the new PHPTestSite item in the tree, and select Edit Permissions. This opens the properties window for the folder. Click the Security tab.
4. Click the Edit button, and then click Add in the Permissions dialog.
5. In the Select Users or Groups dialog, type **IIS_IUSRS**. Then click OK.
6. Click OK again.
7. In Expression Web, select Site, Site Settings to open the Settings dialog.
8. Click the Preview tab.
9. Select the Preview Using Custom URL for the Web Site radio button.
10. Type `http://localhost/PHPTestSite/` for the custom URL.
11. Click OK.
12. Click the Preview in Browser button to view the page in the browser.



PHP Code Doesn't Seem to Execute When I Preview It in the Browser

Sometimes your PHP code won't execute when you preview the page in the browser; instead, the PHP code appears in the page source just as it is in Expression Web. If FastCGI or PHP isn't configured correctly, IIS won't know how or won't be able to parse the PHP code in your page. PHP pages must be processed by the php-cgi engine to display correctly.

To resolve this problem, check the settings for the FastCgiModule configuration in the IIS Manager. Pay particular attention to the module, request path, and executable entered for the FastCgiModule mapping.

You also should verify that PHP itself is configured correctly. You can do this from the command line by executing the PHP command-line interpreter (CLI) directly, with the `-info` parameter. From the command line, type `php -info` and press Enter. You might have to change directories to the `C:\php` folder first, if it isn't in your path.

PHP in Design View

PHP script blocks cannot be processed in Design View in the same way they are in the browser. However, you can see where PHP code blocks are using the formatting marks available for script blocks.

Formatting marks are not enabled by default. Do the following to show them:

1. Select Design View.
2. Select View, Formatting Marks from the menu.
3. Make sure the Show item is selected.
4. Select View, Formatting Marks again.
5. Be sure Script Blocks is selected.
6. PHP blocks appear with the script block icon, as shown in Figure 32.9.

As with other types of scripts, you can view the code within the block by double-clicking the script block icon (see Figure 32.10).

PHP in Code View

Although Design View is somewhat limited in its handling of PHP, Code View offers some additional features. You can customize the display of PHP code in the editor and use IntelliSense to help you write PHP code.

PHP Syntax Highlighting

Syntax highlighting makes editing in Code View easier because it makes code blocks and the elements contained in them stand out from the surrounding code. With syntax highlighting, you can customize each PHP item with the color of your choice. You can customize foreground and background colors and even set font styles such as bold, italic, or underline specifically for each PHP item.

Expression Web is configured with default values for each of the formatting options. (You can change them if you don't like the defaults.) Follow these steps to configure syntax highlighting for PHP elements:

1. Switch to Code View.
2. Select Tools, Page Editor Options from the menu.
3. Select the Color Coding tab, as shown in Figure 32.11.
4. Make sure the Code View Settings radio button is selected.
5. Scroll down the list of display items until you see the PHP items.
6. Select each PHP item in turn and select the foreground color, background color, and any custom font styles you like. Use the Automatic option for the foreground and background colors so those items are determined automatically by the editor.

**tip**

You can also show this tooltip by positioning the cursor at any point in a parameter list and pressing Ctrl+Shift+Space.

Using IntelliSense with PHP

IntelliSense is a feature of Microsoft's code editors that helps you speed up coding by giving you options to select code from a list as you type. IntelliSense also provides you with tooltips to fill in parameters for function calls. IntelliSense works inside a PHP script block regardless of the type of PHP tags you use to identify the block (`<?php ?>`, `<? ?>`, or `<script language="php"> </script>`).

Press Ctrl+L inside a PHP script block to bring up a list of available PHP functions, as shown in Figure 32.12. You can quickly locate a function from the long list by typing one or more letters of the function name right before or after you activate the list. Alternatively, simply scroll down the list using the mouse or the arrow keys and choose a function manually.

When the function you want to add is highlighted in the list, you can enter it in the editor simply by pressing Tab.

Figure 32.11
 PHP syntax highlighting enables you to customize colors and font styles for various PHP code elements.

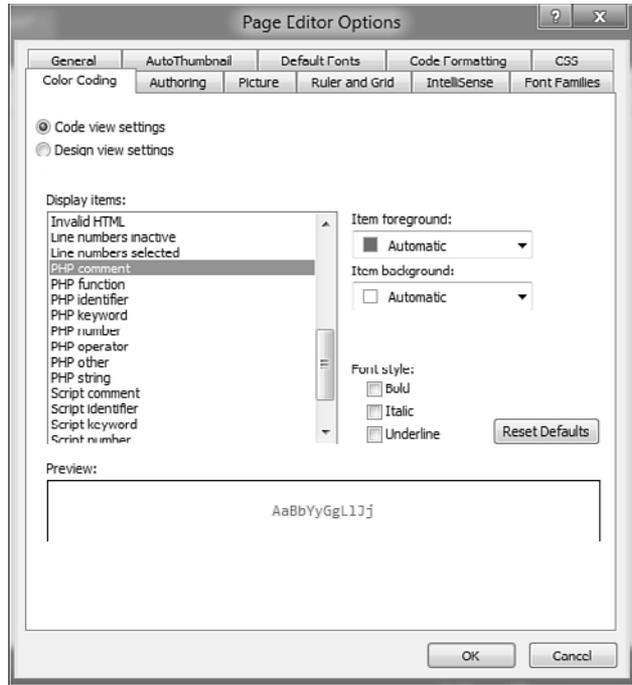
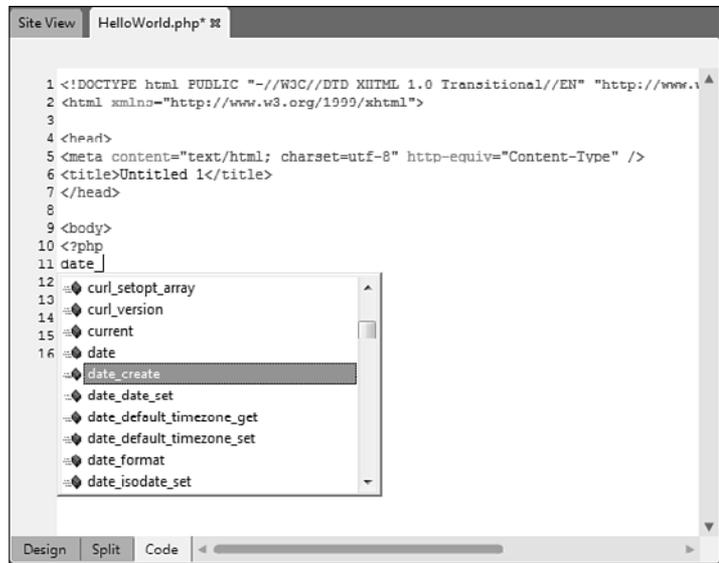
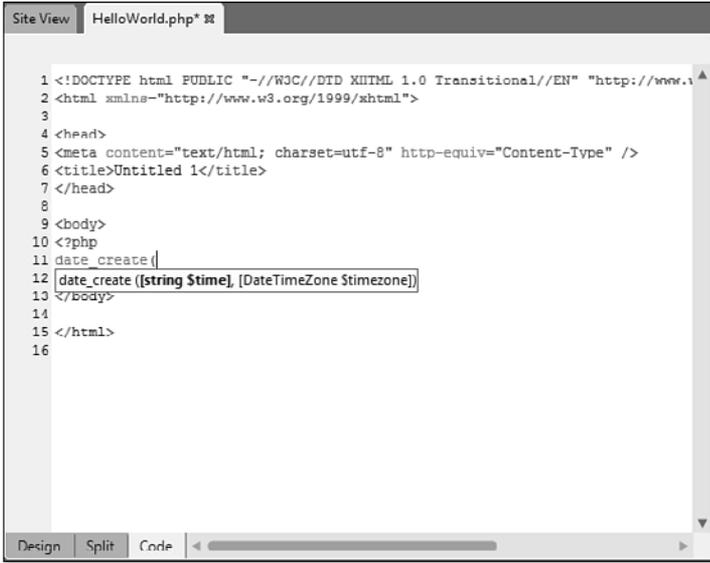


Figure 32.12
 IntelliSense for PHP lets you easily discover PHP techniques and makes entering PHP code faster.



If the selected code is a function, IntelliSense shows you a tooltip with the parameters required by the function when you enter the left parenthesis, as shown in Figure 32.13. As you enter parameters and type the comma separating them, the next parameter to enter appears bolded in the tooltip.



The screenshot shows a code editor window titled 'HelloWorld.php'. The code is as follows:

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/1999/xhtml">
2 <html xmlns="http://www.w3.org/1999/xhtml">
3
4 <head>
5 <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
6 <title>Untitled 1</title>
7 </head>
8
9 <body>
10 <?php
11 date_create(
12 date_create ([string $time], [DateTimeZone $timezone])
13 </body>
14
15 </html>
16
```

A tooltip is displayed over the opening parenthesis of the `date_create` function call on line 11. The tooltip contains the signature: `date_create ([string $time], [DateTimeZone $timezone])`. The parameter `[string $time]` is highlighted in a light blue box, and the parameter `[DateTimeZone $timezone]` is bolded.

Figure 32.13

Tooltips guide you through the process of entering parameters for function calls.

Global variables in PHP are provided by the environment and include several collections, such as session variables, form elements, cookies, and environment variables. IntelliSense can simplify the addition of global variables in your code as well. In PHP, global variables are preceded by a `$_`. If you type those characters into the editor inside a PHP code block, IntelliSense presents you with a list of global variables (see Figure 32.14). As with the function list, you can type a partial name and then press Tab to complete the entry.

Setting PHP-Specific IntelliSense Options

If you prefer not to use IntelliSense, or want only some of the IntelliSense features, you can enable or disable them in the Page Editor Options dialog box.

The following are some PHP IntelliSense options you can enable or disable:

- **PHP Global Variable Completion**—Controls whether to display a list of global variables when you type `$_`.

Figure 32.14

You can choose from a list of global variables when you enter the `$_` prefix.



- **PHP Parameter Information**—Choose whether to display information about parameters when entering a function call.
- **PHP Function Categories**—You can reduce the size of the IntelliSense function list by deselecting categories of functions you might never need.

Follow these steps to set IntelliSense options:

1. Select Tools, Page Editor Options.
2. Select the IntelliSense tab.
3. In the Auto Popup section, select or deselect the IntelliSense items as you see fit, as shown in Figure 32.15.
4. Click OK when you are finished.

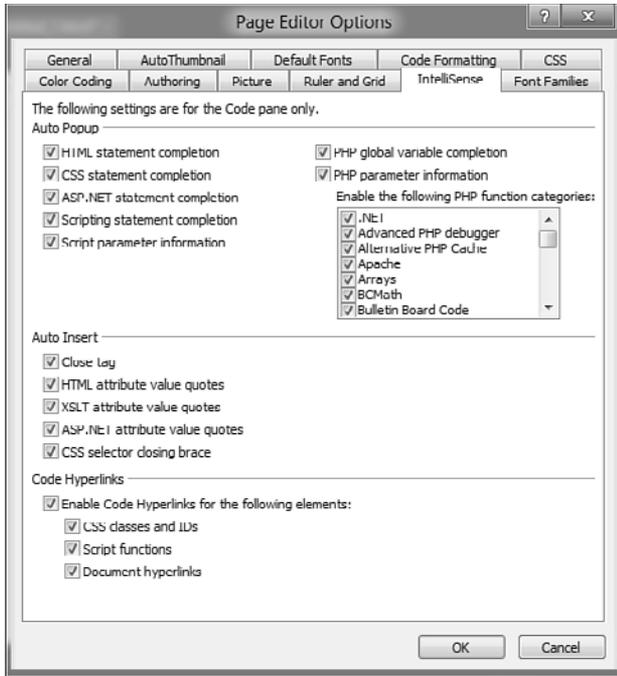


Figure 32.15
Set PHP IntelliSense options in the Page Editor Options dialog box.

PHP Script Options

To further simplify and assist you with your PHP coding, Expression Web lets you insert common PHP script blocks into your code from the Insert menu (see Figure 32.16). This can be helpful if you aren't familiar with the exact syntax of a particular PHP statement or global variable.

The following section describes the PHP script items you can insert using this method.

Form Variable

The Form variable holds information passed to the page from form elements on the requesting page. Form variables are passed using the HTTP POST method; therefore, the global variable name is `$_POST`.

The following code is inserted into a page when you select this option:

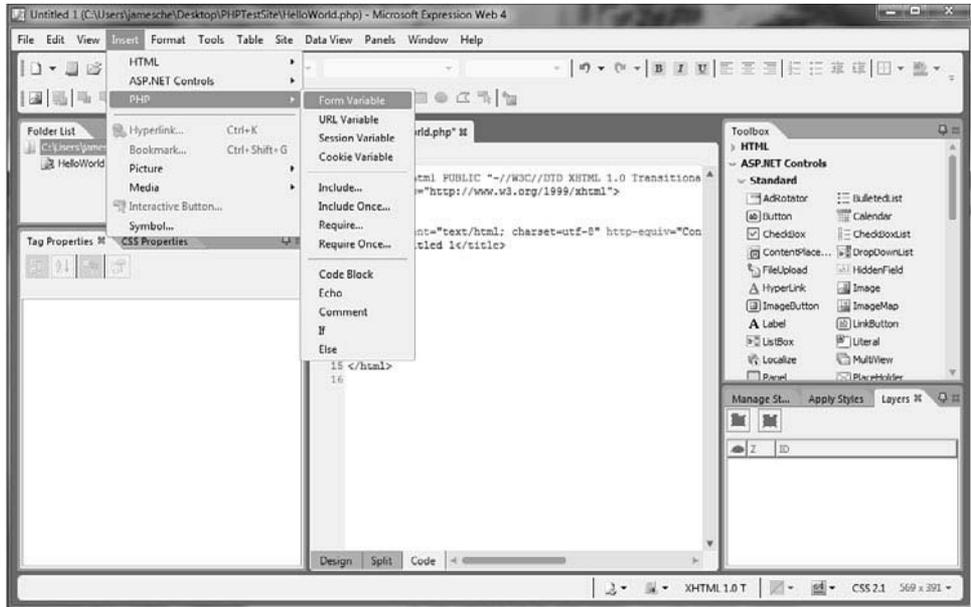
```
<?php $_POST[ ]; ?>
```



tip

It should be mentioned that Expression Web isn't very smart when it comes to inserting script text. If your cursor is already positioned inside a PHP script block, it will still add the PHP script tags around the inserted script. You must remember to delete these extra tags manually.

Figure 32.16
Add segments of PHP script into a page from the Insert menu.



To reference a specific form variable, you must use the name of the form element as a key, as shown in this example:

```
<?php $_POST["LAST_NAME"]; ?>
```

URL Variable

The URL Variable item inserts a script to pull items passed to the page via the HTTP GET method as parameters passed in the calling URL. The following script is inserted:

```
<?php $_GET[]; ?>
```

As with other global variables, you must use the name of a URL variable as a key to reference a specific variable value.

Session Variable

As a user visits pages on your site, the browser passes a session value that the web server uses to identify the specific user session. Session variables can be used to maintain user state information on the server between page requests. Use the Session Variable script item to insert the following script:

```
<?php $_SESSION[]; ?>
```

Instead of having session values determined by the requesting page, you must store values in the session variable yourself. Store it using a key you create:

```
<?php $_SESSION["role"] = "administrator"; ?>
```

Then, on a subsequent page request, retrieve it using the same key:

```
<?php
if ($_SESSION["role"] == "administrator") {
    echo "administrator is logged in.\n";
}
?>
```

Session variables last only as long as the user's session is active.

Cookie Variable

Cookie variables are similar to session variables in that they store information that persists from page to page. Cookies, however, are sent to the browser and stored on the client computer instead of the server. This item inserts the following script:

```
<?php $_COOKIE[]; ?>
```

include

PHP code that will be used in many pages can be separated out and put in its own file, which then can be included in a PHP page using the `include` statement.

When you select the Include item from the Insert menu, you are prompted to select a PHP file to include, as shown in Figure 32.17. This item inserts the following script, with the chosen page as a parameter to the `include` statement:

```
<?php include('common.php'); ?>
```

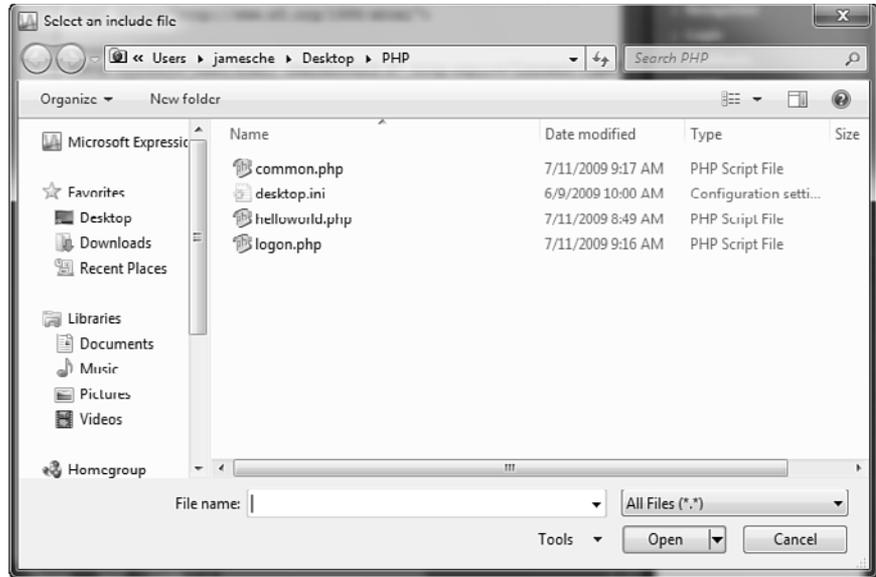
Include Once

The Include Once script is similar to the Include script, except that the inserted script will be included in a page only once to avoid variable and function redefinitions, value reassignments, and other potential problems. When you select this item, the following script is inserted:

```
<?php include_once('common.php'); ?>
```

As with Include, you select the file to add to the statement.

Figure 32.17
Include files let you
keep shared PHP
code in one place.



require

The PHP `require` statement is similar to the `include` statement, except in the way in which errors are handled when it is used. If the file referenced by an `include` statement doesn't exist, an error message will be displayed but script execution will continue. When the `require` statement is used, PHP script execution will stop if there is an error with the included file.

The following script is inserted when this statement is used:

```
<?php require('common.php'); ?>
```

require_once

The `require_once` statement is analogous to the `include_once` statement, but with the same difference in error handling as between `include` and `require`. This script segment is inserted:

```
<?php require_once('common.php'); ?>
```

Code Block

The Code Block item inserts a simple empty PHP script block tag, as follows:

```
<?php ?>
```

This can be useful as shorthand for entering the tag quickly, particularly if you memorize the keyboard shortcut used to enter it: Alt+I, H, B.

echo

As mentioned at the beginning of this chapter, the `echo` statement is a common way to have the PHP script insert dynamic text into a page. This item inserts the following script:

```
<?php echo ?>
```

The cursor is positioned after the `echo` statement to facilitate adding the string or variable values to be echoed.

Comment

The Comment item inserts a PHP C-style comment into the program flow, with the cursor positioned to type the comment text:

```
/* */
```

Note that this script doesn't include the surrounding script block tags. PHP assumes that the comment will be inserted into an existing block.

if

The `if` statement is a commonly used conditional execution statement. This item inserts the following script:

```
<?php if ?>
```

An expression to evaluate is then inserted enclosed in parentheses. The following is an example:

```
<?php if ($user_role == "administrator") ?>
```

An `if` statement typically is followed by a block of code to be executed, contained in braces. The code between the braces is executed only if the expression evaluated by the `if` statement is true. Note that this script does not add the braces; they must be added by hand.

The `if` statement can also use the alternative style described earlier in this chapter, in which case you must manually enter a colon after the expression to be evaluated, as well as a corresponding `endif;` statement. This can be useful when you want to embed HTML code in the `if` block, as the example in the "PHP Program Flow" section of this chapter shows.

else

The `else` statement can be used to follow up an `if` statement as an alternative block of code to execute if the expression evaluated by the `if` statement is false. This item inserts the following script:

```
<?php else ?>
```

The `else` statement should be followed up with a block of code in curly braces. You can also use the alternative syntax, although you need to remember to add a colon after the `else` statement. An example shown earlier in the chapter in the use of the `if...else...endif` syntax is repeated here to demonstrate this:

```
<php? if (%user_role == "administrator") : ?>
  <br />
  <span>administrator login successful</span>
  <br />
<php? else : ?>
  <br />
  <span>user login successful</span>
  <br />
<php? endif; ?>
```

Displaying PHP Information

Although you might know your server intimately, your site's PHP pages could end up deployed to some other server that might not be configured quite the same way as the server on which you developed it. This can often be frustrating to debug, as you end up putting in debugging `echo` statements that loop over global variables and other such techniques to try to glean some information about the server.

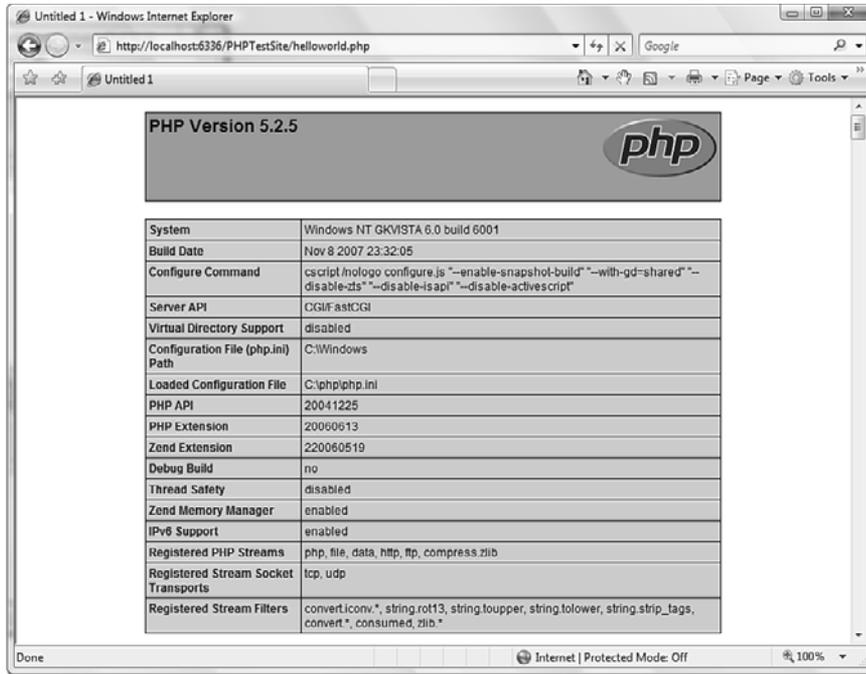
The need to display server information for development and debugging purposes is so common that the `phpinfo()` function was created just to facilitate the display of this type of information. If you are deploying a PHP application to many server configurations, it is in your best interest to include a test page somewhere that calls `phpinfo`.

To use `phpinfo`, simply include it in a PHP page in a PHP script block:

```
<?php
phpinfo();
?>
```

Various bits of server configuration information are displayed in the resulting page, as shown in Figure 32.18.

If you call `phpinfo` with no parameters, as previously shown, you will get all the information that can be displayed—which can be a lot. Give it a try and scroll down the list just to see what is available. If you want to restrict the output to only certain portions of the information, pass in a value as a parameter to restrict the output. The values that can be passed are defined as constants for convenience and appear in Table 32.2.

**Figure 32.18**

The `phpinfo()` function displays useful information about your PHP and server environment.

Table 32.2 Options for `phpinfo`

Option	Value	Description
<code>INFO_GENERAL</code>	1	General server and PHP information
<code>INFO_CREDITS</code>	2	Credits for the PHP developers
<code>INFO_CONFIGURATION</code>	4	Current value of configuration options from <code>php.ini</code>
<code>INFO_MODULES</code>	8	Loaded PHP modules
<code>INFO_ENVIRONMENT</code>	16	All environment variables (<code>\$_ENV</code>)
<code>INFO_VARIABLES</code>	32	All <code>\$_GET</code> , <code>\$_POST</code> , <code>\$_COOKIE</code> , and <code>\$_SERVER</code> variables
<code>INFO_LICENSE</code>	64	PHP license information
<code>INFO_ALL</code>	-1	All information (default value)

Because the arguments for `phpinfo` are bitwise values, they can be combined using the `or` operator (`|`) to display more than one item at a time. For example, you can display `INFO_ENVIRONMENT` and `INFO_VARIABLES` information by calling `phpinfo` as follows:

```
<?php
phpinfo(INFO_ENVIRONMENT | INFO_VARIABLES);
?>
```

USING THE MICROSOFT EXPRESSION DEVELOPMENT SERVER

Introduction to the Microsoft Expression Development Server

As sites have evolved, static sites have largely been replaced by dynamic ones that use a server-side technology such as ASP.NET or PHP to provide a more powerful and dynamic user experience.

The following are some of the things you can do using ASP.NET or PHP:

- Display data from a database on a page
- Allow people to create accounts and log in to your site
- Add robust form validation to your Web Forms
- Add dynamic site navigation
- Take advantage of powerful Ajax features for a better user experience

The likelihood of so many people delving into server-side technologies presents problems because server-side technologies require a web server to run. Although users of Windows XP Professional and many editions of Windows Vista and Windows 7 (and users of any Windows Server edition)



note

The Microsoft Expression Development Server also works fine on operating systems that ship with a web server.

have the benefit of a web server included with the operating system, the following Windows editions do not ship with a web server:

- Windows Vista and Windows 7 Starter
- Windows Vista and Windows 7 Home Basic
- Windows Vista and Windows 7 Home Premium
- Windows XP Home Edition

If you're using one of these operating systems, the Microsoft Expression Development Server enables you to develop and test ASP.NET and PHP applications in Expression Web.

Another problem that widespread use of server-side technologies can introduce is more likely to occur in educational environments. Developing with ASP.NET and PHP has traditionally required a user to have administrative access to the web server, but many educational institutions don't allow users to be administrators. Therefore, developers might not be able to debug applications—and in the worst-case scenarios, they might not be able to run ASP.NET or PHP applications at all.

Both of these problems are nicely solved with the Microsoft Expression Development Server provided with Expression Web. The Microsoft Expression Development Server fully supports ASP.NET and PHP, and it enables the development of ASP.NET and PHP applications on all current Microsoft operating systems and by non-administrators as well.

 **note**

Microsoft recently released a new development tool for web developers called WebMatrix. WebMatrix includes a version of IIS called IIS Developer Express, which can be used to host ASP.NET and PHP applications. You can use IIS Developer Express to browse your Expression Web sites, but there isn't any way to integrate IIS Developer Express into Expression Web.

For more information on WebMatrix and IIS Developer Express, see <http://www.microsoft.com/web/webmatrix/>.

 **note**

To use PHP, you need to install PHP on your computer. You can download PHP from www.php.net.

What Is the .NET Framework?

Ask 10 people what the .NET Framework is and you're likely to get 10 different answers. Most people have a limited understanding of exactly what the .NET Framework is, and many people have no understanding of it at all. That's partly due to Microsoft's marketing techniques.

A few years back, Microsoft started the .NET revolution. It seemed like almost everything Microsoft produced was branded with ".NET." The name even made it into Microsoft's messaging client when they began referring to Windows Messenger as ".NET Messenger." Naturally, customers were confused about just what .NET really meant. Now that the hype around .NET has faded for the most part, we're left with the real meat of what .NET is all about—the .NET Framework. (Incidentally, this whole process has begun again with the Microsoft Live platform, and .NET Messenger is now called Windows Live Messenger.)

Continued...

In a nutshell, the .NET Framework is a framework that developers can use to create applications that run on a common Microsoft platform called the Common Language Runtime. One of the primary benefits of the Common Language Runtime is that it provides developers with a common way of doing complicated tasks. Jobs that used to require a significant amount of code can now be completed with just a few lines of code. Expression Web's user interface is built on a technology called Windows Presentation Foundation (WPF) which is part of the .NET Framework. ASP.NET is a subset of the .NET Framework. When you develop pages that contain ASP.NET controls, you are developing on the Microsoft .NET Framework. Expression Web uses ASP.NET 4.

How to Use the Microsoft Expression Development Server

When you create a site in Expression Web, you have many options from which to choose, but the easiest to work with is a disk-based site. However, because a disk-based site doesn't actually reside on a web server, it doesn't sit on top of a platform that supports ASP.NET or PHP. That's where the Microsoft Expression Development Server comes into the picture. The Microsoft Expression Development Server is used by default for all ASP.NET and PHP pages browsed from a disk-based site.



note

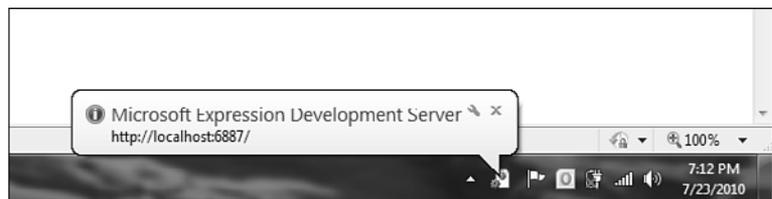
The Microsoft Expression Development Server is supported only on disk-based sites.

➔ *For more information on creating sites in Expression Web, see Chapter 2, “Creating, Opening, and Importing Sites.”*

Expression Web starts the Microsoft Expression Development Server on a random port when an ASP.NET or PHP page is previewed in your browser (see Figure 33.1).

Figure 33.1

The Microsoft Expression Development Server starts automatically when you preview an ASP.NET or PHP page from a disk-based site.



tip

Port numbers are used whenever communication takes place over the Internet. By default, web servers use port 80 for HTTP traffic. Other default ports exist for other services.

The Microsoft Expression Development Server chooses a random port in a port range that is not used by common services. This prevents the port the Microsoft Expression Development Server uses from conflicting with something else running on your computer.

You can also use the Microsoft Expression Development Server for all pages. To configure whether the Microsoft Expression Development Server is used only for ASP.NET and PHP pages or for all content, click Site, Site Settings; then click the Preview tab. As shown in Figure 33.2, two options are available for previewing with the Microsoft Expression Development Server:

- **For Only PHP and ASP.NET Web Pages**—When this option is selected, Expression Web launches the Microsoft Expression Development Server when .aspx files are previewed. If non-ASP.NET pages are previewed, Expression Web uses the path of the site that is open.
- **For All Web Pages**—When this option is selected, Expression Web uses the Microsoft Expression Development Server for all files that are previewed.

 **note**

If the Microsoft Expression Development Server is running when you exit Expression Web, it will be stopped automatically.



Figure 33.2

You configure the use of the Microsoft Expression Development Server through the Site Settings dialog.



Dynamic Content Warning with Microsoft Expression Development Server

You might have your site configured to use the Microsoft Expression Development Server, but when you browse an ASP.NET page, you get the warning that the page contains dynamic content and the page is still served from the disk path. How can you correct this problem?

In my testing, there were several occasions when I had the Microsoft Expression Development Server configured to launch, but for some reason Expression Web would launch pages from the file path on the first attempt. When that happens, simply close the browser and preview the page again and it should work correctly.

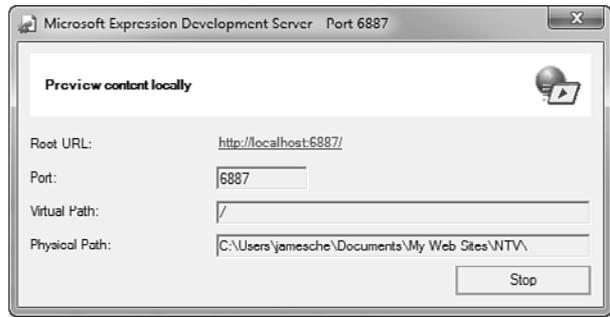
The Microsoft Expression Development Server remains running for as long as Expression Web is running. However, you can stop it. The following are reasons you might want to manually stop the Microsoft Expression Development Server:

- You have an ASP.NET or PHP page that caches information and want to clear the cache so you can test a particular scenario.
- You are testing code in your application that runs when your application first loads.

To manually stop the Microsoft Expression Development Server, right-click the icon in the system tray shown previously in Figure 33.1 and select Stop from the menu. Alternatively, select Show Details from the menu and click the Stop button (see Figure 33.3). Either way, the Microsoft Expression Development Server will stop immediately and will be launched again on a new port the next time it's needed.

Figure 33.3

You can stop the Microsoft Expression Development Server from the Details dialog box. You also can click the provided link to browse the site in your web browser.



Limitations of the Microsoft Expression Development Server

The Microsoft Expression Development Server solves a lot of problems. In addition to providing a solid development platform for ASP.NET and PHP for those using versions of Windows without a web server, it also allows non-administrators the option of debugging ASP.NET sites locally. However, even with those benefits, it has some limitations.

Process Identity

On some occasions in an application, you might need to run code under a specific user's identity. For example, suppose you have an ASP.NET application that makes heavy use of a Microsoft SQL Server database and you want that database to be accessed via a particular user account. In such a scenario, being able to configure ASP.NET to always run under that particular user account is a valuable feature.

The Microsoft Expression Development Server always runs under the user who is logged on to the computer. You cannot configure it to run under a different user account. Therefore, you might not be able to test all scenarios that your application might require.

No Remote Access

The Microsoft Expression Development Server is designed to be browsed from the local machine only. That means you can browse it only using `localhost` or the TCP/IP loopback address: `127.0.0.1`. You cannot browse to the Microsoft Expression Development Server using either your machine name or your computer's IP address.

Because only `localhost` and the loopback address are acceptable to the Microsoft Expression Development Server, you also cannot browse an ASP.NET or PHP site running on the Microsoft Expression Development Server from a remote machine. If you need that functionality, you will need to use another web server (such as IIS 7) to develop your application.

No Support for ASP Pages

Even though you can configure the Microsoft Expression Development Server to serve all pages, as shown previously in Figure 33.2, it is explicitly designed to not allow ASP pages. If you attempt to browse to an ASP page in the Microsoft Expression Development Server, you get an error telling you that ASP pages are not served (see Figure 33.4).



note

If you are using an IPv6 address, the loopback address will be `::1`.



tip

Even though you can't browse the Microsoft Expression Development Server remotely, you can use the Microsoft Expression Development Server to develop against content that's located on a different machine. You'll need to map a drive to the remote content and then start the Microsoft Expression Development Server from a command line using a switch to point it to the content.

For more information on starting the Microsoft Expression Development Server from a command prompt, see "Starting the Microsoft Expression Development Server from the Command Prompt," p. 585.

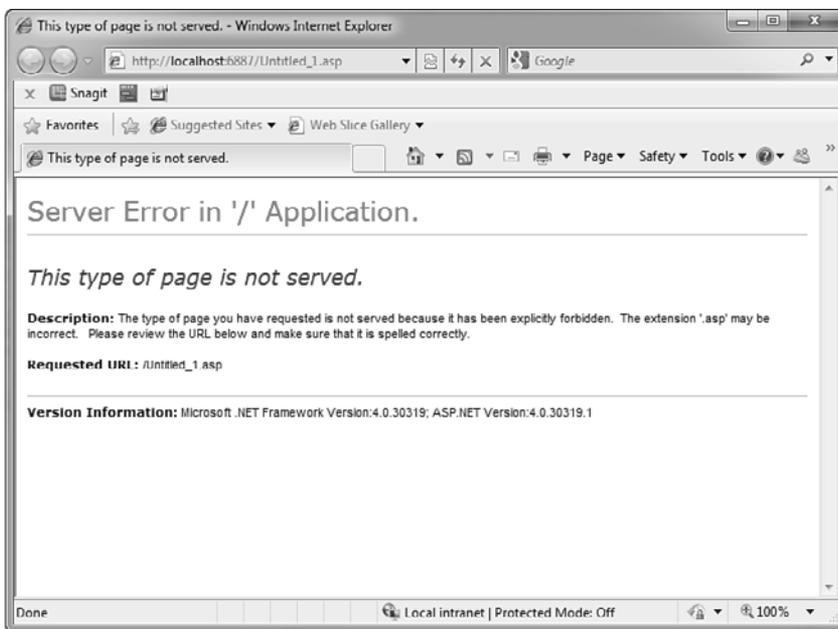


Figure 33.4
The Microsoft Expression Development Server does not serve ASP pages. Only ASP.NET and HTML pages are supported.

 **note**

You might wonder why the Microsoft Expression Development Server doesn't support ASP pages. There really isn't any technical reason; it was simply a design choice Microsoft made, most likely because the Microsoft Expression Development Server was first developed as a web server specifically for ASP.NET and PHP development tools.

Starting the Microsoft Expression Development Server from the Command Prompt

In most scenarios, launching the Microsoft Expression Development Server from within Expression Web will suffice. However, if you want to author against remote content via a mapped drive, you need to take a different approach and launch the Microsoft Expression Development Server from a command prompt.

To launch the Microsoft Expression Development Server from a command prompt, navigate to the C:\Program Files\Microsoft Expression\Web 3 directory, and then run the following command:

```
expression.devserver /port:<port> /path:<path>
```

You need to replace <port> with the port number of your choice and <path> with the path to the mapped drive. For example, to launch the Microsoft Expression Development Server on port 9000 and browse content located at z:\website, you would use the following:

```
expression.devserver /port:9000 /path:z:\website
```

You then are able to browse your content on the port specified.

Adding on to the preceding example, to browse a page called form.aspx, enter the following into your web browser:

```
http://localhost:9000/form.aspx
```

This page intentionally left blank

DISPLAYING AND EDITING DATABASE DATA WITH ASP.NET

A History of Data Access

Over the past several years, sites have evolved into complex web applications. As sites evolved, the consumers of those sites evolved as well. We expect more from our sites, and a large part of meeting that expectation is providing access to dynamic data.

Microsoft first introduced data access to mainstream web developers in FrontPage 97 with the inclusion of IDC/HTX. Although IDC/HTX allowed web developers to create data-driven pages, it was a far cry from the ease of use necessary to really push data-driven sites into the mainstream. However, when Active Server Pages (ASP) technology was released very soon thereafter, developers finally had a way to build dynamic content quickly.

Many years later, Microsoft unveiled the .NET Framework, which included a new server-side technology called ASP.NET. ASP.NET once again revolutionized site development because it finally gave web designers the toolset they needed to develop web applications that felt like Windows applications. In fact, many of today's ASP.NET developers were developers of Windows-based applications just a few years ago.

➔ *For more information on the .NET Framework and ASP.NET, see Chapter 25, “Using Standard ASP.NET Controls.”*



note

ASP.NET used to be referred to by Microsoft as ASP+. In fact, the file extension for ASP.NET Web Forms is .aspx—the x is simply a + turned on its side.

Data Access Technologies in Expression Web

Because Expression Web uses ASP.NET for data access, many powerful data access tools such as the DataView, the GridView, and other tools are available to you. Using these tools, you can connect to data in a database, in an Extensible Markup Language (XML) file, or even in a sitemap file.

➔ *For more information on sitemap files in ASP.NET, see Chapter 26, “Using ASP.NET Navigation Controls.”*

I realize that many web designers are not programmers. In fact, Expression Web is aimed squarely at the designer market and not the programmers among us. However, don't let that dissuade you from digging into the data access features in Expression Web. You certainly don't need to be a coder to appreciate and use the features that ASP.NET provides for Expression Web.

Additionally, some ASP.NET developers are more experienced in developing with ASP.NET, and yet they believe it is proprietary and that they are able to connect only to Microsoft database technologies. In fact, although you can certainly use the features in ASP.NET to connect to SQL Server and Microsoft Access, you can also use the same tools to connect to Oracle databases, IBM DB2 databases, MySQL databases, and so on. You can connect to all these data sources using what Microsoft calls the provider model.

ASP.NET comes with several providers that allow you to connect to the databases mentioned previously. However, anyone can write a provider and plug it into ASP.NET to enable extended functionality. For example, if you wanted to write a provider specific to MySQL, one that added some functionality geared toward MySQL databases, you certainly could. You could then use your provider along with all the existing data access functionality in ASP.NET.

ASP.NET and Other Web Application Platforms

The one drawback to using ASP.NET for data access is that not all hosting companies offer hosting with ASP.NET support. In fact, many of the less expensive hosting companies host sites on Linux or UNIX, and those servers won't work with ASP.NET.

Many of today's web designers are drawn to technologies such as PHP because they think PHP is easier to use than ASP.NET. Still others believe that you have to buy the .NET Framework to use ASP.NET. It is my assertion that both of these positions are flat-out wrong. It is every bit as easy to



note

A sitemap file is a special XML file used by ASP.NET to map out the navigation structure of a site.



note

If you find that you enjoy building ASP.NET pages with the features available in Expression Web, you can easily move to the next level by downloading Visual Web Developer 2010 Express free from Microsoft.



note

You don't need to understand providers to use them. In fact, we'll use some powerful data access features in the next chapter without going into providers at all.

An in-depth discussion of providers is outside the scope of this book, but if you'd like to dig into the topic and work with a sample provider, Microsoft offers one for download at <http://msdn2.microsoft.com/en-us/library/26xsd945.aspx>.

use ASP.NET as it is to use PHP. In fact, it is actually easier to create robust web applications with ASP.NET in Expression Web because of the limited PHP support.

As for those who believe that being an ASP.NET developer is an expensive endeavor, let me say that the .NET Framework is a free download (as is the .NET Framework software development kit [SDK]), and Microsoft provides a feature-rich, professional development environment for creating ASP.NET applications called Visual Web Developer 2010 Express, which costs nothing. If you haven't yet downloaded a copy of Visual Web Developer 2010 Express, you should stop reading right now and do it. You'll be absolutely amazed that Microsoft is giving away this tool! You can download a copy at <http://www.microsoft.com/express/Web/>.

Let's explore how you can take advantage of the data access and ASP.NET features in Expression Web to display and edit database data.

ASP.NET Data Source Controls

The previous chapters on ASP.NET have shown how you can design some pretty impressive ASP.NET pages without writing any code. It might surprise you to know that you also can create some powerful data access pages in ASP.NET without writing any code. The functionality for doing so is encapsulated within the ASP.NET data source controls.

ASP.NET data source controls provide robust connectivity to many types of data sources simply by setting properties on a control using the Tag Properties panel or by specifying properties for a control in Code View declaratively.

➔ *For more information on setting ASP.NET control properties using the Tag Properties panel, see Chapter 25, "Using Standard ASP.NET Controls."*

Four data source controls are included in the Expression Web Toolbox. Each data source control is similar in functionality, but they are specialized for specific types of data.

Let's go over the details of each data source control, and then we'll create some pages that use the `AccessDataSource` and `SqlDataSource` controls to connect to a database.

AccessDataSource Control

The `AccessDataSource` control provides data connectivity to a Microsoft Access database file. It includes some advanced features such as support for paging, sorting, filtering, and more.

To insert an `AccessDataSource` control onto a page, drag it from the Toolbox and drop it on the page. By default, Expression Web is configured to not show nonvisual ASP.NET pages, so when you



note

ASP.NET data source controls are located in the Data section of the ASP.NET Controls Toolbox.



note

Data source controls do not display data on a page. As you'll see later in this chapter, they are used in conjunction with other controls to display data.



note

Throughout this chapter, we will be using the `Nwind.mdb` database to build examples. If you don't already have this database on your system, you can download it from www.microsoft.com/downloads/details.aspx?familyid=C6661372-8DBE-422B-8676-C632D66C529C&displaylang=en. It's a fairly old database, but it's a great way to build sample database access pages.

first insert any data source control, Expression Web asks whether you want to turn on the visual aid for ASP.NET controls. It's recommended that you answer Yes to this prompt so you can work more easily with the data source control.

➔ *For more information on visual aids, see Chapter 4, "Using Page Views."*

After you've inserted the `AccessDataSource` control, the easiest way to configure it is to click the `Configure Data Source` link in the `AccessDataSource` Tasks pop-up, as shown in Figure 34.1.

note

The `AccessDataSource` control cannot connect to a password-protected Access database. To connect to a password-protected database, use the `SqlDataSource` control.

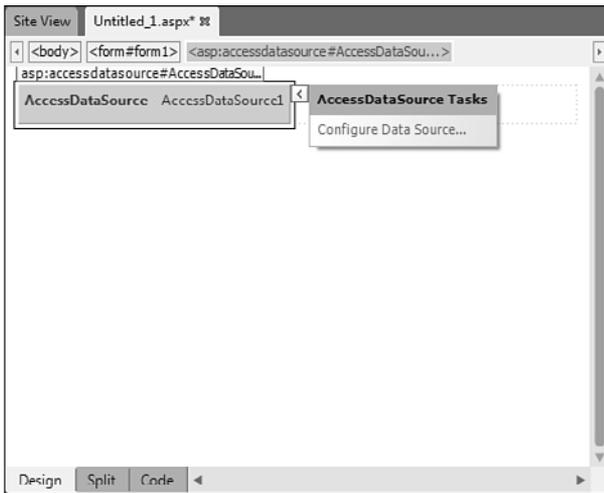


Figure 34.1

Configuring a data source is easily done using the `Configure Data Source` link in the `AccessDataSource` Tasks pop-up.



Unable to See Data Source Control

If you've added a data source control to your page and are trying to configure it, but can't see it in Design View, you haven't enabled the ASP.NET nonvisual controls visual aid. By default, nonvisual controls are not visible. To make them visible, select `View, Visual Aids, ASP.NET Non-visual Controls`.

We'll go over the details of configuring the `AccessDataSource` control and other data source controls later in this chapter. For now, let's review some of the common properties used to configure the `AccessDataSource` control:

- `DataFile`—Specifies the location of the Microsoft Access database file used by the `AccessDataSource` control.

- **DataSourceMode**—This property can be set to either `DataSet` (the default) or `DataReader`. In most cases, you leave this property set to `DataSet`, but if you only need to display data in the database without sorting, filtering, and so on, you can set it to `DataReader` for better performance.
- **SelectQuery**—Specifies the SQL query used to select data from the database. When the `SelectCommandType` is set to `StoredProcedure`, the `SelectQuery` property can be used to specify a stored query.
- **UpdateQuery**—Specifies the SQL query to use when updating the database.
- **DeleteQuery**—Specifies the SQL query to use when deleting records from the database.
- **FilterExpression**—Filters data displayed by the `SelectQuery` property.
- **FilterParameters**—Specifies the parameters used with `FilterExpression` to filter the database results.

 **note**

A discussion of creating SQL queries is outside the scope of this book. If you'd like more information on how to use SQL queries to work with databases, read *Sams Teach Yourself SQL in 24 Hours, 3rd Edition* from Sams Publishing, available at www.quepublishing.com/bookstore/product.asp?isbn=0672324423&rl=1.

SqlDataSource Control

The `SqlDataSource` control can be used to connect to many types of databases, but it contains performance optimizations that specifically target Microsoft SQL Server.

The `SqlDataSource` control shares many of the same properties with the `AccessDataSource` control. All the properties discussed previously for the `AccessDataSource` control also apply to the `SqlDataSource` control.

To insert a `SqlDataSource` control on a page, drag it from the Toolbox onto the page. After inserting the `SqlDataSource` control, you can configure it using the `Configure Data Source` link on the `SqlDataSource` Tasks pop-up. We'll cover the details of the `SqlDataSource` control later in this chapter.

 **note**

Unless you're working with an enterprise-level application, you might not notice the optimizations in the `SqlDataSource` control.

SiteMapDataSource Control

The `SiteMapDataSource` control is a hierarchical data source control designed to work with the ASP.NET navigation controls. ASP.NET navigation controls are covered in detail in Chapter 26, "Using ASP.NET Navigation Controls," so we won't go into detail on this control here.

XmlDataSource Control

The `XmlDataSource` control provides data connectivity to XML files. To insert an `XmlDataSource` control, drag it from the Toolbox onto the page. You can then configure it using the Configure Data Source link on the `XmlDataSource` Tasks pop-up.

When configuring the `XmlDataSource` control, specify the XML data path, an optional XML transform file, and an optional XPath statement for filtering data, as shown in Figure 34.2.

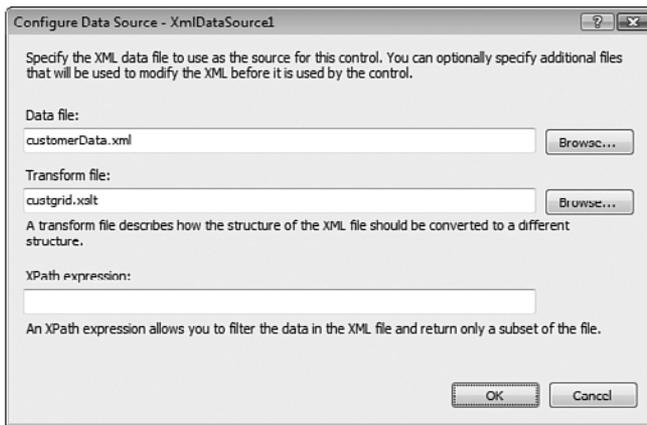


Figure 34.2

For greater flexibility, an XML transform and XPath expression can be configured for an `XmlDataSource` control.

Displaying Data with ASP.NET

There are several approaches to displaying data with ASP.NET. Which one you choose depends largely on the type of data you need to display and your purpose for displaying it. For example, to display a list of products and prices, a tabular display might be the best choice. On the other hand, to display information such as the name and address of one particular user, a data-bound form would be a better choice.

ASP.NET makes it extremely easy to display data using both methods, and you can do it without writing any code because the work is all done by the data source control.

Displaying Data in Tabular Form

A few ASP.NET controls specialize in displaying data in tabular form. The two most common controls are the `DataList` control and the `GridView` control. Of the two, the `GridView` is the easiest to use and offers the largest feature set.

Let's create a new ASP.NET page to display data from the Northwind Traders Access database. You can download the sample `Nwind.mdb` file using the link provided earlier in this chapter if you don't already have the Northwind Traders database.

note

A discussion of using XML transforms and XPath is outside the scope of this book. For details on using these XML features, read *Sams Teach Yourself XML in 10 Minutes* from Sams Publishing, available from www.quepublishing.com/bookstore/product.asp?isbn=0672324717&rl=1.

Adding and Configuring a GridView Control

We need to create a new ASP.NET page so we can add and configure a GridView control to display our data. Here's how:

1. Open an existing site or create a new one.

➔ *For more information on creating a site, see Chapter 2, "Creating, Opening, and Importing Sites."*

➔ *For more information on the Microsoft Expression Development Server, see Chapter 33, "Using the Microsoft Expression Development Server."*

2. Create a new ASP.NET page and save it as `gridview.aspx`.

➔ *For more information on creating ASP.NET pages, see Chapter 3, "Creating Pages and Basic Page Editing."*

3. Drag a GridView control from the ASP.NET section of the Toolbox and drop it on the page.



note

The site can be either server-based or disk-based. If you use a disk-based site, the Microsoft Expression Development Server can be used to test the site.



Toolbox Not Visible

If you'd like to insert an ASP.NET control but the Toolbox is not visible, select Task Panes, Toolbox to toggle on the display of the Toolbox.

4. Import the `Nwind.mdb` file into the root of the site.

➔ *For more information on importing files, see Chapter 3, "Creating Pages and Basic Page Editing."*

5. If the GridView Tasks pop-up is not visible, click the arrow button to display it.

6. Select `<New Data Source...>` from the Choose Data Source drop-down.

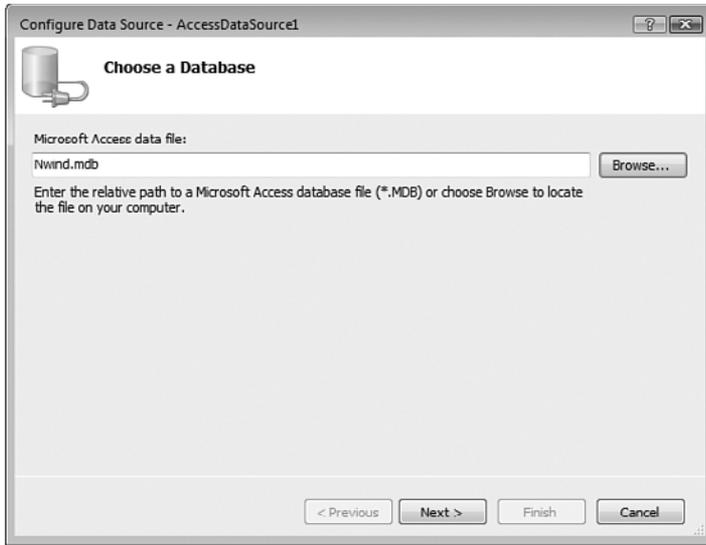
7. Select Access Database in the Data Source Configuration wizard and click OK.

8. Type **Nwind.mdb** in the Configure Data Source dialog, as shown in Figure 34.3; then click Next.

9. Be sure Specify Columns from a Table or View is selected.

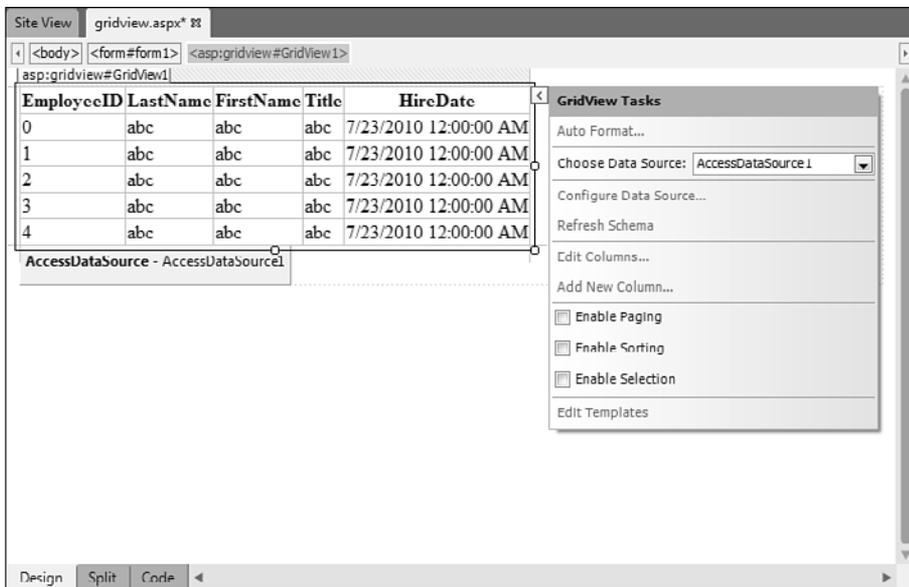
10. From the Name drop-down, select Employees.

11. Check the EmployeeID, LastName, FirstName, Title, and HireDate check boxes; then click Next.

**Figure 34.3**

When displaying data from an Access database, you need to specify the location and filename of the database.

- Click the Test Query button to ensure that you are successfully connecting to the database; then click Finish. After you click Finish, you will see that the GridView control now displays the columns you selected. Several more options are available in the GridView Tasks pop-up, as shown in Figure 34.4.

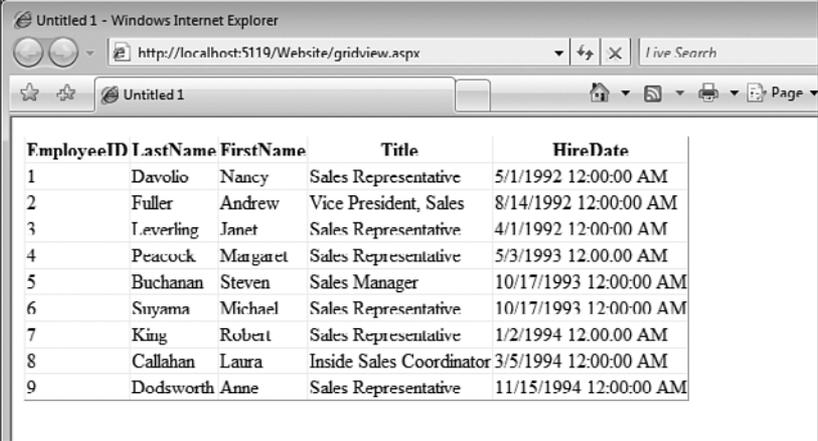
**Figure 34.4**

After you've configured a data source, you have other options available for the GridView. In this case, you can now enable paging, sorting, and selection.

Save the page and test it in your browser. Even though you haven't written any code, you will see that the GridView control is pulling the employee data from the database and displaying it on the page, as shown in Figure 34.5.

Figure 34.5

You have successfully pulled data from the Access database, and you did it without writing a single line of code.



EmployeeID	LastName	FirstName	Title	HireDate
1	Davolio	Nancy	Sales Representative	5/1/1992 12:00:00 AM
2	Fuller	Andrew	Vice President, Sales	8/14/1992 12:00:00 AM
3	Leverling	Janet	Sales Representative	4/1/1992 12:00:00 AM
4	Peacock	Margaret	Sales Representative	5/3/1993 12:00:00 AM
5	Buchanan	Steven	Sales Manager	10/17/1993 12:00:00 AM
6	Snyama	Michael	Sales Representative	10/17/1993 12:00:00 AM
7	King	Robert	Sales Representative	1/2/1994 12:00:00 AM
8	Callahan	Laura	Inside Sales Coordinator	3/5/1994 12:00:00 AM
9	Dodsworth	Anne	Sales Representative	11/15/1994 12:00:00 AM

Sorting the GridView

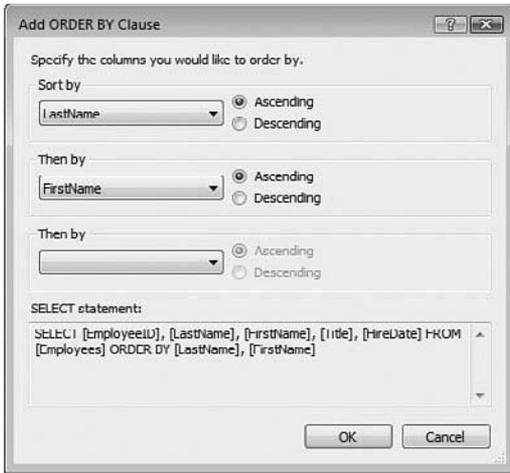
Right now the employees are in employee number order. Let's reconfigure the display so they are listed in order of last name and then first name:

1. Select the GridView and click the arrow button to display the GridView Tasks pop-up.
2. Click the Configure Data Source link.
3. In the Configure Data Source dialog, click Next.
4. Click the Order By button.
5. In the Sort By drop-down, select LastName.
6. In the Then By drop-down, select FirstName, as shown in Figure 34.6.
7. Click OK.
8. Click Next and then click Finish in the Configure Data Source dialog.

Save and browse the page. Notice that the grid is now sorted by last name. If any two employees with the same last name are added to the database, the grid displays them sorted by first name.

In a real-world application, it's usually best to give the user a choice as to how to display the data. Let's reconfigure the GridView control so the user can decide how to sort it:

1. Select the GridView control.
2. Click the arrow button to display the GridView Tasks pop-up.

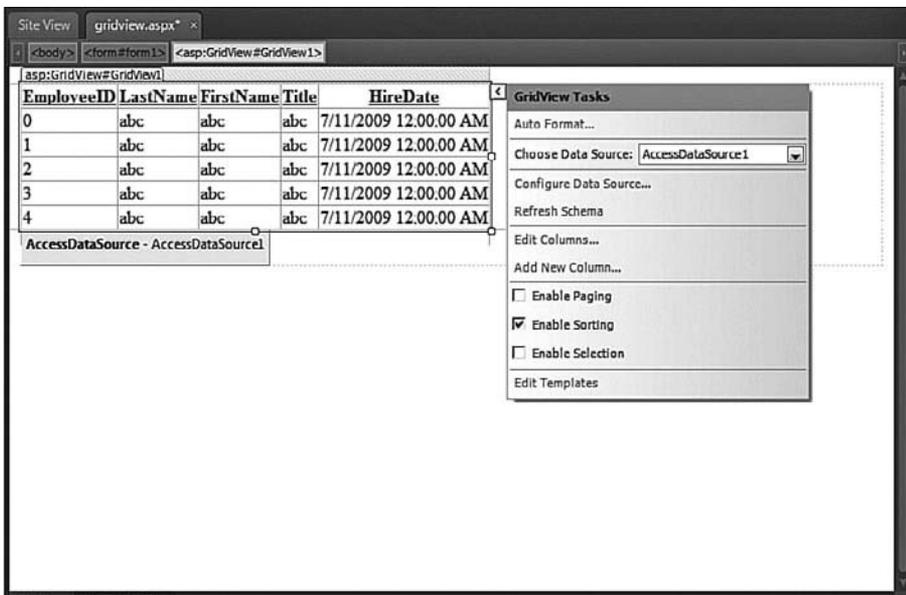
**Figure 34.6**

Configuring a data source so that the results are sorted is easy using the Add ORDER BY Clause dialog.

3. Check the Enable Sorting check box, as shown in Figure 34.7.

Save and view the page in your browser. Notice that the header on each column is now a hyperlink. Clicking a link sorts by that column in ascending order. Clicking the same link again sorts by that column in descending order.

In addition to binding data in a tabular fashion, you can bind data in forms and other controls in a page. We'll explore data binding in the "Lagniappe" section of this chapter.

**Figure 34.7**

You can give the user a choice for how to sort a GridView by checking the Enable Sorting check box.

Editing Data with ASP.NET

Let's create a page that allows us to edit employee information and save updated information back to the database. The `GridView` control makes this convenient by enabling you to switch a row into edit mode. When you switch a row into edit mode, the information in the row is presented in text boxes so you can make changes.

Open the `gridview.aspx` page and save it as `edit.aspx`.

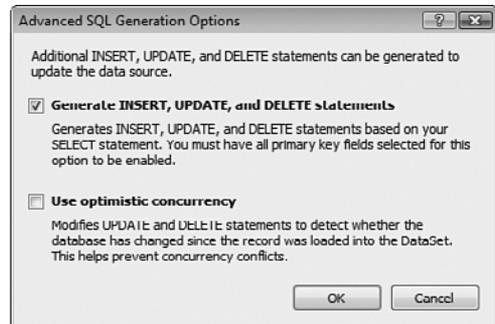
Configuring the Data Source

Now you need to reconfigure the data source to allow for updates and add a column to the `GridView` so users can switch a row into edit mode.

1. Select the `GridView` control and click the arrow button to display the `GridView Tasks` pop-up.
2. Click the `Configure Data Source` link to display the `Configure Data Source` dialog.
3. Click `Next`.
4. Click the `Advanced` button.
5. Check the `Generate INSERT, UPDATE, and DELETE Statements` check box (see Figure 34.8). This causes `Expression Web` to generate the necessary SQL statements to enable us to edit records from the database.

Figure 34.8

The `Advanced SQL Generation Options` dialog makes adding the code necessary to edit the database a one-step process.



6. Click `OK`, and then click `Next` and `Finish` to close the `Configure Data Source` dialog.

Configuring the GridView

Now that the data source is capable of updating information in the database, the final step is to configure the GridView and add a new column so a specific row can be switched into edit mode.

I'm going to walk you through a long way of doing this because it will give you some experience in using some of the features of the GridView that you wouldn't see otherwise. After you've configured the GridView, I'll tell you an easier way to accomplish the same task. Here's how:

1. Select the GridView and click the arrow button to display the GridView Tasks pop-up.
2. Click the Edit Columns link to display the Fields dialog.
3. Scroll down in the Available Fields list, and click the plus sign next to CommandField.
4. Select the Edit, Update, Cancel field. Then click Add to add it to the GridView.
5. Select the Edit, Update, Cancel field in the Selected Fields list, and click the up arrow button repeatedly to move it to the top of the list.
6. Click OK in the Fields dialog.

Your page should now look like Figure 34.9.

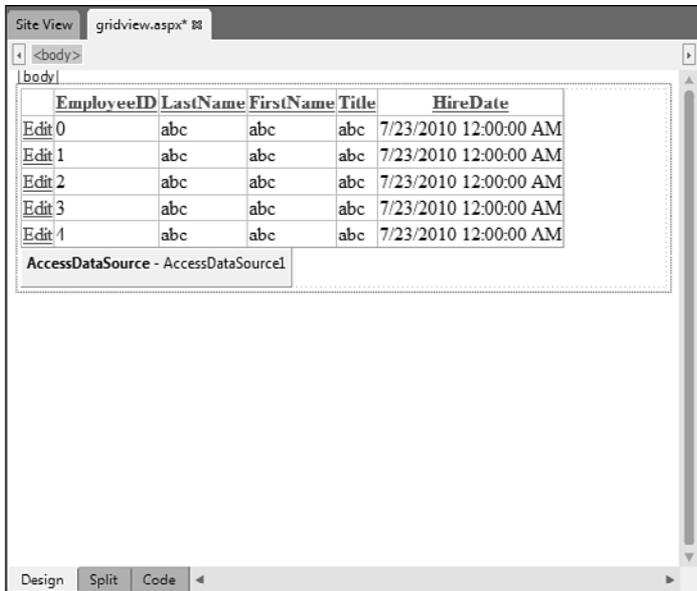
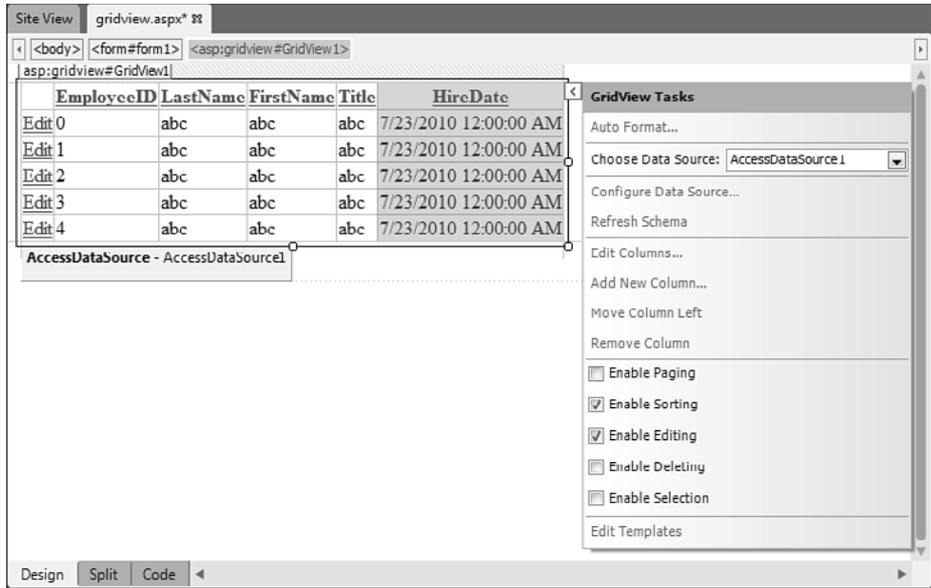


Figure 34.9

The GridView now has a new column so employees can be edited.

When I started this section, I said I would describe the easy way to configure a GridView for editing the database. After you've configured the data source so it will generate the code necessary to update the database, new check boxes will be added to the GridView Tasks pop-up, as shown in Figure 34.10.

Figure 34.10
The GridView Tasks pop-up offers different options based on how the data source is configured. You now have the option to enable editing by simply checking a box.



The easy way to configure the GridView so records can be edited is to simply check the Enable Editing check box. When you do that, Expression Web automatically adds the Edit, Update, Cancel column to the GridView.

Testing the Page

View the edit.aspx page in your browser. Click the Edit link next to the employee of your choice. When you do, the Edit link changes to an Update and Cancel link and all the data will be displayed in text boxes, as shown in Figure 34.11.

To update data in the database, enter the new information and click Update. Alternatively, click Cancel and revert to the original data from the database.

Using the methods you've learned here, you can easily create a robust web application that allows for not only the display of, but also the easy editing of, data from a database. The example used in this chapter used an Access database, but the same methods apply to other databases.

There's much more you can do with data access and ASP.NET. The easiest way to learn how to take advantage of this powerful feature set is to explore it and experiment with creating data-enabled pages.

	EmployeeID	LastName	FirstName	Title	HireDate
Edit	5	Buchanan	Steven	Sales Manager	10/17/1993 12:00:00 AM
Edit	8	Callahan	Laura	Inside Sales Coordinator	3/5/1994 12:00:00 AM
Edit	1	Davolio	Nancy	Sales Representative	5/1/1992 12:00:00 AM
Update Cancel	9	Dodsworth	Anne	Sales Representative	11/15/1994 12:00:00 AM
Edit	2	Fuller	Andrew	Vice President, Sales	8/14/1992 12:00:00 AM
Edit	7	King	Robert	Sales Representative	1/2/1994 12:00:00 AM
Edit	3	Leverling	Janet	Sales Representative	4/1/1992 12:00:00 AM
Edit	4	Peacock	Margaret	Sales Representative	5/3/1993 12:00:00 AM
Edit	6	Suyama	Michael	Sales Representative	10/17/1993 12:00:00 AM

Figure 34.11
The GridView is now a full-featured data editor.

Creating a Master/Detail View

In many cases, it's preferable to display a small subset of data and allow users to drill down into more information. This kind of approach is typically called a *master/detail view*, and creating such an interface is simple using the ASP.NET controls in Expression Web.

Creating the Master View

The master view contains a subset of the Products table in the Northwind Traders database. We'll use a GridView to display this data.

1. Create a new ASP.NET page and save it as `masterdetail.aspx`.
2. Add a GridView control to the page and select the option to add a new data source from the Choose Data Source drop-down in the GridView Tasks pop-up.
3. In the Data Source Configuration wizard, select Access Database and change the ID for the data source to `MasterDataSource`.
4. Select the `Nwind.mdb` database and click Next.
5. Select the Products database from the Name drop-down.

6. In the Columns list, select the ProductID, ProductName, and UnitPrice check boxes.
7. Click Next, and then click Finish.

You'll need to use the ProductID field to determine which record to display in the Detail View, but you don't want the user to see the ProductID field. Therefore, you need to make the ProductID field invisible to the reader. Here's how:

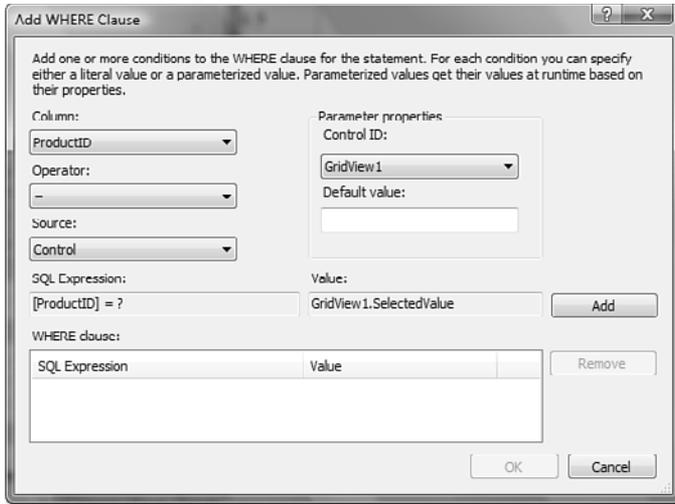
1. Select the GridView and click the arrow button to display the GridView Tasks pop-up.
2. Select the Edit Columns link and select the ProductID field in the Selected Fields list.
3. Change the Visible property to False.
4. Click OK to dismiss the Fields dialog.
5. Check the Enable Paging and Enable Selection check boxes in the GridView Tasks pop-up.

The next step is to create the detail view using a DetailsView control.

Creating the Detail View

Unlike the GridView you added previously, the DetailsView control displays all the fields for the selected record. Therefore, you need to insert a new data source control for the DetailsView control, and you need to configure the new data source to retrieve only the record you select in the GridView. Do the following:

1. Add a new AccessDataSource control to the page.
2. Click the Configure Data Source link in the AccessDataSource Tasks pop-up.
3. Click Browse and select the Nwind.mdb database in the first step of the Configure Data Source wizard. Click Next.
4. Select the Products table from the Name drop-down.
5. Place a check in the * check box, so that all fields are retrieved.
6. Click the WHERE button.
7. Select ProductID from the Column drop-down in the Add WHERE Clause dialog.
8. Select = in the Operator drop-down.
9. Select Control in the Source drop-down.
10. Select GridView1 in the Control ID drop-down. The Add WHERE Clause dialog should now look like the one shown in Figure 34.12.

**Figure 34.12**

Adding the WHERE clause for the data source control so that it retrieves only the record that is selected in the GridView.

11. Click the Add button to add the new WHERE clause and then click OK.
12. Click Next and then Finish to complete the Configure Data Source wizard.
13. Select the new `AccessDataSource` control, if it's not already selected; then change the ID property in the Tag Properties panel to `DetailsDataSource`.

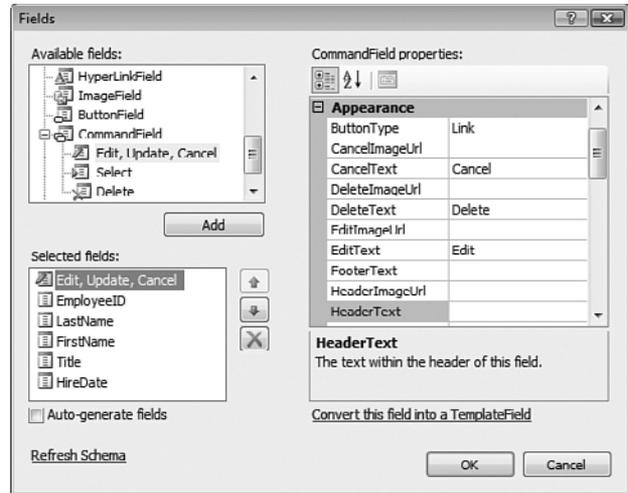
The changes you made in the Add WHERE Clause dialog cause the `ProductID` for the record you select in the `GridView` control to be passed to the query that retrieves the selected record from the database.

To complete the page, add a new `DetailsView` control under the existing `GridView` control. In the Choose Data Source drop-down on the `DetailsView` Tasks drop-down, select `DetailsDataSource`. If you want to improve the appearance of your page, drag the right side of the `DetailsView` control so that it's about 300 pixels wide; then save the page and preview it in your browser. When you click the Select link for one of the records in the `GridView` control, the details for that record are displayed in the `DetailsView` control as shown in Figure 34.13.

You can add more features to this page by using server-side code to control when the `DetailsView` control is displayed, but doing so is outside the scope of this book. However, hopefully you've learned enough about using the ASP.NET data controls available to you in Expression Web so you can implement powerful database functionality in your own sites.

Figure 34.13

The master/detail page is now fully functional, and we did it without writing any code.



This page intentionally left blank

SENDING EMAIL USING ASP.NET

A Typical Contact Form

I often hear from readers asking me how to create a form in Expression Web that collects information from a site visitor and sends an email containing the information collected from the form. Many sites offer just such a contact form. In fact, these forms have become so commonplace on the Internet that Expression Web users expect to be able to create one with a few clicks here and there. Such an expectation is certainly understandable, but the truth is that creating a contact form with email capabilities isn't that easy.

In this chapter, we'll walk through creating an email contact form that sends email using ASP.NET. One of the main benefits of using ASP.NET is that you can take advantage of the ASP.NET validation controls available in Expression Web. As you'll see later in this chapter, these validation controls provide a powerful means of ensuring that the data you collect is what you expect.

The first step in creating an email contact form is to create the actual form itself.



note

If you'd like to download the completed form, you can do so from the website that accompanies this book at www.informit.com/register.

Creating the Contact Form

The first step in creating our contact form is to create the form itself. We'll create a fairly simple form, but you can easily add additional fields to the form later. To create the form, follow these steps:

1. Click File, New, Page.
2. Select ASPX from the page types and make sure to select C# as your language.
3. Type **Enter your name:** and press Enter.
4. Add a new ASP.NET TextBox control from the Standard section of the ASP.NET controls in the Toolbox.
5. Make sure that the TextBox you just added is selected, and activate the Tag Properties panel.
6. Change the ID property of the TextBox to Name, as shown in Figure 35.1. If you want, you can also change the width of the TextBox so that it's wider than the default size.

**tip**

If the Tag Properties panel isn't visible, select Panels, Tag Properties to activate it.

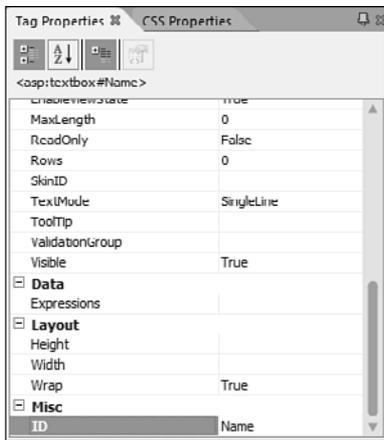


Figure 35.1

The Tag Properties panel is a convenient and easy way to set properties on ASP.NET controls.

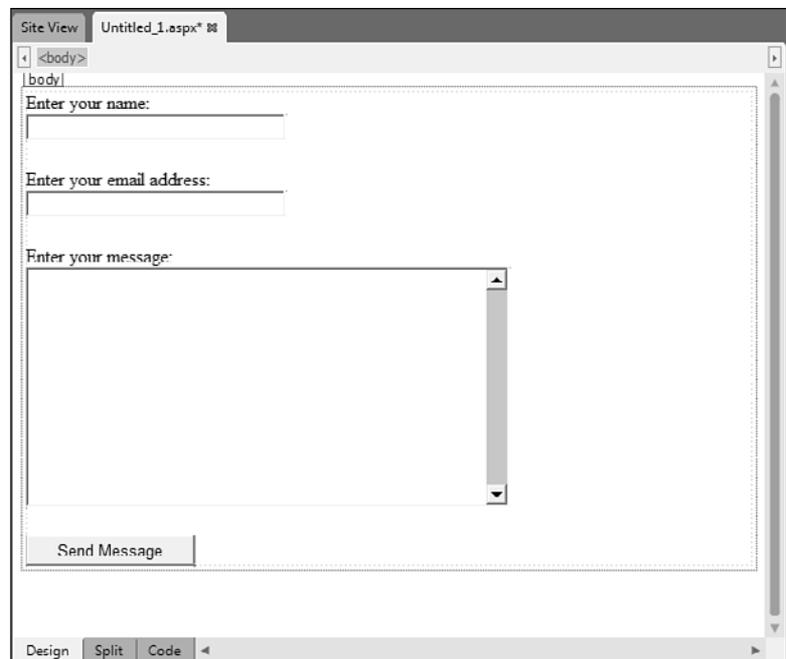
7. Press the right arrow key to deselect the TextBox control, and press Enter twice to add two new lines.
8. Type **Enter your email address:** and press Enter.
9. Add a new TextBox control.
10. Set the ID property of the TextBox control to Email.

11. Add two new lines after the TextBox control.
12. Type **Enter your message:** and press Enter.
13. Add a new TextBox control.
14. Set the ID of the TextBox control to Message.
15. Set the TextMode property to MultiLine.
16. Set the Height property to 200 and the width property to 400.
17. Press Enter twice to add two new lines.
18. Add a new ASP.NET Button control.
19. Set the Width property to 140.
20. Set the Text property to Send Message.
21. Set the ID property to Send.
22. Save the page as default.aspx.

Your form should now look like the one shown in Figure 35.2.

Figure 35.2

The form's design has been completed, but it doesn't do anything yet.



We now have the necessary ASP.NET controls to collect information from site visitors. However, visitors can submit the form without adding any information or entering a valid email address. Fortunately, ASP.NET offers controls that allow us to add form validation easily.

Adding and Configuring ASP.NET Validation Controls

When you use forms in a site, you should always validate the user's input so that you ensure that you get the data you are looking for. In our case, we want to make sure that none of the form fields are empty. We also want to make sure that the email appears to be a valid email address so that we can send a copy of the form to the visitor who filled it out. We'll use two different ASP.NET validation controls to do this: the `RequiredFieldValidator` and `RegularExpressionValidator` controls.

Adding the Validation Controls

Because we want to ensure that visitors fill in all our fields, we'll add a `RequiredFieldValidator` to each control. Click the `Name` `TextBox` control, and then press the right arrow key to move the insertion point just to the right of the `TextBox`. Press the spacebar to add a couple of spaces, and then drag and drop a `RequiredFieldValidator` control from the Validation section of the toolbox as shown in Figure 35.3.

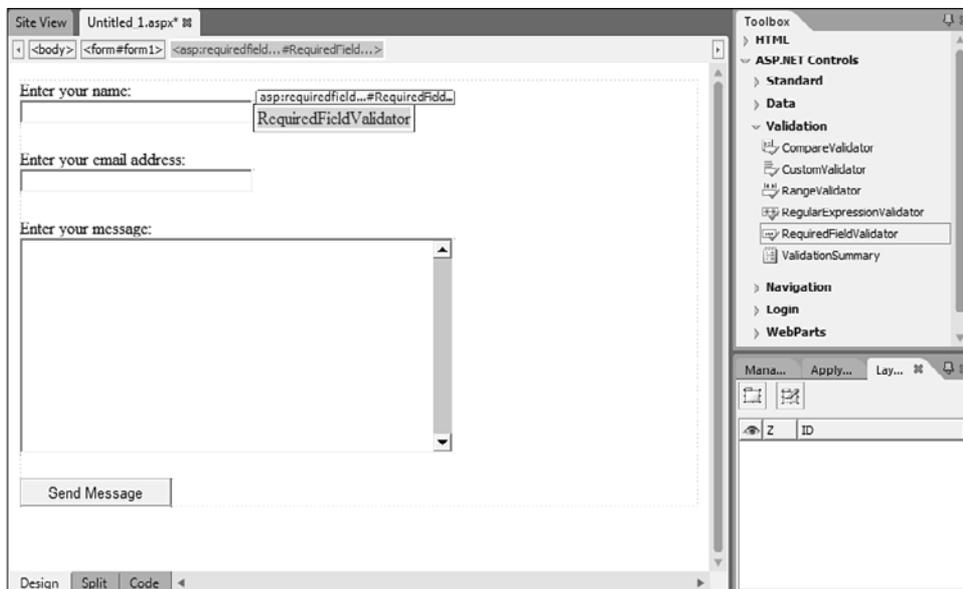
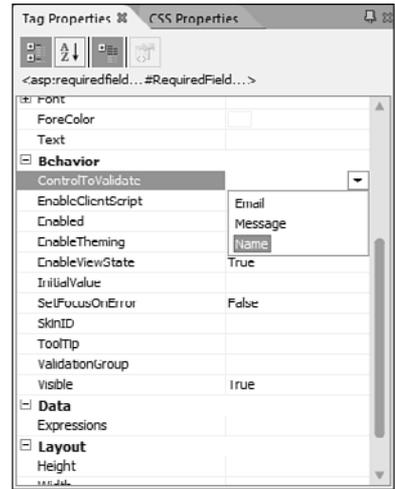


Figure 35.3 ASP.NET validation controls are located in the Validation section of the Toolbox.

Select the `RequiredFieldValidator` control you just added to the page and set the `ErrorMessage` property to `Name is Required`. Click the drop-down next to the `ControlToValidate` property and select `Name`, as shown in Figure 35.4.

Figure 35.4

The `ControlToValidate` property associates a validator control with a control on the page.



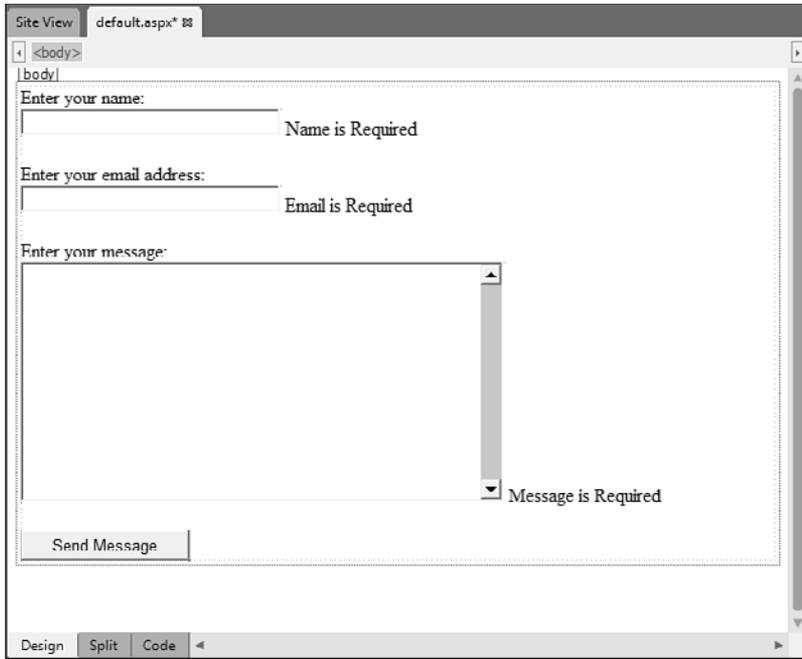
Repeat the same process to add a `RequiredFieldValidator` control next to the `Email` `TextBox` and the `Message` `TextBox` control. Set the `ErrorMessage` property appropriately for each of the `TextBox`s, and select the correct control for the `ControlToValidate` property. Your form should now look like the one shown in Figure 35.5.

➡ *For more information on ASP.NET validation controls, see Chapter 29, “Form Validation with ASP.NET.”*

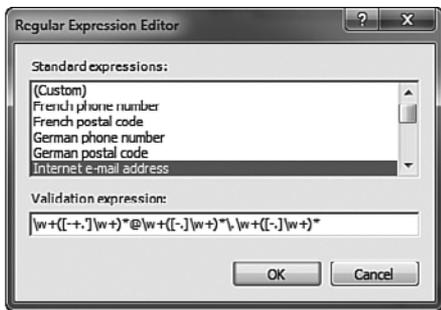
Finally, we need to add a `RegularExpressionValidator` to the `Email` `TextBox` so that we can ensure that the text entered in that field is in the form of an email address.

Add a `RegularExpressionValidator` control to the right of the `RequiredFieldValidator` control that you added to the `Email` `TextBox` earlier. Set the `ErrorMessage` property to `Email Invalid` and set the `ControlToValidate` property to `Email`. Click the ellipse next to the `ValidationExpression` property and select `Internet Email Address` from the list of expressions, as shown in Figure 35.6. Click `OK`.

We have one final step to complete the configuration of the validation controls. If a site visitor enters a value in the `Email` `TextBox` that isn't in the format of an email address, the error message for the `RegularExpressionValidator` control appears too far away from the `Email` field, as shown in Figure 35.7, because ASP.NET is reserving space for the `RequiredFieldValidator` control that's also used by the `Email` field.

**Figure 35.5**

The form now has a RequiredFieldValidator control next to each TextBox.

**Figure 35.6**

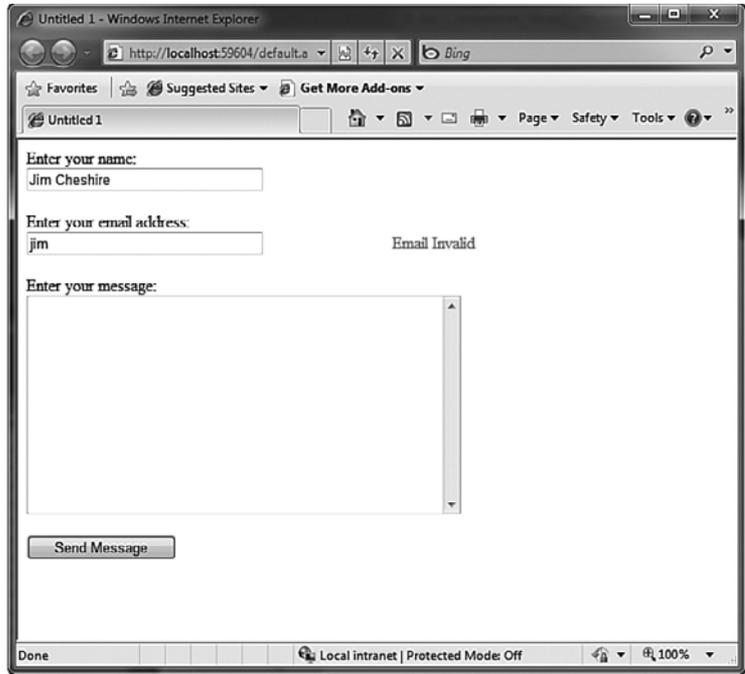
The ValidationExpression property of the RegularExpressionValidator control makes it easy to validate form data against popular patterns.

To fix this problem, select the RequiredFieldValidator control next to the Email field and change the DisplayMode property from Static to Dynamic. ASP.NET will then no longer reserve space on the page for the RequiredFieldValidator control, and the error message for the RegularExpressionValidator control will appear immediately to the right of the Email field.

Our form now has a significant amount of functionality with regard to validation, but it doesn't do anything when you submit it. For the form to send email, we need to add some ASP.NET code to it.

Figure 35.7

The `RegularExpressionValidator`'s error message appears too far from the Email field by default.



Writing ASP.NET Code to Send Email

When a visitor fills out our form and clicks the Send Message button, we want an email to be sent to an email address that we specify and a copy to be sent to the user who filled out the form. You might be surprised at how little code is required to send email using ASP.NET. Listing 35.1 shows the code that is used to send the email. (Line numbers appear for reference only.)

Listing 35.1 Code to Send Email

```
1 <%@ Import Namespace="System.Net.Mail" %>
2 <%@ Import Namespace="System.Text" %>
3 <script runat="server">
4     protected void Page_Load(object sender, EventArgs e)
5     {
6         if (IsPostBack)
7         {
8             SmtpClient sc = new SmtpClient("smtp.yourServer.com");
9             StringBuilder sb = new StringBuilder();
10            MailMessage msg = null;
11            sb.Append("Email from: ");
12            sb.Append(Name.Text);
13            sb.Append(" ");
```

```
14     sb.Append(Email.Text);
15     sb.Append("\n");
16     sb.Append("Message  : ");
17     sb.Append(Message.Text);
18     sb.Append("\n");
19     msg = new MailMessage(Email.Text,
20     "yourEmail@domain.com", "Message from Web Site",
21     sb.ToString());
22     MailAddress CopyAddress = new MailAddress(Email.Text);
23     msg.CC.Add(CopyAddress);
24     sc.Send(msg);
25     if (msg != null)
26     {
27         msg.Dispose();
28     }
29 }
30 }
31 </script>
```

Lines 1 and 2 of this code are `@Import` directives that import a couple of .NET Framework namespaces that we use in the code. By importing these namespaces, we can refer to classes within them (such as `StringBuilder` and `MailMessage`) without using the entire name. In other words, instead of `System.Text.StringBuilder`, after we import the `System.Text` namespace, we can simply use `StringBuilder`.

Line 3 is a typical `<script>` element with the addition of a `runat` attribute set to `server`. The `runat` attribute lets ASP.NET know that the code in the `<script>` block is server-side ASP.NET code and not client-side script.

Line 4 is the signature of the `Page_Load` event. It specifies that the code within the curly braces (lines 5 and 28) runs every time the ASP.NET page loads. Obviously, we don't want the form to try to send email unless the page is loading because the Send Message button was clicked, so line 6 checks to see whether the loading of the page is caused by a postback (the result of a form being posted) and runs the code to send mail only when a postback occurs.

Line 8 creates a new `SmtpClient` instance. `SmtpClient` is a class in the .NET Framework that allows you to easily connect to an Internet mail server. You will want to replace `"smtp.YourServer.com"` with the address of your SMTP mail server. The SMTP mail server is typically the same SMTP server that you use when configuring your email software, but if you don't know what it is, ask your hosting company.

Line 9 creates a new instance of the `StringBuilder` class. The `StringBuilder` class is a specialized class that efficiently handles combining string (text) values.

Line 10 declares a `MailMessage` object. The `MailMessage` class is part of the .NET Framework as well and allows for easily sending mail using the .NET Framework.

 **note**

Namespaces and classes are terms used in object-oriented programming. It's not necessary to understand what these terms mean in order to use this code. If you are interested in learning more about object-oriented programming, read *Sams Teach Yourself Object Oriented Programming in 24 Hours* from Sams Publishing.

Lines 11–18 use the `Append` method of the `StringBuilder` class to build the body of the email message. The value that appears in parentheses after the `Append` method is tacked onto the end of the existing `StringBuilder` value until the entire mail message has been created.

Lines 19–21 set the `msg` variable declared on line 10 to a new instance of the `MailMessage` class. When we create the `MailMessage` instance, we specify the sending address of the email, the destination email address, the subject of the email, and the body of the email. The body of the email is created using the `ToString` method of the `StringBuilder`. The `ToString` method gives you a string made up of all the text you appended to the `StringBuilder` in lines 11–18.

Line 22 creates a new `MailAddress` instance that is used to copy the user who filled out the form when the mail is sent. Line 23 adds the `MailAddress` created in line 22 to the CC for the mail.

Finally, line 24 sends the message using the `Send` method of the `SmtplibClient` instance that you created on line 8.

Lines 25–28 contain cleanup code that uses the `Dispose` method to clean up the `MailMessage` after the mail has been sent. Doing this is a best practice when dealing with the .NET Framework.

I realize that this is a lot of code to throw at those of you who aren't programmers. If you find yourself feeling completely lost at this point, don't worry about it—you don't need to understand all this code to send email with ASP.NET. Simply copy the code to your page, replace `smtp.yourServer.com` with your server name, and replace `yourEmail@domain.com` with your email address, and you're good to go.

If you browse the form at this point, fill in the information, and click `Send Message`, you should receive an email with the information you entered into the form.

**tip**

If you get an error when submitting the form, you can get more information about what went wrong by changing the `@Page` directive on the page. Simply change `<%@Page Language="C#" %>` to `<%@Page Language="C#" Debug="True" %>` and you'll get additional information about any errors.

Be sure to remove the `Debug` attribute (or set it to `False`) before your page goes live.

Displaying a Confirmation Page

When you click `Send Message` in the form, the mail is sent as expected, but the user isn't presented with any kind of helpful message indicating that an email has been sent. It would be much more user-friendly to have a confirmation page that lets the user know that his or her form submission was successful.

If you want to display a confirmation page after sending the email, first create a confirmation page that displays a helpful message such as "Thank you for sending your message!" You then need to add one line to the ASP.NET code to redirect the user to the confirmation page after the form is submitted.

Save your confirmation page as `confirm.aspx`, and then add the following line of code immediately below line 28 in Listing 35.1:

```
Response.Redirect("confirm.aspx");
```

This line of code causes the user's browser to be redirected to a page called `confirm.aspx` after the form is submitted.

This page intentionally left blank

EXPRESSION WEB 4 ADD-IN BASICS

Add-ins in Expression Web

No software application can meet all needs for all people. No matter how many features Microsoft packs into Expression Web, there will always be designers who find that something's missing. Fortunately, Expression Web is extensible so that users of the application can add functionality.

Previous versions of Expression Web were extensible only using a technology called Component Object Model, or COM. Using Visual Studio, Microsoft's development studio, a developer can create a COM add-in that adds functionality to Expression Web. (If you've used any of my add-ins that I distribute from Jimco Software, you've used a COM add-in.) There is a significant barrier to entry for COM add-in development, and most web designers don't possess the skill set necessary to develop them.

Expression Web 4 still allows for COM add-ins, but it adds an exciting new add-in architecture that allows web designers to use existing skills in HTML, XML, and JavaScript to develop add-ins. HTML is used to create user-interfaces, and JavaScript is used to add functionality to Expression Web 4 add-ins.



note

You can view a tutorial on how to create a COM add-in at <http://jimcobook.com/articles/061130/Default.aspx>.

➔ *For more information on adding functionality to an add-in using JavaScript, see "Adding Functionality with JavaScript," in the online Chapter 37 on page 673.*

In this book, I cover everything you need to know to build add-ins using the new JavaScript and HTML extensibility model. I also cover the basics of how you can call into a COM add-in from a JavaScript add-in.

Expression Web 4 JavaScript Add-ins

Using the JavaScript add-in model in Expression Web 4, you can create add-ins with three different main components: panels, dialog boxes, and commands.

Panels are available from the Panels menu. The interface for the panel is developed using HTML code, and functionality is provided using JavaScript. Panels open as floating panels, but users of the add-in can drag and drop the panel to dock it if desired.

Dialog boxes are displayed as a modal dialog of a specific size. The dialog box interface is developed using HTML, and functionality is provided using JavaScript. JavaScript can be used to control the functionality of the dialog box itself as well as add functionality for the add-in.

Commands add a menu command or a toolbar button that runs JavaScript code when the menu item or toolbar button is clicked. Commands are also used to launch dialog boxes.

You may already be familiar with using JavaScript to add functionality to a web page running inside a browser. When you use JavaScript to develop Expression Web add-ins, you are not interacting with a web page in the browser. Instead, you are using JavaScript to interact with files (web pages and other files) inside the Expression Web interface. You can also use JavaScript to interact with Expression Web itself.

I'll explain how to do all of that in the next few chapters, but before we get into the details of implementing an add-in, it's important to understand some of the basics that make up an Expression Web add-in.

The Makeup of Expression Web Add-ins

Expression Web add-ins are made up of a collection of files. HTML files are used to present a user interface for panels and dialog-boxes. JavaScript code is often included directly in the HTML file, but can also be included as a separate script file with a .js file extension. Add-ins can also include any number of collateral files such as image files, CSS files, and so on.

note

Microsoft intends for JavaScript add-ins to be used with disk-based sites only. If you use a JavaScript add-in with a server-based site accessed using HTTP or FTP, you may encounter unexpected results.

I'll provide guidance on what you can expect with server-based sites in Chapter 39, "Expression Web 4 JavaScript API Reference."

tip

Microsoft's documentation says that there are three types of add-ins: panel, dialog box, and command add-ins. However, since it's possible for a single add-in to have any combination of those three, I believe it's more accurate to call these components of an add-in and not refer to them as types of add-ins.

tip

Since add-ins can be created right from within Expression Web, the easiest way to create an add-in is to create a new disk-based site and then create your add-in files within that site.

Add-ins also contain a special XML file called a *manifest*. The add-in manifest (named `addin.xml`) describes the add-in to Expression Web. It's used to tell Expression Web the components included with the add-in, the files included with the add-in, and other information necessary for the proper functioning of the add-in. The manifest also includes information such as the add-in's name, the name of the developer, the version number, and so on.

Listing 36.1 shows a simple add-in manifest for an add-in that displays a panel.

Listing 36.1 A Simple Manifest File

```
<addin>
  <name>Simple Panel</name>
  <version>1.0</version>
  <description>A simple panel add-in for Expression Web.</description>
  <author>Jim Cheshire</author>
  <homepage>http://www.jimcosoftware.com</homepage>
  <panel src="default.html" id="panel1"
    title="Simple Panel" filetype="HTML-DOM"
    activate="enableControls" deactivate="disableControls" />
</addin>
```

Add-in files can be saved into any folder, but the manifest must be inside the root folder for the add-in. Once you're ready to use an add-in, you simply Zip the folder where the files are located and then rename the Zip file so that it has a `.xadd` file extension. You can then install the add-in using the Add-ins dialog available by selecting Tools, Add-ins in Expression Web.

I'll cover all of this in explicit detail in the next few chapters, but what I want you to take away from this is that no specialized tools or complicated skill sets are required to create and deploy Expression Web 4 add-ins. You can create them right inside Expression Web, and you can create an installable package by simply renaming a Zip file. It really couldn't be simpler.

XML Basics

Before I go into the details of the add-in manifest, you'll need to know some basics about XML. XML syntax is actually used for the XHTML code with which you are probably at least somewhat familiar.

At the top level of an XML file is the *root element*. There is exactly one root element; no fewer, no more. Beneath the root element are *child elements*. There can be any number of child elements in an XML file, and each child element can also have its own child elements. The element directly above a child element is known as the *parent element*.

In the manifest in Listing 36.1, the `<addin>` element is the root element. All other elements are child elements.

All elements may contain one or more *attributes*. An attribute is defined within the XML element itself and consists of the attribute name followed by the attribute value in quotes. The following line of XML shows an attribute called `developer` with a value of `yes`.

```
<addin developer="yes">
```



caution

XML is case-sensitive, so pay attention to case when writing XML.

Attribute names are case-sensitive, and attribute values must be enclosed in quotes.

All child elements in an XML file may contain attributes or text content. Text content is textual information surrounded by a particular element. In the XML code that follows, `Simple Panel` is the text content for the `<name>` element.

```
<name>Simple Panel</name>
```

The `<` and `&` characters are not legal characters for XML content. If you want to use these characters for attribute or text content, you need to use what's called an *entity reference*. The entity reference for `<` is `<`; and the entity reference for `&` is `&`. You might notice that both entity references contain the `&` character. The only time the `&` character is valid in an XML file is when it is used with an entity reference.

The following XML snippet defines a name of Jack & Jill Add-ins.

```
<name>Jack &amp; Jill Add-ins</name>
```

Armed with this basic information about XML, you are now ready to learn the details about the add-in manifest. This chapter covers all the elements and attributes that you can use in an add-in manifest. Later chapters go into much more detail about how to use specific elements and attributes.

**tip**

Some XML files use camel case for attribute names. However, in an add-in manifest, all attribute names are lowercase.

**note**

An XML element can contain both attributes and text content.

**note**

If you'd like more information about XML, read *Special Edition Using XML* from Que Publishing.

General Manifest Elements and Attributes

As mentioned previously, the `<addin>` element is the root element of the add-in manifest. Four optional attributes are available for the `<addin>` element.

src (optional)

```
<addin src='script.js'>
```

The `src` attribute is used to specify JavaScript source files that contain functions that the manifest references. This attribute is typically used for commands that don't have a user interface. Scripts that are used with panels and dialog boxes are referenced within the HTML files used with those components.

Source file paths are relative to the add-in's top-level folder, and you can add multiple source files by separating them with a comma.

**tip**

Each element and attribute documented in the following sections includes a graphical representation of where that particular element or attribute fits within the manifest's hierarchy.

legacy (optional)

```
<addin legacy='yes'>
```

The `legacy` attribute is set to `yes` when an add-in needs to access the legacy object model for Expression Web. The legacy object model provides access to the object model used by add-in developers in previous versions of Expression Web, and it has some additional functionality compared to the JavaScript API.

Use of the legacy object model is outside the scope of this book.

developer (optional)

```
<addin developer='yes'>
```

When set to `yes`, the `developer` attribute aids in testing and troubleshooting your add-in during development. By default, the context menu that appears when you right-click on a page is disabled for add-in panels and dialog boxes. By including the `developer` attribute with a value of `yes`, you can remove this restriction, allowing you to easily refresh the user interface of your panel or dialog box, view the source of your interface, and so on.

There are other benefits to setting the `developer` attribute to `yes` when you are developing your add-in. I'll cover how you can use developer mode to debug your add-in in Chapter 39.



note

You should not set the `developer` attribute to `yes` unless you are in the process of developing your add-in because doing so enables debugging tools that are not meant for end-users.

navigationalallowed (optional)

```
<addin navigationalallowed='yes'>
```

By default, Expression Web does not allow you to navigate away from the current page from within a panel or a dialog box. For example, if your dialog box contains a hyperlink, clicking the hyperlink does nothing by default.

In some situations, it may be necessary to allow a user of your add-in to navigate to other pages within your add-in's user interface. For example, if your add-in's interface is in the form of a wizard, you may want to implement each step of the wizard using a separate page. In such a scenario, setting the `navigationalallowed` attribute to `yes` allows a user to navigate through the pages of your wizard.

<name> (required)

```
<addin>  
  <name='My Add-in' />
```

The <name> element specifies the name of your add-in. Once your add-in is installed into Expression Web, the value specified by the <name> attribute is displayed in the Manage Add-ins dialog as shown in Figure 36.1.

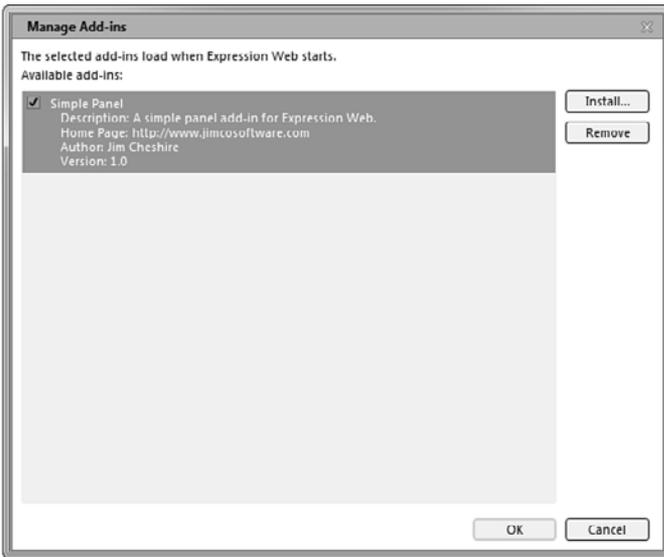


Figure 36.1

The name shown in the Manage Add-ins dialog is controlled by the <name> element. This add-in uses the manifest shown in Listing 36.1.

<description> (optional)

```
<addin>  
  <description='Simple Add-in' />
```

The <description> element is used to provide a description of your add-in that appears in the Manage Add-ins dialog as shown previously in Figure 36.1

The <description> element is optional, but it's a good idea to include it so that users of your add-in can more easily identify it inside the Manage Add-ins dialog.

<author> (optional)

```
<addin>  
  <author='Jim Cheshire' />
```

The `<author>` element provides a means of displaying the add-in author's name in the Manage Add-ins dialog.

`<version>` (optional)

```
<addin>  
  <version='1.0.21.12' />
```

The `<version>` element allows for the inclusion of a version number for your add-in, and the value specified here is displayed in the Manage Add-ins dialog. The format of the version number is `##.##.##` where the first number is the major version and the others are minor version numbers.

If you include a `version` element, the major version number is required and minor version numbers are optional.

`<homepage>` (optional)

```
<addin>  
  <homepage='www.jimcobook.com' />
```

The `<homepage>` element allows you to include a URL where users of your add-in can get additional information. The URL you specify is included in the Manage Add-ins dialog, but it is not displayed as a clickable link.

`<minversion>` (optional)

```
<addin>  
  <minversion='4.5' />
```

The `<minversion>` element is used to specify the minimum version of Expression Web that is required to use your add-in. The version number is specified using the same syntax used with the `<version>` element.

In the current version of Expression Web, the `<minversion>` element is ignored.

`<guid>` (optional)

```
<addin>  
  <guid='a85b04a4-1176-4ef6...'/>
```

The `<guid>` element is used to specify a unique identifier for your add-in in the form of a GUID. The current version of Expression Web doesn't use this element, but it may be used in the future for helping with upgrading add-ins.

**tip**

You can use the page at www.guidgen.com to generate a GUID for your add-in.

<load> (optional)

```
<addin>
  <load type="MyNs.MyAddin, MyNS" name="myAsmbly" />
```

The `<load>` element is used to load a *managed assembly* (a .NET Framework DLL) that can add additional functionality to your add-in.

The `<load>` element has two attributes. The `type` attribute specifies the namespace and class name of your managed class along with the *assembly* (the DLL file) that contains the class. The `name` attribute specifies the JavaScript namespace you want to use to access the managed class from JavaScript.

The following `<load>` element loads a class called `ManagedNamespace.UtilClass` in an assembly called `myClass.dll`. Functions in `UtilClass` can be called from JavaScript using the `util` namespace.

```
<load type="ManagedNamespace.UtilClass, myClass" name="util" />
```

I cover calling managed classes from JavaScript in much more detail in Chapter 37, “Creating a Basic Add-in,” so if this is somewhat confusing to you at this point, don’t worry.

➡ *For more information on accessing managed classes from JavaScript, see “Accessing Managed Classes from JavaScript,” in the online Chapter 37 on page 682.*

Commands and Dialog Boxes

As mentioned previously, a command is a menu item or a toolbar item that runs a JavaScript function when clicked. A command can also be used to launch a dialog box. (Panels aren’t launched using a command because they are automatically added to the Panels menu in Expression Web.)

The `<command>` element is a child of the `<addin>` element, and it has three attributes.

id (required)

```
<addin>
  <command id='MyAddinCommand'>
```

The `id` attribute is a unique string that identifies the command. Expression Web uses the `id` attribute to correctly identify your command and respond to a user interacting with it.

filetype (optional)

```
<addin>
  <command filetype='HTML-DOM'>
```



tip

It’s a good idea to use a value such as part of your name in your ID so that you don’t end up using the same ID as another add-in. For example, if I use `MetaTag` as the ID of my command, it’s possible that another add-in developer might use the same value. However, if I use `JimcoMetaTag` as my ID, it’s almost certainly going to be unique in all cases.

It's likely that your add-in will target specific types of files. For example, if your add-in is designed to generate HTML code for web pages, you wouldn't want users to be able to access your add-in when a CSS file is active in Expression Web.

The `filetype` attribute allows you to define which types of files are valid for your add-in. If the type of file that is active in Expression Web matches a type specified in the `filetype` attribute, any menu items or toolbar buttons for the command are enabled. Otherwise, they are disabled.

The following are valid values for the `filetype` attribute:

- **HTML**—A file that has a `.htm` or `.html` file extension.
- **PHP**—A PHP page with a `.php` file extension.
- **ASP**—An ASP or ASP.NET page. File extensions can be `.asp`, `.aspx`, `.asmx`, or `.ascx`.
- **CSS**—A style sheet file with a `.css` file extension.
- **JS**—A JavaScript file with a `.js` file extension.
- **HTML-DOM**—Any file that contains an HTML DOM. For example, any `.aspx` file, `.php` file, `.htm` file, and so on.
- **TXT**—A file with a `.txt` file extension.
- **XML**—A file with a `.xml` file extension.

You can specify multiple file types by separating multiple values with a comma.

onclick (optional)

```
<addin>
  <command onclick='myFunction()' '>
```

The `onclick` attribute is used to run a JavaScript function or expression when a command's menu item or toolbar button is clicked. The `onclick` attribute can refer to a function inside an HTML file when using a dialog box or within a JavaScript file specified by the `src` attribute of the `<addin>` element.

If your add-in uses a dialog box, the `onclick` method is used to launch the dialog box by calling the method `xweb.application.showModalDialog`. In the following example, the `onclick` attribute specifies that a dialog box will be displayed that contains the contents of `default.html`, is titled My Dialog, has a width of 400 pixels and a height of 600 pixels, and is resizable and scrollable.

```
<command id="myDialog"
  onclick="xweb.application.showModalDialog('default.html', 'My Dialog',
  'dialogwidth:400; dialogHeight:600; resizable:yes; scroll:yes')">
```



tip

If you don't specify a value for the `filetype` attribute, menu items or toolbar items for your command will always be enabled.



note

Keep in mind that the `filetype` attribute of the `<command>` element doesn't affect whether a panel's menu item is enabled or disabled. Panels are completely separate from commands. I'll cover panels later in this chapter.

Menus and Toolbars

As mentioned earlier, panels are launched using the Panels menu in Expression Web. However, to run commands or launch dialog boxes, you need to add a menu item or a toolbar button (or both) for your add-in. Menu items are added using the `<menuitem>` element, and toolbar buttons are added using the `<toolbaritem>` element, both of which are child elements of the `<command>` element.



tip

You may be wondering what you should click to cause the function or expression specified in the `onclick` attribute to run. I'll cover that in the next section.

`<menuitem>` (optional)

```
<addin>
  <command>
    <menuitem>
```

The `<menuitem>` element is a child of the `<command>` element and is used to add a menu item to an existing menu in Expression Web. Expression Web will then control whether the menu item is enabled based on the `filetype` attribute in the `<command>` element. When the menu item is clicked, the function or expression you specified in the `onclick` attribute of the `<command>` element will be executed.

The `<menuitem>` element has five available attributes.



tip

It's not uncommon for an add-in to define a menu item and a toolbar item for the same command.

parent (required)

```
<addin>
  <command>
    <menuitem parent='MENU_Tools'>
```

The `parent` attribute is used to specify which menu in Expression Web should contain your menu item. The value specified for the `parent` attribute is the internal ID used by Expression Web for the menu. The internal ID is in the format of `MENU_` followed the name of the menu. For example, the internal ID of the File menu is `MENU_File`.

Unless you also include a value for the `before` attribute, your command's menu item appears last on the menu that you specify.



For a complete reference of internal IDs for Expression Web menus, see "Menu and Command Bar Reference," p. 630, later in this chapter.

before (required)

```
<addin>
  <command>
    <menuitem before='MENU_Tools_AddIns'>
```

The `before` attribute allows you to specify where to place your menu item on the menu that you specified using the `parent` attribute. The value used for the `before` attribute is the internal ID of a menu item. The format is typically the internal ID of the menu followed by the text of the menu item. For example, the internal ID of the Save item on the File menu is `MENU_File_Save`.

Menus also have horizontal bars that separate groups of menu items. The internal IDs of these separator bars don't follow the naming convention of other menu items. However, you can refer to the "Menu and Command Bar Reference" section at the end of this chapter to find the internal ID for them.

label (required)

```
<addin>
  <command>
    <menuitem label='Cool &Add-in'>
```

The `label` attribute is used to set the text that appears on the menu item for your add-in. When specifying a value for the `label` attribute, you should always define a hotkey for your menu item so that it is accessible for all users. To define a hotkey, add an underscore prior to the letter you want to use for the hotkey.

imageSrc (optional)

```
<addin>
  <command>
    <menuitem imageSrc='MyAddin.png'>
```

If you'd like for your menu item to include an image, specify the path to the image using the `imageSrc` attribute. The image you specify appears to the left of the menu item, and it must be 16 pixels by 16 pixels and either JPG or PNG file format. Expression Web does not support JPEG 2000 format, so PNG is the only option if you need a transparent background.



tip

There's no guarantee that your menu item will appear exactly where you specify. For example, if two add-ins are installed and both specify that a menu item should appear before Save on the File menu, the last one to load appears before Save, and the other add-in's menu item is bumped up one position on the File menu.



tip

Hotkeys appear underlined when the Alt key on the keyboard is pressed. Users can select menus and menu items by pressing the hotkey after pressing Alt.



tip

It's best to use PNG for your menu item's image so that you can take advantage of transparency. Expression Web allows a user to use his or her Windows color scheme so you can't predict the background color of menus in Expression Web. If you use PNG with transparency when you save your menu item's image, it looks more professional.

tooltip (optional)

```
<addin>
  <command>
    <menuitem tooltip='Insert a Widget'>
```

The `tooltip` attribute is used to specify the text that should appear in the pop-up tooltip when a user hovers the mouse over your menu item. The tooltip should concisely describe what the menu item does.

<toolbaritem> (optional)

```
<addin>
  <command>
    <toolbaritem>
```

The `<toolbaritem>` element is a child of the `<command>` element and is used to define a toolbar button on a toolbar. When the toolbar button is clicked, the function specified in the `onclick` attribute of the `<command>` element is executed.

The `<toolbaritem>` element has five attributes.

parent (required)

```
<addin>
  <command>
    <toolbaritem parent="COMMANDBAR_Standard">
```

The `parent` attribute specifies which Expression Web toolbar will contain your add-in's toolbar item. The value you assign to the `parent` attribute is in the format `COMMANDBAR_` followed by the name of the toolbar.

To find the name you should use for a specific toolbar, right-click on a toolbar in Expression Web. The name used for the `parent` attribute is the name you see on the context menu, but you need to remove any spaces. For example, to add your toolbar item to the Style Application toolbar, use `COMMANDBAR_StyleApplication` for the `parent` attribute.



tip

You can create your own toolbar for your toolbar item by specifying a toolbar name that doesn't already exist. For example, to create a new toolbar called `MyToolbar`, use the value `MyToolbar` for the `parent` attribute. Spaces are not allowed.

before (optional)

```
<addin>
  <command>
    <toolbaritem before="COMMANDBAR_Common_Save">
```

The `before` attribute controls where your toolbar item appears on the toolbar specified in the `parent` attribute. Your toolbar item appears before the item identified by the `before` attribute.

The value for the `before` attribute is typically the value of the `parent` attribute followed by an underscore and the name (without spaces) of the toolbar item before which your toolbar item should appear. Toolbars have vertical separator bars between groups of buttons, and you can use the internal ID of one of these separators for the `before` attribute as well.

➔ *For a complete reference of internal IDs for Expression Web menus, see “Menu and Command Bar Reference,” p. 630, later in this chapter.*

label (optional)

```
<addin>
  <command>
    <toolbaritem label="Add Widget">
```

Use the `label` attribute to set a text label for your toolbar item. A toolbar item can appear either as text or an image. If you want your item to appear as text, specify a value for `label` and do not specify a value for the `imagesrc` attribute.

imagesrc (optional)

```
<addin>
  <command>
    <toolbaritem imagesrc="addinitem.png">
```

Use the `imagesrc` attribute when using an image for your toolbar item. The image should be 16×16 pixels and either in PNG or JPEG format.

Because a toolbar item can be either a text button or an image button, if a value is specified for the `imagesrc` attribute, a value specified for the `label` attribute will be ignored.

tooltip (optional)

```
<addin>
  <command>
    <toolbaritem tooltip="Insert Widget">
```

The `tooltip` attribute allows you to add a textual tooltip that appears when a user hovers the mouse over your toolbar item.

Panels

If your add-in's user interface is around 275 or fewer pixels wide, it may be suitable for a panel instead of a dialog box. Panels are defined using the `<panel>` element.

<panel> (optional)

```
<addin>  
  <panel>
```

The `<panel>` element defines a panel that can be opened using the Panels menu. Panels are floating by default, but a user can dock the panel by dragging and dropping it to an edge in the Expression Web interface.

The `<panel>` element has nine attributes.

id (required)

```
<addin>  
  <panel id="MyPanel">
```

The `id` is a unique value that identifies your panel to Expression Web. Just as with the value used for the `id` attribute of the `<command>` element, you should ensure that the value is unique by including a common string not likely to be included by another add-in developer.

title (required)

```
<addin>  
  <panel title="My Add-in">
```

The value given for the `title` attribute appears on both the Panels menu and also in the tab for your panel. Just as with menu items, you can use an underscore to define a hotkey for your panel's menu item on the Panels menu.

src (required)

```
<addin>  
  <panel src="panel.htm">
```

The `src` attribute is used to specify the web page that contains the source for your panel's user interface.

filetype (optional)

```
<addin>  
  <panel filetype="HTML-DOM">
```

The `filetype` attribute is used to specify when the function or expression that is specified for the `activate` and `deactivate` attributes will run. When assigning a value to the `filetype` attribute, use one of the same values I defined previously for the `filetype` attribute of the `<command>` element.

activate (optional)

```
<addin>
  <panel activate="activatePanel()">
```

Expression Web allows users to open a panel at any time. Therefore, to control whether a user can interact with your panel, you need to control whether the controls in your panel's user interface are enabled.

Use the `filetype` attribute to specify which page type is applicable for your panel. You can then use the `activate` attribute to specify a JavaScript function or expression that enables the necessary control in your panel's interface.

deactivate (optional)

```
<addin>
  <panel deactivate="deactivatePanel()">
```

The `deactivate` attribute is used to deactivate your panel's user interface when the page that is open in Expression Web doesn't match a type specified by the `filetype` attribute. Just as with the `activate` attribute, the value for the `deactivate` attribute is a JavaScript function or expression.

ondocumentchanged (optional)

```
<addin>
  <panel ondocumentchanged="resetPanel()">
```

The `activate` and `deactivate` attributes can be used to control two states of your panel: active and inactive. However, your panel's user interface may also need to be updated in other situations. The `ondocumentchanged`, `ondocumentsaved`, and `onelementchanged` attributes are available for controlling your panel's interface in other situations.

The `ondocumentchanged` attribute allows you to specify a JavaScript function or expression that executes when the page or file that is open in Expression Web is changed to a new page or a different page. You might, for example, want to use the `ondocumentchanged` attribute to specify a JavaScript function that resets your panel's user interface to its initial state.



note

The name `ondocumentchanged` makes you think that changing some content in the active document causes the function or expression you specify to execute. In fact, changing the content isn't what causes a document change event. It's actually changing the document itself that causes the `ondocumentchanged` attribute to come into play.

ondocumentsaved (optional)

```
<addin>
  <panel ondocumentsaved="cleanPanel()">
```

If you require that your panel's interface react to the active document being saved, you can use the `ondocumentsaved` attribute to specify a JavaScript function or expression that should run when the document is saved.

onelementchanged (optional)

```
<addin>
  <panel onelementchanged="evalPanel()">
```

It's possible that your panel might not be usable for any element on a page. For example, if your panel is designed to work with HTML tables, it doesn't make sense for your panel to be available for user interaction unless a table is active in Expression Web. The `onelementchanged` attribute is designed to handle just such a situation.

By specifying a JavaScript function or expression in the `onelementchanged` attribute, you can determine whether the selected element is applicable to your panel and react accordingly.

Once you have an understanding of the add-in manifest, you have all the skills necessary to create menu items and toolbar buttons, control when those menu items and toolbar buttons are enabled or disabled, create panels, and much more.

In the next few chapters, you learn how to control the user interface of your panels and dialog boxes and how to programmatically interact with Expression Web and the web pages open within it.

Menu and Command Bar Reference

When defining menu items and toolbar items in your manifest file, it's necessary to know the internal ID of Expression Web's menu items and toolbar items. The following reference allows you to determine the correct value for your `parent` and `before` attributes.



note

An element is a representation of an HTML tag in a page.



tip

If you need to enable or disable your panel based on the selected element, you can usually use the same JavaScript function or expressions used for the `activate` and `deactivate` attributes.



caution

As a user is editing a document in Expression Web, the `onelementchanged` event happens often. To prevent performance problems, make sure that the JavaScript function or expression you specify for the `onelementchanged` attribute is lightweight.

Menus

The following tables list the internal IDs of all menus and menu items in Expression Web. Use the value specified in the Parent column for the `parent` attribute and the value in the Internal ID column for the `before` attribute in your manifest file.

Table 36.1 shows the internal IDs of all items on the File menu.

Table 36.2 lists the internal IDs of items on the Edit menu.

Table 36.3 lists the internal IDs of items on the View menu.

Table 36.4 lists the internal IDs of items on the Insert menu.

Table 36.5 lists the internal IDs of items on the Format menu.

Table 36.6 lists the internal IDs of items on the Tools menu.

Table 36.7 lists the internal IDs of items on the Table menu.

Table 36.8 lists the internal IDs of items on the Site menu.

Table 36.9 lists the internal IDs of items on the Data View menu.

Table 36.10 lists the internal IDs of items on the Window menu.

Table 36.11 lists the internal IDs of items on the Help menu.

Table 36.1 File Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	File	MENU_File
MENU_File	New	MENU_File_New
MENU_File_New	Page...	MENU_File_New_Page
MENU_File_New	[Separator]	Separator_1
MENU_File_New	HTML	MENU_File_New_HTML
MENU_File_New	ASPX	MENU_File_New_ASPX
MENU_File_New	ASP	MENU_File_New_ASP
MENU_File_New	PHP	MENU_File_New_PHP
MENU_File_New	CSS	MENU_File_New_CSS
MENU_File_New	[Separator]	Separator_2
MENU_File_New	Folder	MENU_File_New_Folder
MENU_File_New	[Separator]	Separator_3
MENU_File_New	Create from Dynamic Web Template...	MENU_File_New_CreateFromDynamicWebTemplate
MENU_File_New	Create from Master Page...	MENU_File_New_CreateFromMasterPage
MENU_File	Open...	MENU_File_Open

Table 36.1 Continued

Parent	Menu Item	Internal ID
MENU_File	Recent Files	MENU_File_RecentFiles
MENU_File	Close	MENU_File_Close
MENU_File	[Separator]	Separator_4
MENU_File	Save	MENU_File_Save
MENU_File	Save As...	MENU_File_SaveAs
MENU_File	Save All	MENU_File_SaveAll
MENU_File	[Separator]	Separator_5
MENU_File	Import	MENU_File_Import
MENU_File_Import	File...	MENU_File_Import_File
MENU_File_Import	Adobe Photoshop (.psd)...	Menu_File_Import_AdobePhotoshopPSD
MENU_File	[Separator]	Separator_6
MENU_File	Display in SuperPreview	MENU_File_PreviewInBrowser_OpenInSuperPreview
MENU_File	Preview in Browser	MENU_File_PreviewInBrowser
MENU_File_PreviewInBrowser	Preview in Browser 1[-8]	MENU_File_PreviewInBrowser_1[-8]
MENU_File_PreviewInBrowser	[Separator]	Separator_7
MENU_File_PreviewInBrowser	Preview in Multiple Browsers	MENU_File_PreviewInBrowser_PreviewInMultipleBrowsers
MENU_File_PreviewInBrowser	Preview in Multiple Browsers (640x480)	MENU_File_PreviewInBrowser_PreviewInMultipleBrowsers640x480
MENU_File_PreviewInBrowser	Preview in Multiple Browsers (800x600)	MENU_File_PreviewInBrowser_PreviewInMultipleBrowsers800x600
MENU_File_PreviewInBrowser	Preview in Multiple Browsers (1024x768)	MENU_File_PreviewInBrowser_PreviewInMultipleBrowsers1024x768
MENU_File_PreviewInBrowser	[Separator]	Separator_7
MENU_File_PreviewInBrowser	Edit Browser List...	MENU_File_PreviewInBrowser_EditBrowserList
MENU_File	Print	MENU_File_Print
MENU_File_Print	Print...	MENU_File_Print_Print
MENU_File_Print	Print Preview	MENU_File_Print_PrintPreview
MENU_File_Print	Page Setup...	MENU_File_Print_PageSetup

Parent	Menu Item	Internal ID
MENU_File	Properties...	MENU_File_Properties
MENU_File	[Separator]	Separator_9
MENU_File	Source Control	MENU_File_SourceControl
MENU_File	[Separator]	Separator_10
MENU_File	Exit	MENU_File_Exit

Table 36.2 Edit Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Edit	MENU_Edit
MENU_Edit	Undo	MENU_Edit_Undo
MENU_Edit	Redo	MENU_Edit_Redo
MENU_Edit	[Separator]	Separator_11
MENU_Edit	Cut	MENU_Edit_Cut
MENU_Edit	Copy	MENU_Edit_Copy
MENU_Edit	Paste	MENU_Edit_Paste
MENU_Edit	Paste Text...	MENU_Edit_PasteText
MENU_Edit	[Separator]	Separator_12
MENU_Edit	Delete	MENU_Edit_Delete
MENU_Edit	Select All	MENU_Edit_SelectAll
MENU_Edit	[Separator]	Separator_13
MENU_Edit	Find...	MENU_Edit_Find
MENU_Edit	Replace...	MENU_Edit_Replace
MENU_Edit	Incremental Search	MENU_Edit_IncrementalSearch
MENU_Edit	Go To...	MENU_Edit_DynamicLabel3
MENU_Edit	Quick Tag Editor	MENU_Edit_QuickTagEditor
MENU_Edit	[Separator]	Separator_14
MENU_Edit	Code View	MENU_Edit_CodeView
MENU_Edit_CodeView	Increase Indent	MENU_Edit_CodeView_IncreaseIndent
MENU_Edit_CodeView	Decrease Indent	MENU_Edit_CodeView_DecreaseIndent
MENU_Edit_CodeView	[Separator]	Separator_16
MENU_Edit_CodeView	Select Tag	MENU_Edit_CodeView_SelectTag

Table 36.2 Continued

Parent	Menu Item	Internal ID
MENU_Edit_CodeView	Find Matching Tag	MENU_Edit_CodeView_FindMatchingTag
MENU_Edit_CodeView	Select Block	MENU_Edit_CodeView_SelectBlock
MENU_Edit_CodeView	Find Matching Brace	MENU_Edit_CodeView_FindMatchingBrace
MENU_Edit_CodeView	[Separator]	Separator_17
MENU_Edit_CodeView	Insert Start Tag	MENU_Edit_CodeView_InsertStartTag
MENU_Edit_CodeView	Insert End Tag	MENU_Edit_CodeView_InsertEndTag
MENU_Edit_CodeView	Insert HTML Comment	MENU_Edit_CodeView_InsertHTMLComment
MENU_Edit_CodeView	Insert CSS Comment	MENU_Edit_CodeView_InsertCSSComment
MENU_Edit_CodeView	[Separator]	Separator_18
MENU_Edit_CodeView	Toggle Bookmark	MENU_Edit_CodeView_ToggleBookmark
MENU_Edit_CodeView	Next Bookmark	MENU_Edit_CodeView_NextBookmark
MENU_Edit_CodeView	Previous Bookmark	MENU_Edit_CodeView_PreviousBookmark
MENU_Edit_CodeView	Clear Bookmarks	MENU_Edit_CodeView_ClearBookmarks
MENU_Edit_CodeView	[Separator]	Separator_19
MENU_Edit_CodeView	Follow Code Hyperlink	MENU_Edit_CodeView_FollowCodeHyperlink
MENU_Edit_CodeView	Previous Code Hyperlink	MENU_Edit_CodeView_PreviousCodeHyperlink
MENU_Edit_CodeView	Next Code Hyperlink	MENU_Edit_CodeView_NextCodeHyperlink
MENU_Edit	IntelliSense	MENU_Edit_IntelliSense
MENU_Edit_IntelliSense	List Members	MENU_Edit_IntelliSense_ListMembers
MENU_Edit_IntelliSense	Parameter Info	MENU_Edit_IntelliSense_ParameterInfo
MENU_Edit_IntelliSense	Complete Word	MENU_Edit_IntelliSense_CompleteWord
MENU_Edit_IntelliSense	List Code Snippets	MENU_Edit_IntelliSense_ListCodeSnippets

Table 36.3 View Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	View	MENU_View
MENU_View	Page	MENU_View_Page
MENU_View_Page	Design	MENU_View_Page_Design
MENU_View_Page	Split	MENU_View_Page_Split
MENU_View_Page	Code	MENU_View_Page_Code
MENU_View_Page	[Separator]	Separator_20
MENU_View_Page	Snapshot	MENU_View_Page_Snapshot
MENU_View	Site	MENU_View_Site
MENU_View_Site	Folders	MENU_View_Site_Folders
MENU_View_Site	Publishing	MENU_View_Site_Publishing
MENU_View_Site	Site Summary	MENU_View_Site_SiteSummary
MENU_View_Site	Files	MENU_View_Site_Files
MENU_View_Site_Files	All Files	MENU_View_Site_Files_AllFiles
MENU_View_Site_Files	Recently Added Files	MENU_View_Site_Files_RecentlyAddedFiles
MENU_View_Site_Files	Recently Changed Files	MENU_View_Site_Files_RecentlyChangedFiles
MENU_View_Site_Files	Older Files	MENU_View_Site_Files_OlderFiles
MENU_View_Site	Shared Content	MENU_View_Site_SharedContent
MENU_View_Site_SharedContent	Dynamic Web Templates	MENU_View_Site_SharedContent_DynamicWebTemplates
MENU_View_Site_SharedContent	Master Pages	MENU_View_Site_SharedContent_MasterPages
MENU_View_Site_SharedContent	Style Sheet Links	MENU_View_Site_SharedContent_StyleSheetLinks
MENU_View_Site	Problems	MENU_View_Site_Problems
MENU_View_Site_Problems	Unlinked Files	MENU_View_Site_Problems_UnlinkedFiles
MENU_View_Site_Problems	Slow Pages	MENU_View_Site_Problems_SlowPages
MENU_View_Site_Problems	Hyperlinks	MENU_View_Site_Problems_Hyperlinks
MENU_View_Site	Hyperlinks	MENU_View_Site_Hyperlinks
MENU_View	[Separator]	Separator_21
MENU_View	Visual Aids	MENU_View_VisualAids
MENU_View_VisualAids	Show	MENU_View_VisualAids_Show

Table 36.3 Continued

Parent	Menu Item	Internal ID
MENU_View_VisualAids	[Separator]	Separator_22
MENU_View_VisualAids	Block Selection	MENU_View_VisualAids_BlockSelection
MENU_View_VisualAids	Visible Borders	MENU_View_VisualAids_VisibleBorders
MENU_View_VisualAids	Empty Containers	MENU_View_VisualAids_EmptyContainers
MENU_View_VisualAids	Margins and Padding	MENU_View_VisualAids_MarginsAndPadding
MENU_View_VisualAids	CSS Display:none Elements	MENU_View_VisualAids_CSSDisplayNoneElements
MENU_View_VisualAids	CSS Visibility:hidden Elements	MENU_View_VisualAids_CSSVisibilityHiddenElements
MENU_View_VisualAids	ASP.NET Non-visual Controls	MENU_View_VisualAids_ASPNETNonVisualControls
MENU_View_VisualAids	ASP.NET Control Errors	MENU_View_VisualAids_ASPNETControlErrors
MENU_View_VisualAids	Template Region Labels	MENU_View_VisualAids_TemplateRegionLabels
MENU_View	Formatting Marks	MENU_View_FormattingMarks
MENU_View_FormattingMarks	Show	MENU_View_FormattingMarks_Show
MENU_View_FormattingMarks	[Separator]	Separator_23
MENU_View_FormattingMarks	Tag Marks	MENU_View_FormattingMarks_TagMarks
MENU_View_FormattingMarks	Paragraph Marks	MENU_View_FormattingMarks_ParagraphMarks
MENU_View_FormattingMarks	Line Breaks	MENU_View_FormattingMarks_LineBreaks
MENU_View_FormattingMarks	Spaces	MENU_View_FormattingMarks_Spaces
MENU_View_FormattingMarks	Comments	MENU_View_FormattingMarks_Comments
MENU_View_FormattingMarks	Script Blocks	MENU_View_FormattingMarks_ScriptBlocks

Parent	Menu Item	Internal ID
MENU_View_FormattingMarks	Positioned Absolute	MENU_View_FormattingMarks_PositionedAbsolute
MENU_View_FormattingMarks	Aligned Elements	MENU_View_FormattingMarks_AlignedElements
MENU_View_FormattingMarks	CSS Display: none	MENU_View_FormattingMarks_CSSDisplayNone
MENU_View_FormattingMarks	Hidden Form Fields	MENU_View_FormattingMarks_HiddenFormFields
MENU_View	Quick Tag Selector	MENU_View_QuickTagSelector
MENU_View	[Separator]	Separator_24
MENU_View	Ruler and Grid	MENU_View_RulerAndGrid
MENU_View_RulerAndGrid	Show Ruler	MENU_View_RulerAndGrid_ShowRuler
MENU_View_RulerAndGrid	Show Grid	MENU_View_RulerAndGrid_ShowGrid
MENU_View_RulerAndGrid	Snap to Grid	MENU_View_RulerAndGrid_SnapToGrid
MENU_View_RulerAndGrid	[Separator]	Separator_25
MENU_View_RulerAndGrid	Set Origin from Selection	MENU_View_RulerAndGrid_SetOriginFromSelection
MENU_View_RulerAndGrid	Reset Origin	MENU_View_RulerAndGrid_ResetOrigin
MENU_View_RulerAndGrid	[Separator]	Separator_26
MENU_View_RulerAndGrid	Configure...	MENU_View_RulerAndGrid_Configure
MENU_View	Tracing Image	MENU_View_TracingImage
MENU_View_TracingImage	Show Image	MENU_View_TracingImage_ShowImage
MENU_View_TracingImage	Configure...	MENU_View_TracingImage_Configure
MENU_View	Page Size	MENU_View_PageSize
MENU_View_PageSize	W x H Current Page Size	MENU_View_PageSize_DynamicLabel1
MENU_View_PageSize	[Separator]	Separator_27
MENU_View_PageSize	# [Page Size Setting]	MENU_View_PageSize_#
MENU_View_PageSize	[Separator]	Separator_28
MENU_View_PageSize	Modify Page Sizes...	MENU_View_PageSize_ModifyPageSizes
MENU_View	[Separator]	Separator_29
MENU_View	Toolbars	MENU_View_Toolbars
MENU_View_Toolbars	Standard	MENU_View_Toolbars_Standard
MENU_View_Toolbars	Formatting	MENU_View_Toolbars_Formatting

Table 36.3 Continued

Parent	Menu Item	Internal ID
MENU_View_Toolbars	Code View	MENU_View_Toolbars_CodeView
MENU_View_Toolbars	Common	MENU_View_Toolbars_Common
MENU_View_Toolbars	Dynamic Web Template	MENU_View_Toolbars_DynamicWebTemplate
MENU_View_Toolbars	Master Page	MENU_View_Toolbars_MasterPage
MENU_View_Toolbars	Pictures	MENU_View_Toolbars_Pictures
MENU_View_Toolbars	Positioning	MENU_View_Toolbars_Positioning
MENU_View_Toolbars	Style	MENU_View_Toolbars_Style
MENU_View_Toolbars	Style Application	MENU_View_Toolbars_StyleApplication
MENU_View_Toolbars	Tables	MENU_View_Toolbars_Tables
MENU_View	Refresh	MENU_View_Refresh

Table 36.4 Insert Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Insert	MENU_Insert
MENU_Insert	HTML	MENU_Insert_HTML
MENU_Insert_HTML	<div>	MENU_Insert_HTML_Div
MENU_Insert_HTML		MENU_Insert_HTML_Span
MENU_Insert_HTML	Break...	MENU_Insert_HTML_Break
MENU_Insert_HTML	Horizontal Line	MENU_Insert_HTML_HorizontalLine
MENU_Insert_HTML	Image	MENU_Insert_HTML_Image
MENU_Insert_HTML	Inline Frame	MENU_Insert_HTML_InlineFrame
MENU_Insert_HTML	Layer	MENU_Insert_HTML_Layer
MENU_Insert_HTML	Paragraph	MENU_Insert_HTML_Paragraph
MENU_Insert_HTML	[Separator]	Separator_30
MENU_Insert_HTML	More HTML Tags...	MENU_Insert_HTML_MoreHTMLTags
MENU_Insert	ASP.NET Controls	MENU_Insert_ASPNETControls
MENU_Insert_ASPNETControls	Button	MENU_Insert_ASPNETControls_Button
MENU_Insert_ASPNETControls	Checkbox	MENU_Insert_ASPNETControls_Checkbox

Parent	Menu Item	Internal ID
MENU_Insert_ASPNETControls	Checkbox List	MENU_Insert_ASPNETControls_CheckboxList
MENU_Insert_ASPNETControls	Dropdown List	MENU_Insert_ASPNETControls_DropDownList
MENU_Insert_ASPNETControls	Image Button	MENU_Insert_ASPNETControls_ImageButton
MENU_Insert_ASPNETControls	Label	MENU_Insert_ASPNETControls_Label
MENU_Insert_ASPNETControls	Listbox	MENU_Insert_ASPNETControls_Listbox
MENU_Insert_ASPNETControls	Radio Button	MENU_Insert_ASPNETControls_RadioButton
MENU_Insert_ASPNETControls	Radio Button List	MENU_Insert_ASPNETControls_RadioButtonList
MENU_Insert_ASPNETControls	Textbox	MENU_Insert_ASPNETControls_Textbox
MENU_Insert_ASPNETControls	[Separator]	Separator_31
MENU_Insert_ASPNETControls	Script Manager	MENU_Insert_ASPNETControls_ScriptManager
MENU_Insert_ASPNETControls	Script Manager Proxy	MENU_Insert_ASPNETControls_ScriptManagerProxy
MENU_Insert_ASPNETControls	Timer	MENU_Insert_ASPNETControls_Timer
MENU_Insert_ASPNETControls	Update Panel	MENU_Insert_ASPNETControls_UpdatePanel
MENU_Insert_ASPNETControls	Update Progress	MENU_Insert_ASPNETControls_UpdateProgress
MENU_Insert_ASPNETControls	[Separator]	Separator_32
MENU_Insert_ASPNETControls	More ASP.NET Controls...	MENU_Insert_ASPNETControls_MoreASPNETControls
MENU_Insert	PHP	MENU_Insert_PHP
MENU_Insert_PHP	Form Variable	MENU_Insert_PHP_FormVariable
MENU_Insert_PHP	URL Variable	MENU_Insert_PHP_URLVariable
MENU_Insert_PHP	Session Variable	MENU_Insert_PHP_SessionVariable
MENU_Insert_PHP	Cookie Variable	MENU_Insert_PHP_CookieVariable
MENU_Insert_PHP	[Separator]	Separator_33
MENU_Insert_PHP	Include...	MENU_Insert_PHP_Include

Table 36.4 Continued

Parent	Menu Item	Internal ID
MENU_Insert_PHP	Include Once...	MENU_Insert_PHP_IncludeOnce
MENU_Insert_PHP	Require...	MENU_Insert_PHP_Require
MENU_Insert_PHP	Require Once...	MENU_Insert_PHP_RequireOnce
MENU_Insert_PHP	[Separator]	Separator_34
MENU_Insert_PHP	Code Block	MENU_Insert_PHP_CodeBlock
MENU_Insert_PHP	Echo	MENU_Insert_PHP_Echo
MENU_Insert_PHP	Comment	MENU_Insert_PHP_Comment
MENU_Insert_PHP	If	MENU_Insert_PHP_If
MENU_Insert_PHP	Else	MENU_Insert_PHP_Else
MENU_Insert	[Separator]	Separator_35
MENU_Insert	Hyperlink...	MENU_Insert_Hyperlink
MENU_Insert	Bookmark...	MENU_Insert_Bookmark
MENU_Insert	Picture	MENU_Insert_Picture
MENU_Insert_Picture	From File...	MENU_Insert_Picture_FromFile
MENU_Insert_Picture	From Adobe Photoshop (.psd)...	MENU_Insert_Picture_FromAdobePhotoshopPSD
MENU_Insert	Media	MENU_Insert_Media
MENU_Insert_Media	Flash Movie...	MENU_Insert_Media_FlashMovie
MENU_Insert_Media	Silverlight...	MENU_Insert_Media_Silverlight
MENU_Insert_Media	Silverlight Video...	MENU_Insert_Media_SilverlightVideo
MENU_Insert_Media	Deep Zoom...	MENU_Insert_Media_DeepZoom
MENU_Insert_Media	Windows Media Player...	MENU_Insert_Media_WindowsMediaPlayer
MENU_Insert	Interactive Button...	MENU_Insert_InteractiveButton
MENU_Insert	Symbol...	MENU_Insert_Symbol

**note**

It is not possible to add a new menu item before the Mark of the Web menu item on the Insert, HTML menu.

Table 36.5 Format Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Format	MENU_Format
MENU_Format	New Style...	MENU_Format_NewStyle
MENU_Format	CSS Styles	MENU_Format_CSSStyles
MENU_Format_CSSStyles	Manage Styles...	MENU_Format_CSSStyles_ManageStyles
MENU_Format_CSSStyles	Apply Styles...	MENU_Format_CSSStyles_ApplyStyles
MENU_Format_CSSStyles	[Separator]	Separator_36
MENU_Format_CSSStyles	Attach Style Sheet...	MENU_Format_CSSStyles_AttachStyleSheet
MENU_Format_CSSStyles	Manage Style Sheet Links...	MENU_Format_CSSStyles_ManageStyleSheetLinks
MENU_Format_CSSStyles	[Separator]	Separator_37
MENU_Format_CSSStyles	Style Application	MENU_Format_CSSStyles_StyleApplication
MENU_Format	CSS Properties...	MENU_Format_CSSProperties
MENU_Format	Tag Properties...	MENU_Format_CSSStyles_TagProperties
MENU_Format	[Separator]	Separator_38
MENU_Format	Font...	MENU_Format_CSSStyles_Font
MENU_Format	Paragraph...	MENU_Format_Paragraph
MENU_Format	Bullets and Numbering...	MENU_Format_BulletsAndNumbering
MENU_Format	Borders and Shading...	MENU_Format_BordersAndShading
MENU_Format	Position...	MENU_Format_Position
MENU_Format	Behaviors...	MENU_Format_Behaviors
MENU_Format	Layers...	MENU_Format_Layers
MENU_Format	[Separator]	Separator_39
MENU_Format	Dynamic Web Template	MENU_Format_DynamicWebTemplate
MENU_Format_DynamicWebTemplate	Attach Dynamic Web Template...	MENU_Format_DynamicWebTemplate_AttachDynamicWebTemplate
MENU_Format_DynamicWebTemplate	Detach from Dynamic Web Template	MENU_Format_DynamicWebTemplate_DetachFromDynamicWebTemplate

Table 36.5 Continued

Parent	Menu Item	Internal ID
MENU_Format_DynamicWebTemplate	Open Attached Dynamic Web Template	MENU_Format_DynamicWebTemplate_OpenAttachedDynamicWebTemplate
MENU_Format_DynamicWebTemplate	[Separator]	Separator_40
MENU_Format_DynamicWebTemplate	Update Selected Page	MENU_Format_DynamicWebTemplate_UpdateSelectedPage
MENU_Format_DynamicWebTemplate	Update All Pages	MENU_Format_DynamicWebTemplate_UpdateAllPages
MENU_Format_DynamicWebTemplate	Update Attached Pages	MENU_Format_DynamicWebTemplate_UpdateAttachedPages
MENU_Format_DynamicWebTemplate	[Separator]	Separator_41
MENU_Format_DynamicWebTemplate	Manage Editable Regions...	MENU_Format_DynamicWebTemplate_ManageEditableRegions
MENU_Format	Master Page	MENU_Format_MasterPage
MENU_Format_MasterPage	Attach Master Page...	MENU_Format_MasterPage_AttachMasterPage
MENU_Format_MasterPage	Detach from Master Page	MENU_Format_MasterPage_DetachFromMasterPage
MENU_Format_MasterPage	Open Attached Master Page	MENU_Format_MasterPage_OpenAttachedMasterPage
MENU_Format_MasterPage	[Separator]	Separator_42
MENU_Format_MasterPage	Manage Content Regions...	MENU_Format_MasterPage_ManageContentRegions
MENU_Format	Frames	MENU_Format_Frames
MENU_Format_Frames	Split Frame	MENU_Format_Frames_SplitFrame
MENU_Format_Frames	Delete Frame	MENU_Format_Frames_DeleteFrame
MENU_Format_Frames	[Separator]	Separator_43
MENU_Format_Frames	Open Page in New Window	MENU_Format_Frames_OpenPageInNewWindow
MENU_Format_Frames	Save Page	MENU_Format_Frames_SavePage

Parent	Menu Item	Internal ID
MENU_Format_Frames	Save Page As...	MENU_Format_Frames_SavePageAs
MENU_Format_Frames	[Separator]	Separator_44
MENU_Format_Frames	Frame Properties...	MENU_Format_Frames_FrameProperties
MENU_Format	Background...	MENU_Format_Background
MENU_Format	Page Transition...	MENU_Format_PageTransition
MENU_Format	[Separator]	Separator_45
MENU_Format	Remove Formatting	MENU_Format_RemoveFormatting
MENU_Format	[Separator]	Separator_46
MENU_Format	Properties...	MENU_Format_Properties

Table 36.6 Tools Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Tools	MENU_Tools
MENU_Tools	Spelling	MENU_Tools_Spelling
MENU_Tools_Spelling	Spelling...	MENU_Tools_Spelling_Spelling
MENU_Tools_Spelling	Spelling Options...	MENU_Tools_Spelling_SpellingOptions
MENU_Tools	Thesaurus...	MENU_Tools_Thesaurus
MENU_Tools	Set Language...	MENU_Tools_SetLanguage
MENU_Tools	[Separator]	Separator_47
MENU_Tools	Accessibility Reports...	MENU_Tools_AccessibilityReports
MENU_Tools	Compatibility Reports...	MENU_Tools_CompatibilityReports
MENU_Tools	SEO Reports...	MENU_Tools_SeoReports
MENU_Tools	CSS Reports...	MENU_Tools_CSSReports
MENU_Tools	[Separator]	Separator_48
MENU_Tools	Add-Ins...	MENU_Tools_AddIns
MENU_Tools	[Separator]	Separator_49
MENU_Tools	Recalculate Hyperlinks...	MENU_Tools_RecalculateHyperlinks

Table 36.6 Continued

Parent	Menu Item	Internal ID
MENU_Tools	Optimize HTML...	MENU_Tools_OptimizeHTML
MENU_Tools	[Separator]	Separator_50
MENU_Tools	Application Options...	MENU_Tools_ApplicationOptions
MENU_Tools	Page Editor Options...	MENU_Tools_PageEditorOptions

Table 36.7 Table Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Table	MENU_Table
MENU_Table	Insert Table...	MENU_Table_InsertTable
MENU_Table	[Separator]	Separator_51
MENU_Table	Insert	MENU_Table_Insert
MENU_Table_Insert	Rows or Columns...	MENU_Table_Insert_RowsOrColumns
MENU_Table_Insert	[Separator]	Separator_52
MENU_Table_Insert	Column to the Left	MENU_Table_Insert_ColumnToTheLeft
MENU_Table_Insert	Column to the Right	MENU_Table_Insert_ColumnToTheRight
MENU_Table_Insert	[Separator]	Separator_53
MENU_Table_Insert	Row Above	MENU_Table_Insert_RowAbove
MENU_Table_Insert	Row Below	MENU_Table_Insert_RowBelow
MENU_Table_Insert	[Separator]	Separator_54
MENU_Table_Insert	Cell to the Left	MENU_Table_Insert_CellToTheLeft
MENU_Table_Insert	Cell to the Right	MENU_Table_Insert_CellToTheRight
MENU_Table_Insert	[Separator]	Separator_55
MENU_Table_Insert	Caption	MENU_Table_Insert_Caption
MENU_Table	Delete	MENU_Table_Delete
MENU_Table_Delete	Table	MENU_Table_Delete_Table
MENU_Table_Delete	Delete Columns	MENU_Table_Delete_DeleteColumns
MENU_Table_Delete	Delete Rows	MENU_Table_Delete_DeleteRows
MENU_Table_Delete	Delete Cells	MENU_Table_Delete_DeleteCells

Parent	Menu Item	Internal ID
MENU_Table	Select	MENU_Table_Select
MENU_Table_Select	Table	MENU_Table_Select_Table
MENU_Table_Select	Column	MENU_Table_Select_Column
MENU_Table_Select	Row	MENU_Table_Select_Row
MENU_Table_Select	Cell	MENU_Table_Select_Cell
MENU_Table	Modify	MENU_Table_Modify
MENU_Table_Modify	Merge Cells	MENU_Table_Modify_MergeCells
MENU_Table_Modify	Split Cells...	MENU_Table_Modify_SplitCells
MENU_Table_Modify	Split Table	MENU_Table_Modify_SplitTable
MENU_Table_Modify	Table AutoFormat...	MENU_Table_Modify_TableAutoFormat
MENU_Table_Modify	Distribute Rows Evenly	MENU_Table_Modify_DistributeRowsEvenly
MENU_Table_Modify	Distribute Columns Evenly	MENU_Table_Modify_DistributeColumnsEvenly
MENU_Table_Modify	AutoFit to Contents	MENU_Table_Modify_AutoFitToContents
MENU_Table	[Separator]	Separator_56
MENU_Table	Convert	MENU_Table_Convert
MENU_Table_Convert	Text to Table...	MENU_Table_Convert_TextToTable
MENU_Table_Convert	Table to Text	MENU_Table_Convert_TableToText
MENU_Table	Fill	MENU_Table_Fill
MENU_Table_Fill	Down	MENU_Table_Fill_Down
MENU_Table_Fill	Right	MENU_Table_Fill_Right
MENU_Table	[Separator]	Separator_57
MENU_Table	Table Properties	MENU_Table_TableProperties
MENU_Table_TableProperties	Table...	MENU_Table_TableProperties_Table
MENU_Table_TableProperties	Cell	MENU_Table_TableProperties_Cell
MENU_Table_TableProperties	Caption...	MENU_Table_TableProperties_Caption

Table 36.8 Site Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Site	MENU_Site
MENU_Site	New Site...	MENU_Site_NewSite
MENU_Site	Open Site...	MENU_Site_OpenSite
MENU_Site	Recent Sites	MENU_Site_RecentSites
MENU_Site	Close	MENU_Site_Close
MENU_Site	[Separator]	Separator_58
MENU_Site	Publishing	MENU_Site_Publishing
MENU_Site	Publishing Settings...	MENU_Site_PublishingSettings
MENU_Site	Publish Current File...	MENU_Site_Publish_Current
MENU_Site	Publish Changed Files...	MENU_Site_Publish_Changed
MENU_Site	Publish All Files...	MENU_Site_Publish_All
MENU_Site	[Separator]	Separator_59
MENU_Site	Delete...	MENU_Site_Delete
MENU_Site	Export to Web Package...	MENU_Site_ExportToWebPackage
MENU_Site	Import	MENU_Site_Import
MENU_Site_Import	Import from Web Package...	MENU_Site_Import_ImportFromWebPackage
MENU_Site_Import	Import Site Wizard...	MENU_Site_Import_ImportSiteWizard
MENU_Site	[Separator]	Separator_60
MENU_Site	Manage Sites List...	MENU_Site_ManageSitesList
MENU_Site	Site Settings...	MENU_Site_SiteSettings

Table 36.9 Data View Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Data View	MENU_DataView
MENU_DataView	Insert Data View...	MENU_DataView_InsertDataView
MENU_DataView	Manage Data Sources...	MENU_DataView_ManageDataSources
MENU_DataView	[Separator]	Separator_61
MENU_DataView	Filter...	MENU_DataView_Filter
MENU_DataView	Sort.../Sort and Group...	MENU_DataView_DynamicLabel1
MENU_DataView	Separator	Separator_62
MENU_DataView	Edit Columns...	MENU_DataView_EditColumns
MENU_DataView	Change Layout...	MENU_DataView_ChangeLayout
MENU_DataView	[Separator]	Separator_63
MENU_DataView	Conditional Formatting...	MENU_DataView_ConditionalFormatting
MENU_DataView	[Separator]	Separator_64
MENU_DataView	Format Item as	MENU_DataView_FormatItemAs
MENU_DataView_FormatItemAs	Text	MENU_DataView_FormatItemAs_Text
MENU_DataView_FormatItemAs	Boolean	MENU_DataView_FormatItemAs_Boolean
MENU_DataView_FormatItemAs	Number...	MENU_DataView_FormatItemAs_Number
MENU_DataView_FormatItemAs	Currency...	MENU_DataView_FormatItemAs_Currency
MENU_DataView_FormatItemAs	Date Time...	MENU_DataView_FormatItemAs_DateTime
MENU_DataView_FormatItemAs	Label	MENU_DataView_FormatItemAs_Label
MENU_DataView_FormatItemAs	[Separator]	Separator_65
MENU_DataView_FormatItemAs	Hyperlink	MENU_DataView_FormatItemAs_Hyperlink
MENU_DataView_FormatItemAs	Picture	MENU_DataView_FormatItemAs_Picture
MENU_DataView_FormatItemAs	[Separator]	Separator_66

Table 36.9 Continued

Parent	Menu Item	Internal ID
MENU_DataView_FormatItemAs	TextBox	MENU_DataView_FormatItemAs_TextBox
MENU_DataView_FormatItemAs	Multiline TextBox	MENU_DataView_FormatItemAs_MultilineTextBox
MENU_DataView_FormatItemAs	Password TextBox	MENU_DataView_FormatItemAs_PasswordTextBox
MENU_DataView_FormatItemAs	Dropdown List	MENU_DataView_FormatItemAs_DropDownList
MENU_DataView_FormatItemAs	List Box	MENU_DataView_FormatItemAs_ListBox
MENU_DataView_FormatItemAs	Date Picker	MENU_DataView_FormatItemAs_DatePicker
MENU_DataView_FormatItemAs	Check Box	MENU_DataView_FormatItemAs_CheckBox
MENU_DataView_FormatItemAs	Check Box List	MENU_DataView_FormatItemAs_CheckBoxList
MENU_DataView_FormatItemAs	Radio Button	MENU_DataView_FormatItemAs_RadioButton
MENU_DataView_FormatItemAs	Insert Formula...	MENU_DataView_FormatItemAs_InsertFormula
MENU_DataView_FormatItemAs	Edit Formula...	MENU_DataView_FormatItemAs_EditFormula
MENU_DataView_FormatItemAs	[Separator]	Separator_67
MENU_DataView_FormatItemAs	Refresh Data	MENU_DataView_FormatItemAs_RefreshData
MENU_DataView_FormatItemAs	Data View Properties...	MENU_DataView_FormatItemAs_DataViewProperties

To add a menu item to the Panels menu, create a Panel using the <panel> element in the manifest.

Table 36.10 Window Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Window	MENU_Window
MENU_Window	[Separator]	Separator_74
MENU_Window	Close All Pages	MENU_Window_CloseAllPages

Table 36.11 Help Menu Internal IDs

Parent	Menu Item	Internal ID
N/A	Help	MENU_Help
MENU_Help	User Guide	MENU_Help_ExpressionWebHelp
MENU_Help	Expression Web SDK User Guide	MENU_Help_SDKUserGuide
MENU_Help	[Separator]	Separator_74
MENU_Help	Online Community	MENU_Help_OnlineCommunity
MENU_Help_OnlineCommunity	Expression Community Home	MENU_Help_OnlineCommunity_Home
MENU_Help_OnlineCommunity	Expression Web Online Forums	MENU_Help_OnlineCommunity_OnlineForums
MENU_Help_OnlineCommunity	[Separator]	Separator_76
MENU_Help_OnlineCommunity	Learn Expression Web	MENU_Help_OnlineCommunity_LearnWeb
MENU_Help_OnlineCommunity	Add-Ins Community Gallery	MENU_Help_OnlineCommunity_Gallery
MENU_Help_OnlineCommunity	Microsoft Web Platform Installer	MENU_Help_OnlineCommunity_PlatformInstaller
MENU_Help_OnlineCommunity	[Separator]	Separator_77
MENU_Help_OnlineCommunity	Expression Web Team Blog	MENU_Help_OnlineCommunity_TeamBlog
MENU_Help	Keyboard Shortcuts	MENU_Help_KeyboardShortcuts
MENU_Help	Privacy Statement	MENU_Help_PrivacyStatement
MENU_Help	[Separator]	Separator_78
MENU_Help	Customer Experience Improvement Program	MENU_Help_CustomerExperience
MENU_Help	Register Expression Web	MENU_Help_RegisterExpressionWeb
MENU_Help	Enter Product Key	MENU_Help_EnterProductKey
MENU_Help	Activate Product	MENU_Help_ActivateProduct
MENU_Help	[Separator]	Separator_79
MENU_Help	About Microsoft Expression Web	MENU_Help_AboutMicrosoftExpressionWeb

Toolbars

The following tables list the internal IDs of all toolbars and toolbar buttons in Expression Web.

Table 36.12 shows the internal IDs on the Standard toolbar.

Table 36.13 shows the internal IDs on the Formatting toolbar.

Table 36.14 shows the internal IDs on the Code View toolbar.

Table 36.15 shows the internal IDs on the Common toolbar.

Table 36.16 shows the internal IDs on the Dynamic Web Template toolbar.

Table 36.17 shows the internal IDs on the Master Page toolbar.

Table 36.18 shows the internal IDs on the Pictures toolbar.

Table 36.19 shows the internal IDs on the Positioning toolbar.

Table 36.20 shows the internal IDs on the Style toolbar.

Table 36.21 shows the internal IDs on the Style Application toolbar.

Table 36.22 shows the internal IDs on the Tables toolbar.

Table 36.12 Standard Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Standard
COMMANDBAR_Standard	New Document	COMMANDBAR_Standard_New
COMMANDBAR_Standard	New Site...	COMMANDBAR_Standard_NewSite
COMMANDBAR_Standard	Open...	COMMANDBAR_Standard_Open
COMMANDBAR_Standard	Save	COMMANDBAR_Standard_Save
COMMANDBAR_Standard	Save All	COMMANDBAR_Standard_SaveAll
COMMANDBAR_Standard	Find...	COMMANDBAR_Standard_Find
COMMANDBAR_Standard	[Separator]	Separator_16
COMMANDBAR_Standard	Publish the current file	COMMANDBAR_Standard_Publish_Current
COMMANDBAR_Standard	Publish all changed files in your site	COMMANDBAR_Standard_Publish_Changed
COMMANDBAR_Standard	Publish all files in your site	COMMANDBAR_Standard_Publish_All
COMMANDBAR_Standard	Get all changed files in your site	COMMANDBAR_Standard_Get_Changed
COMMANDBAR_Standard	Get all files in your site	COMMANDBAR_Standard_Get_All

Parent	Toolbar Button	Internal ID
COMMANDBAR_Standard	Print	COMMANDBAR_Standard_Print
COMMANDBAR_Standard	Display in SuperPreview	COMMANDBAR_Standard_OpenInSuperPreview
COMMANDBAR_Standard	Preview in Browser	COMMANDBAR_Standard_PreviewInBrowser
COMMANDBAR_Standard	Spelling...	COMMANDBAR_Standard_Spelling
COMMANDBAR_Standard	[Separator]	Separator_19
COMMANDBAR_Standard	Cut	COMMANDBAR_Standard_Cut
COMMANDBAR_Standard	Copy	COMMANDBAR_Standard_Copy
COMMANDBAR_Standard	Paste	COMMANDBAR_Standard_Paste
COMMANDBAR_Standard	Format Painter	COMMANDBAR_Standard_FormatPainter
COMMANDBAR_Standard	[Separator]	Separator_20
COMMANDBAR_Standard	Undo	COMMANDBAR_Standard_Undo
COMMANDBAR_Standard	Redo	COMMANDBAR_Standard_Redo
COMMANDBAR_Standard	Insert Table	COMMANDBAR_Standard_InsertTable
COMMANDBAR_Standard	Insert Layer	COMMANDBAR_Standard_Layer
COMMANDBAR_Standard	Insert Picture from File	COMMANDBAR_Standard_FromFile
COMMANDBAR_Standard	[Separator]	Separator_22
COMMANDBAR_Standard	Insert Hyperlink	COMMANDBAR_Standard_Hyperlink
COMMANDBAR_Standard	Include Page...	COMMANDBAR_Standard_IncludePage
COMMANDBAR_Standard	[Separator]	Separator_23
COMMANDBAR_Standard	Refresh	COMMANDBAR_Standard_Refresh
COMMANDBAR_Standard	Stop	COMMANDBAR_Standard_Stop
COMMANDBAR_Standard	[Separator]	Separator_24
COMMANDBAR_Standard	Show formatting marks	COMMANDBAR_Standard_ShowFormattingMarks
COMMANDBAR_Standard	[Separator]	Separator_25
COMMANDBAR_Standard	Expression Web Help	COMMANDBAR_Standard_ExpressionWebHelp

Table 36.13 Formatting Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Formatting
COMMANDBAR_Formatting	Manage Styles...	COMMANDBAR_Formatting_ManageStyles
COMMANDBAR_Formatting	Style:	COMMANDBAR_Formatting_Style
COMMANDBAR_Formatting	Font:	COMMANDBAR_Formatting_Font
COMMANDBAR_Formatting	Font Size:	COMMANDBAR_Formatting_FontSize
COMMANDBAR_Formatting	[Separator]	Separator_26
COMMANDBAR_Formatting	Bold	COMMANDBAR_Formatting_Bold
COMMANDBAR_Formatting	Italic	COMMANDBAR_Formatting_Italic
COMMANDBAR_Formatting	Underline	COMMANDBAR_Formatting_Underline
COMMANDBAR_Formatting	[Separator]	Separator_27
COMMANDBAR_Formatting	Align Text Left	COMMANDBAR_Formatting_AlignLeft
COMMANDBAR_Formatting	Center	COMMANDBAR_Formatting_Center
COMMANDBAR_Formatting	Align Text Right	COMMANDBAR_Formatting_AlignRight
COMMANDBAR_Formatting	Justify	COMMANDBAR_Formatting_Justify
COMMANDBAR_Formatting	[Separator]	Separator_28
COMMANDBAR_Formatting	Increase Font Size	COMMANDBAR_Formatting_IncreaseFontSize
COMMANDBAR_Formatting	Decrease Font Size	COMMANDBAR_Formatting_DecreaseFontSize
COMMANDBAR_Formatting	[Separator]	Separator_29
COMMANDBAR_Formatting	Numbering	COMMANDBAR_Formatting_Numbering
COMMANDBAR_Formatting	Bullets	COMMANDBAR_Formatting_Bullets
COMMANDBAR_Formatting	Decrease Indent Position	COMMANDBAR_Formatting_DecreaseIndent
COMMANDBAR_Formatting	Increase Indent Position	COMMANDBAR_Formatting_IncreaseIndent
COMMANDBAR_Formatting	[Separator]	Separator_30
COMMANDBAR_Formatting	Borders	COMMANDBAR_Formatting_Borders
COMMANDBAR_Formatting	Highlight	COMMANDBAR_Formatting_Highlight
COMMANDBAR_Formatting	Font Color	COMMANDBAR_Formatting_FontColor
COMMANDBAR_Formatting	[Separator]	Separator_31

Table 36.14 Code View Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_CodeView
COMMANDBAR_CodeView	List Members	COMMANDBAR_CodeView_ListMembers
COMMANDBAR_CodeView	Parameter Info	COMMANDBAR_CodeView_ParameterInfo
COMMANDBAR_CodeView	Complete Word	COMMANDBAR_CodeView_CompleteWord
COMMANDBAR_CodeView	List Code Snippets	COMMANDBAR_CodeView_ListCodeSnippets
COMMANDBAR_CodeView	[Separator]	Separator_49
COMMANDBAR_CodeView	Follow Code Hyperlink	COMMANDBAR_CodeView_FollowCodeHyperlink
COMMANDBAR_CodeView	Previous Code Hyperlink	COMMANDBAR_CodeView_PreviousCodeHyperlink
COMMANDBAR_CodeView	Next Code Hyperlink	COMMANDBAR_CodeView_NextCodeHyperlink
COMMANDBAR_CodeView	[Separator]	Separator_44
COMMANDBAR_CodeView	Function Lookup	COMMANDBAR_CodeView_FunctionLookup
COMMANDBAR_CodeView	[Separator]	Separator_45
COMMANDBAR_CodeView	Toggle Bookmark	COMMANDBAR_CodeView_ToggleBookmark
COMMANDBAR_CodeView	Next Bookmark	COMMANDBAR_CodeView_NextBookmark
COMMANDBAR_CodeView	Previous Bookmark	COMMANDBAR_CodeView_PreviousBookmark
COMMANDBAR_CodeView	Clear Bookmarks	COMMANDBAR_CodeView_ClearBookmarks
COMMANDBAR_CodeView	[Separator]	Separator_46
COMMANDBAR_CodeView	Select Tag	COMMANDBAR_CodeView_SelectTag
COMMANDBAR_CodeView	Find Matching Tag	COMMANDBAR_CodeView_FindMatchingTag
COMMANDBAR_CodeView	Select Block	COMMANDBAR_CodeView_SelectBlock
COMMANDBAR_CodeView	Find Matching Brace	COMMANDBAR_CodeView_FindMatchingBrace
COMMANDBAR_CodeView	[Separator]	Separator_47

Table 36.14 Continued

Parent	Toolbar Button	Internal ID
COMMANDBAR_CodeView	Insert Start Tag	COMMANDBAR_CodeView_InsertStartTag
COMMANDBAR_CodeView	Insert End Tag	COMMANDBAR_CodeView_InsertEndTag
COMMANDBAR_CodeView	Insert HTML Comment	COMMANDBAR_CodeView_InsertHTMLComment
COMMANDBAR_CodeView	Insert CSS Comment	COMMANDBAR_CodeView_InsertCSSComment
COMMANDBAR_CodeView	[Separator]	Separator_48
COMMANDBAR_CodeView	Options	COMMANDBAR_CodeView_Options

Table 36.15 Common Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Common
COMMANDBAR_Common	New Document	COMMANDBAR_Common_New
COMMANDBAR_Common	New Site...	COMMANDBAR_Common_NewSite
COMMANDBAR_Common	Open...	COMMANDBAR_Common_Open
COMMANDBAR_Common	Save	COMMANDBAR_Common_Save
COMMANDBAR_Common	Save All	COMMANDBAR_Common_SaveAll
COMMANDBAR_Common	Display in SuperPreview	COMMANDBAR_Common_OpenInSuperPreview
COMMANDBAR_Common	Preview in Browser	COMMANDBAR_Common_PreviewInBrowser
COMMANDBAR_Common	[Separator]	Separator_6
COMMANDBAR_Common	Style:	COMMANDBAR_Common_Style
COMMANDBAR_Common	Font:	COMMANDBAR_Common_Font
COMMANDBAR_Common	Font Size:	COMMANDBAR_Common_FontSize
COMMANDBAR_Common	[Separator]	Separator_7
COMMANDBAR_Common	Undo	COMMANDBAR_Common_Undo
COMMANDBAR_Common	Redo	COMMANDBAR_Common_Redo
COMMANDBAR_Common	[Separator]	Separator_8
COMMANDBAR_Common	Bold	COMMANDBAR_Common_Bold
COMMANDBAR_Common	Italic	COMMANDBAR_Common_Italic

Parent	Toolbar Button	Internal ID
COMMANDBAR_Common	Underline	COMMANDBAR_Common_Underline
COMMANDBAR_Common	[Separator]	Separator_9
COMMANDBAR_Common	Align Text Left	COMMANDBAR_Common_AlignLeft
COMMANDBAR_Common	Center	COMMANDBAR_Common_Center
COMMANDBAR_Common	Align Text Right	COMMANDBAR_Common_AlignRight
COMMANDBAR_Common	[Separator]	Separator_10
COMMANDBAR_Common	Numbering	COMMANDBAR_Common_Numbering
COMMANDBAR_Common	Bullets	COMMANDBAR_Common_Bullets
COMMANDBAR_Common	Decrease Indent Position	COMMANDBAR_Common_DecreaseIndent
COMMANDBAR_Common	Increase Indent Position	COMMANDBAR_Common_IncreaseIndent
COMMANDBAR_Common	[Separator]	Separator_11
COMMANDBAR_Common	Borders	COMMANDBAR_Common_Borders
COMMANDBAR_Common	Highlight	COMMANDBAR_Common_Highlight
COMMANDBAR_Common	Font Color	COMMANDBAR_Common_FontColor
COMMANDBAR_Common	[Separator]	Separator_12
COMMANDBAR_Common	Insert Table	COMMANDBAR_Common_InsertTable
COMMANDBAR_Common	<div>	COMMANDBAR_Common_Div
COMMANDBAR_Common	Insert Picture from File	COMMANDBAR_Common_FromFile
COMMANDBAR_Common	Insert Hyperlink	COMMANDBAR_Common_Hyperlink
COMMANDBAR_Common	Stop	COMMANDBAR_Common_Stop

Table 36.16 Dynamic Web Template Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_DynamicWebTemplate
COMMANDBAR_DynamicWebTemplate	Regions:	COMMANDBAR_DynamicWebTemplate_Regions
COMMANDBAR_DynamicWebTemplate	Manage Editable Regions...	COMMANDBAR_DynamicWebTemplate_ManageEditableRegions

Table 36.16 Continued

Parent	Toolbar Button	Internal ID
COMMANDBAR_DynamicWebTemplate	Update Attached Pages	COMMANDBAR_DynamicWebTemplate_UpdateAttachedPages
COMMANDBAR_DynamicWebTemplate	Template Region Labels	COMMANDBAR_DynamicWebTemplate_TemplateRegionLabels

Table 36.17 Master Page Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_MasterPage
COMMANDBAR_MasterPage	Regions	COMMANDBAR_MasterPage_Regions
COMMANDBAR_MasterPage	Manage Content Regions	COMMANDBAR_MasterPage_ManageContentRegions
COMMANDBAR_MasterPage	Template Region Labels	COMMANDBAR_MasterPage_TemplateRegionLabels

Table 36.18 Pictures Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Pictures
COMMANDBAR_Pictures	Insert Picture from File	COMMANDBAR_Pictures_FromFile
COMMANDBAR_Pictures	[Separator]	Separator_32
COMMANDBAR_Pictures	Auto Thumbnail	COMMANDBAR_Pictures_AutoThumbnail
COMMANDBAR_Pictures	[Separator]	Separator_33
COMMANDBAR_Pictures	Bring Forward	COMMANDBAR_Pictures_BringForward
COMMANDBAR_Pictures	Send Backward	COMMANDBAR_Pictures_SendBackward
COMMANDBAR_Pictures	[Separator]	Separator_34
COMMANDBAR_Pictures	Rotate Left 90°	COMMANDBAR_Pictures_RotateLeft90°
COMMANDBAR_Pictures	Rotate Right 90°	COMMANDBAR_Pictures_RotateRight90°

Parent	Toolbar Button	Internal ID
COMMANDBAR_Pictures	Flip Horizontal	COMMANDBAR_Pictures_FlipHorizontal
COMMANDBAR_Pictures	Flip Vertical	COMMANDBAR_Pictures_FlipVertical
COMMANDBAR_Pictures	[Separator]	Separator_35
COMMANDBAR_Pictures	More Contrast	COMMANDBAR_Pictures_MoreContrast
COMMANDBAR_Pictures	Less Contrast	COMMANDBAR_Pictures_LessContrast
COMMANDBAR_Pictures	More Brightness	COMMANDBAR_Pictures_MoreBrightness
COMMANDBAR_Pictures	Less Brightness	COMMANDBAR_Pictures_LessBrightness
COMMANDBAR_Pictures	[Separator]	Separator_36
COMMANDBAR_Pictures	Crop	COMMANDBAR_Pictures_Crop
COMMANDBAR_Pictures	Set Transparent Color	COMMANDBAR_Pictures_SetTransparentColor
COMMANDBAR_Pictures	Bevel	COMMANDBAR_Pictures_Bevel
COMMANDBAR_Pictures	Resample	COMMANDBAR_Pictures_Resample
COMMANDBAR_Pictures	[Separator]	Separator_37
COMMANDBAR_Pictures	Select	COMMANDBAR_Pictures_Select
COMMANDBAR_Pictures	Rectangular Hotspot	COMMANDBAR_Pictures_RectangularHotspot
COMMANDBAR_Pictures	Circular Hotspot	COMMANDBAR_Pictures_CircularHotspot
COMMANDBAR_Pictures	Polygonal Hotspot	COMMANDBAR_Pictures_PolygonalHotspot
COMMANDBAR_Pictures	Highlight Hotspots	COMMANDBAR_Pictures_HighlightHotspots
COMMANDBAR_Pictures	[Separator]	Separator_38
COMMANDBAR_Pictures	Restore	COMMANDBAR_Pictures_Restore

Table 36.19 Positioning Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Positioning
COMMANDBAR_Positioning	Edit the position property	COMMANDBAR_Positioning_Position
COMMANDBAR_Positioning	Left	LABEL_COMMANDBAR_Positioning_Left
COMMANDBAR_Positioning	Top	COMMANDBAR_Positioning_Left
COMMANDBAR_Positioning	Top	LABEL_COMMANDBAR_Positioning_Top
COMMANDBAR_Positioning	Right	LABEL_COMMANDBAR_Positioning_Right
COMMANDBAR_Positioning	Bottom	LABEL_COMMANDBAR_Positioning_Bottom
COMMANDBAR_Positioning	Width	LABEL_COMMANDBAR_Positioning_Width
COMMANDBAR_Positioning	Height	LABEL_COMMANDBAR_Positioning_Height
COMMANDBAR_Positioning	Z-Index	LABEL_COMMANDBAR_Positioning_Zindex
COMMANDBAR_Positioning	Bring Forward	COMMANDBAR_Positioning_BringForward
COMMANDBAR_Positioning	Send Backward	COMMANDBAR_Positioning_SendBackward

Table 36.20 Style Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Style
COMMANDBAR_Style	Class:	LABEL_COMMANDBAR_Style_Class
COMMANDBAR_Style	ID:	LABEL_COMMANDBAR_Style_ID
COMMANDBAR_Style	New Style...	COMMANDBAR_Style_NewStyle
COMMANDBAR_Style	Attach Style Sheet...	COMMANDBAR_Style_AttachStyleSheet

Table 36.21 Style Application Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_StyleApplication
COMMANDBAR_StyleApplication	Target Rule:	LABEL_COMMANDBAR_StyleApplication_TargetRule
COMMANDBAR_StyleApplication	Reuse Properties	COMMANDBAR_StyleApplication_ReuseProperties
COMMANDBAR_StyleApplication	Show Overlay	COMMANDBAR_StyleApplication_ShowOverlay

Table 36.22 Tables Toolbar Internal IDs

Parent	Toolbar Button	Internal ID
N/A	N/A	COMMANDBAR_Tables
COMMANDBAR_Tables	Column to the Left	COMMANDBAR_Tables_ColumnToTheLeft
COMMANDBAR_Tables	Column to the Right	COMMANDBAR_Tables_ColumnToTheRight
COMMANDBAR_Tables	Row Above	COMMANDBAR_Tables_RowAbove
COMMANDBAR_Tables	Row Below	COMMANDBAR_Tables_RowBelow
COMMANDBAR_Tables	Delete Cells	COMMANDBAR_Tables_DeleteCells
COMMANDBAR_Tables	[Separator]	Separator_39
COMMANDBAR_Tables	Merge Cells	COMMANDBAR_Tables_MergeCells
COMMANDBAR_Tables	Split Cells...	COMMANDBAR_Tables_SplitCells
COMMANDBAR_Tables	[Separator]	Separator_40
COMMANDBAR_Tables	Align Top	COMMANDBAR_Tables_AlignTop
COMMANDBAR_Tables	Center Vertically	COMMANDBAR_Tables_CenterVertically
COMMANDBAR_Tables	Align Bottom	COMMANDBAR_Tables_AlignBottom
COMMANDBAR_Tables	[Separator]	Separator_41
COMMANDBAR_Tables	Distribute Rows Evenly	COMMANDBAR_Tables_DistributeRowsEvenly
COMMANDBAR_Tables	Distribute Columns Evenly	COMMANDBAR_Tables_DistributeColumnsEvenly
COMMANDBAR_Tables	AutoFit to Contents	COMMANDBAR_Tables_AutoFitToContents
COMMANDBAR_Tables	[Separator]	Separator_42
COMMANDBAR_Tables	Fill Color	COMMANDBAR_Tables_FillColor
COMMANDBAR_Tables	[Separator]	Separator_43
COMMANDBAR_Tables	Table AutoFormat Combo	COMMANDBAR_Tables_TableAutoFormatCombo
COMMANDBAR_Tables	Table AutoFormat	COMMANDBAR_Tables_TableAutoFormat

This page intentionally left blank

CREATING AND MANIPULATING AN ADD-IN USER INTERFACE

Planning an Add-in

In Chapter 36, “Expression Web 4 Add-in Basics,” you learned the details of the Expression Web add-in manifest. You learned how to create panels, menu items, and toolbar items. In this chapter, you’ll put that knowledge to use and create an add-in with all of those elements.

In this chapter, you’ll create an add-in that shows you various properties about the document currently open in Expression Web. Information is displayed in a panel, but the add-in also creates a menu option on the Tools menu that opens an options dialog for setting options for the add-in.

You develop this add-in in five steps.

1. Create the manifest with the Add-in Builder.
2. Create the user interface for the main interface as a panel.
3. Create the user interface for the options dialog.
4. Develop the JavaScript code for the panel.
5. Develop the JavaScript code for the options dialog.

The first step in creating any add-in is to create the manifest.



note

The add-in you create in this chapter is designed for use with disk-based sites only.

Creating the Manifest with the Add-in Builder

The easiest way to create an add-in manifest is to use the Add-in Builder, an add-in that Microsoft wrote for the purpose of creating add-ins easily without messing with XML code.

Download the Add-in Builder from <http://gallery.expression.microsoft.com/en-us/AddinBuilder>. The file that you download is a `.xadd` file, and it doesn't matter where you save it. Once you've successfully downloaded the Add-in Builder, follow these steps to install it:

1. Launch Expression Web.
2. Select Tools, Add-Ins.
3. Click the Install button.
4. Browse to the location where you saved the `AddinBuilder.xadd` file, select it, and click Open.
5. In the Add-in Install Complete dialog, click Yes to activate the add-in.

What Happens When an Add-in Is Installed?

In Chapter 36, I explained that a `.xadd` file is simply a Zip file with a `.xadd` file extension. When you install an add-in, Expression Web unzips the `.xadd` file and places its contents in the `%userprofile%\AppData\Roaming\Microsoft\Expression\Web 4\Addins` folder. Expression Web looks for an add-in manifest in any folder within this folder, and when it finds one, it uses it to load the add-in. You can actually install an add-in manually by simply renaming the `.xadd` extension to `.zip` and then unzipping the file into the Addins folder. No other action is necessary to install an add-in.

When you install an add-in using the Add-Ins menu in Expression Web, Expression Web also adds a text file with the name `installedByXWeb.txt`. When this file is in the add-in's folder, Expression Web enables the Remove button in the Manage Add-ins dialog. If you distribute your add-in using some kind of automated installation method, make sure that you include a text file called `installedByXWeb.txt` (the text file doesn't need to include any text content) in the add-in's folder so that users can remove your add-in from the Manage Add-ins dialog in Expression Web.

The add-in you will create has a panel and a dialog. The dialog is launched using a menu item on the Tools menu or a toolbar button on the Common toolbar. Let's use the Add-in Builder to create a manifest for the add-in.

Creating the Manifest

To create a manifest with the Add-in Builder, launch Expression Web. If a site is already open in Expression Web, first close it. The Add-in Builder isn't designed to work with an existing site. Instead, it creates a new site in the Add-ins folder that contains the files necessary for your add-in.

Select Tools, Add-in Builder to launch the Add-in Builder. The top portion of the Add-in Builder dialog contains general add-in settings such as Name, Description, and Author. The bottom portion of the dialog contains three tabs: Panels, Dialogs, and Assemblies.

In this section, I cover the Panels and Dialogs tabs. The Assemblies tab is used to load a managed assembly so that you can access it using JavaScript. I cover doing that later.

➔ *For more information on accessing a managed assembly using JavaScript, see “Accessing Managed Classes from JavaScript,” p. 682, later in this chapter.*

Start by filling in the general information about your add-in.

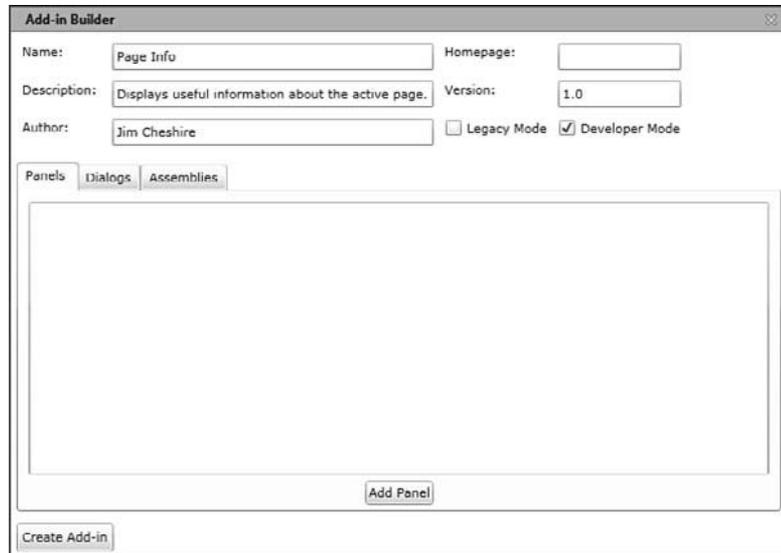
1. Enter **Page Info** in the Name box.
2. Enter a description in the Description box. The description should be brief.
3. Enter your name in the Author box.
4. Enter **1.0** in the Version box.
5. Check the Developer Mode box.

The Add-in Builder dialog should now look like Figure 37.1.



The Add-in Builder highlights fields in red when they are required.

Figure 37.1
The Add-in Builder with the general information for our add-in filled in.



Now you need to add the panel that serves as the main interface for your add-in. To do that, follow these steps:

1. Make sure that the Panels tab is selected and click Add Panel.
2. Enter **Page Info** in the Name box.
3. Enter `panel.htm` in the Source box.
4. Enter `MainUI` in the Id box.
5. Check the HTML-DOM check box.
6. Enter `ActivatePanel()` in the Activate box.
7. Enter `ClearPanel()` in the Deactivate box.

The Define a Panel dialog should look like the one shown in Figure 37.2. Click Insert to add the panel to your manifest.

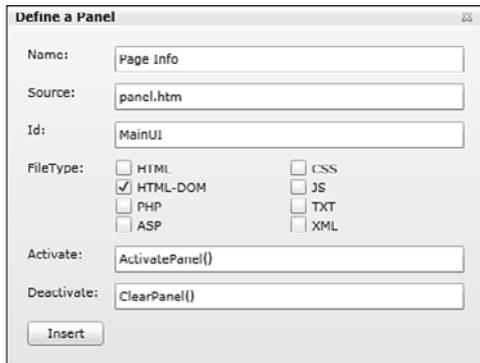


Figure 37.2

The completed Define a Panel dialog with all the properties entered for the main interface.

Don't worry about the fact that the `panel.htm` file doesn't currently exist. When you click Create Add-in, the Add-in Builder automatically creates a blank HTML file named `panel.htm`. The `ActivatePanel` and `ClearPanel` entries you made are JavaScript functions that you'll write later in this chapter.

Now you need to add a dialog for the options dialog so that users can configure your add-in. To do that, follow these steps:

1. Click the Dialogs tab and click Add Dialog.
2. Enter **Page Info Options** in the Title box.
3. Enter `options.htm` in the Source box.

caution

You can't edit an add-in with the Add-in Builder, so don't click the Create Add-in button until you have finished entering all of the information for your add-in. If you do, you'll have to manually edit the manifest to add any additional features such as a panel, a dialog, or menu or toolbar items.

tip

Remember that panels automatically have a menu item added to the Panels menu in Expression Web. Therefore, there's no need to create a menu item for a panel, and in fact, the Expression Web manifest doesn't even provide a means for doing so.

4. Enter `pageInfoOptions` in the Id box.
5. Enter **125** in the Height box.
6. Enter **300** in the Width box.
7. Leave all of the File Type check boxes unchecked.

The Define a Modal Dialog dialog should now look like the one shown in Figure 37.3.

**tip**

By leaving all of the File Type check boxes unchecked, the Page Info Options menu item will always be available to the user.

Figure 37.3

The Define a Modal Dialog dialog now contains the settings for the options dialog.

Define a Modal Dialog

Title:

Source:

Id:

Height:

Width:

File Type:

<input type="checkbox"/> HTML	<input type="checkbox"/> CSS
<input type="checkbox"/> HTML-DOM	<input type="checkbox"/> JS
<input type="checkbox"/> PHP	<input type="checkbox"/> TXT
<input type="checkbox"/> ASP	<input type="checkbox"/> XML

Resizable Scrollable

Finally, you need to create the menu item for the options dialog.

1. Click the Add Menu Item button.
2. Select **&Tools** from the Parent Menu drop-down.
3. Leave the Before box empty so that the menu item will be placed at the bottom of the Tools menu.
4. Enter **_Page Info Options...** in the Label box.
5. Enter **Page Info Options** in the Tooltip box.

The Define a Menu dialog should now look like the one shown in Figure 37.4. Click the Create button to add the menu item.

Now click Insert to add the dialog to the manifest. Once you've done that, click the Create Add-in button to create the manifest. When you do, the Add-in Builder creates a new site in the Add-ins folder. That site contains all the files for your add-in as shown in Figure 37.5.

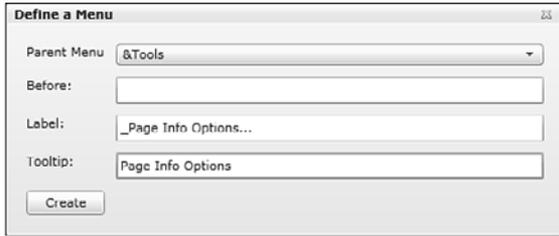


Figure 37.4
The Define a Menu dialog with all the settings for the menu item.

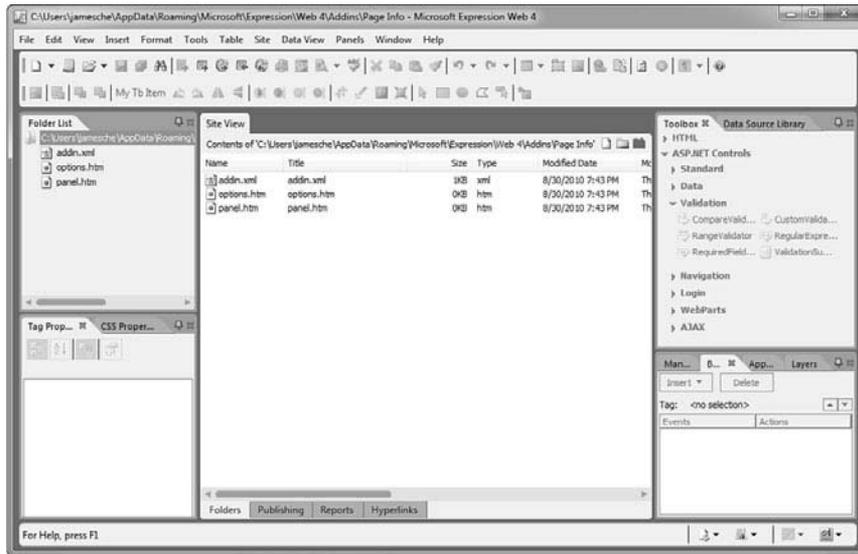


Figure 37.5
The Add-in Builder creates all the files necessary for your add-in.

At this point, close Expression Web and then launch it again. Click on the File menu, and you should see your add-in's menu option at the bottom of the menu. Why is the menu item on the File menu when you chose to place it on the Tools menu in the Add-in Builder? There's a bug in the Add-in Builder that causes it to always add menu items to the File menu regardless of which menu you choose in the Parent Menu drop-down, and to fix that, we need to edit the manifest.

**tip**

If you select Tools, Add-Ins, you see your add-in listed. However, the Remove button is not enabled for your add-in. To enable the Remove button, create an empty text file called `installedByXWeb.txt` in your add-in's folder.

Editing the Manifest

At this point, the Page Info site should be open in Expression Web. If it's not, open it from %user-profile%\AppData\Roaming\Microsoft\Expression\Web 4\Addins\Page Info. Double-click on the manifest (addin.xml) to open it and look for the following line:

```
<menuitem label="_Page Info Options..." parent="MENU_File" tooltip="Page Info Options" />
```

The parent attribute identifies the menu on which the menu item should appear. The parent attribute is currently MENU_File. Change it to MENU_Tools.

We also need to add the attributes to specify the size of the Options dialog. We entered these properties in the Add-in Builder, but there's another bug in the Add-in Builder that prevents them from being added to the manifest.

Locate the following line in the manifest:

```
<command id="pageInfoOptions"
onclick="xweb.application.showModalDialog('options.htm',
'Page Info Options');">
```

Change this line to read as follows:

```
<command id="pageInfoOptions"
onclick="xweb.application.showModalDialog('options.htm',
'Page Info Options', 'dialogHeight:125;dialogWidth:300;scroll:no');">
```

After making that change, close and save the manifest. Close Expression Web and launch it again. When it reopens, you see that the menu item now correctly appears on the Tools menu where it should be, and if you select the Page Info Options menu item, it launches the Page Info Options dialog.

At this point, both the Page Info panel and the Page Info Options dialog are blank. In the next section, I show you how to create the user interface for the panel and the dialog.



note

It's important to realize that when you are developing an add-in in Expression Web, you typically want to open the add-in's folder as a site so that you can work on the site. That's why the Add-in Builder opens the add-in's folder as a site.

Creating the User Interfaces

Add-in user interfaces are built using HTML elements. You can use any of the tools in the HTML section of the toolbox when creating your user interfaces. You can even use multimedia components in your user interface.

In this section, you'll create the user interface for the Page Info panel and for the Page Info Options dialog.

Creating a Custom Page Size for Panels

As I mentioned in the Panels section in Chapter 36, panels are a suitable interface for your add-in as long as the interface can fit within a space of 275 pixels in width. To better visualize my panel's interface, I like to create a custom page size in Expression Web of 275 pixels by 600 pixels. I can then use this custom page size to easily design my panel's interface.

To create a custom page size for a panel, follow these steps:

1. Make sure that a page is open in Design view in Expression Web.
2. Select View, Page Size, Modify Page Sizes.
3. In the Modify Page Sizes dialog, click the Add button.
4. Enter **275** in the Width box and **600** in the Height box.
5. Enter **Panel Add-In** in the Description box.
6. Click OK to add the new page size.
7. Click OK to return to Design view in Expression Web.

You can now select the new page size using the View, Page Size menu in Expression Web or by clicking on the page size indicator on the Status Bar as shown in Figure 37.6.

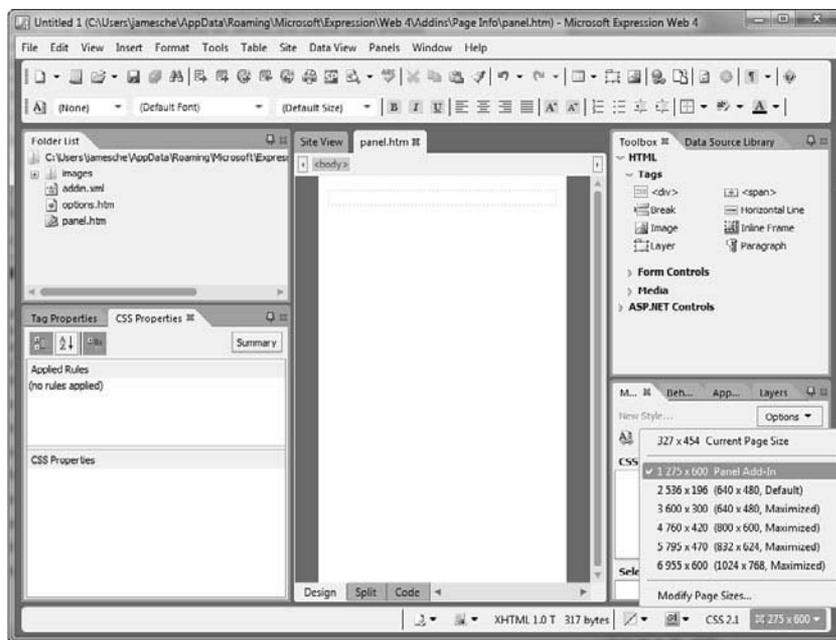


Figure 37.6
Creating a custom page size helps you to better visualize your panel's interface.

Creating the Panel's Interface

The user interface for panels and dialog boxes is created using HTML. However, creating an add-in interface presents challenges that you often don't encounter when designing web pages. Unlike pages of a website, panels and dialog boxes are designed using a single page. Therefore, if you need to design an interface that has multiple states, you need to take that into account as you design your interface.

The panel interface for the Page Info add-in has three states:

- **State 1**—No pages are open in Expression Web.
- **State 2**—An unsaved page is open in Expression Web.
- **State 3**—A saved page is open in Expression Web.

Depending on which state exists, a different interface needs to be displayed to the user.

As a general rule, I use a `<div>` to define each state of my add-in. I can then use CSS rules to set the display property of each `<div>` so that I can control which state is visible to the user. However, for the Panel Info panel, I used two `<div>` elements, one for state 1 and 2 and a second for state 3.

To create the interface for the panel, open `panel.htm` in Expression Web. Switch to Code View and add the HTML code shown in Listing 37.1.

Listing 37.1 HTML Code for the Panel Interface

```
<html>
<head>
<meta content="en-us" http-equiv="Content-Language">
<style type="text/css">
p {
    font-size: .7em;
    font-family: Arial, Helvetica, sans-serif;
    color: #808080;
}

.centered {
    text-align: center;
    padding-top: 25px;
}

#pageInfo {
    overflow: hidden;
}

#panelHeader {
    background-color: #C0C0C0;
}
</style>
</head>
```

```
<body>
  <div id="blank" class="centered">
    <p>Open a page to display properties.</p>
  </div>
  <div id="pageInfo">
    <div id="panelHeader">
      <p style="color: black">
        <strong>Current Page:</strong><span id="pageUr1">
&nbsp;</span>
      </p>
    </div>
    <p>
      <strong>Title:</strong><span id="pageTitle">&nbsp;</span>
      <br><br>
      <strong>Created:</strong><span id="pageCreated">&nbsp;</span>
      <br><strong>Modified:</strong><span id="pageModified">&nbsp;</span>
    </p>
    <p>
      <strong>Size:</strong><span id="fileSize">&nbsp;</span>&nbsp;<span id="fileSize">&nbsp;</span>bytes
    </p>
    <div id="titleWarning">
      <hr size="1">
      <p style="color: black; margin-top: -4px">
        <p style="color: red; margin-top: -4px; background-color: white">
          Title is undefined.
        </p>
      </div>
    </div>
  </body>
</html>
```

After adding this code to `panel.htm`, the page should look like the one shown in Figure 37.7.

There are two top-level `div` elements in `panel.htm`, one with an ID of "blank" and another with an ID of "pageInfo". The "blank" `div` displays a message telling the user to open a page to display properties in the panel while the "pageInfo" `div` contains all of the page elements used to display page properties. Each property is identified by a `span`, and each `span` contains an ID attribute so that I can set the text for the `span` using JavaScript code.

Save `panel.htm` and then select Panels, Page Info to see what the panel looks like in its current state. Notice that both the "blank" `div` and the "pageInfo" `div` are visible as shown in Figure 37.8.

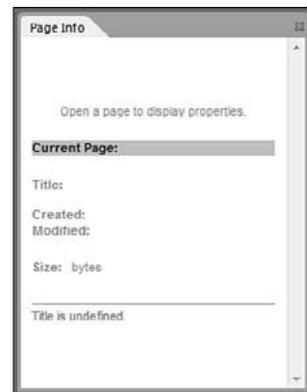
 **note**

You may have noticed that there's no DOCTYPE specified in `panel.htm`. You can specify a DOCTYPE if you want to, but since this page will only be viewed from within a panel in Expression Web, it's not necessary to use one.

Figure 37-7
The completed user interface for the Page Info panel.



Figure 37-8
Both the blank and pageInfo div are visible at this point. We'll fix that soon with some JavaScript.



Creating the Options Dialog Interface

The next step in creating the Page Info add-in is to create the interface for the Options dialog. The Options dialog allows a user to specify whether to display a warning when a title isn't defined for the current page.

Open `options.htm`, switch to Code View, and add the code in Listing 37.2 to the page.

Listing 37.2 HTML Code for `options.htm`

```
<html>
<head>
<style type="text/css">
  p {
    font-family: Arial, Helvetica, sans-serif;
    font-size: .75em;
  }
  input {
    font-family: Arial, Helvetica, sans-serif;
    font-size: .85em;
  }
}
</style>
</head>
<body onload="SetCheckboxByAppSettings();">
<p>
<input name="WarnCheckbox" type="checkbox" checked="true">
  &nbsp;Display warning when <title> not detected.
</p>
<p style="text-align: right;">
<input name="OK" type="button" value="OK" style="width: 60px"
  onclick="SetWarningOption();">
</p>
</body>
</html>
```

Notice that the OK button is configured to call the `SetWarningOption` JavaScript function when it's clicked. You'll add that function in the next section. You'll also add the `SetCheckboxByAppSettings` function that is called in the `onload` event of the page.

Save `options.htm` and close it. You can now select Tools, Page Info Options, and the Page Info Options dialog loads. The Page Info Options dialog is shown in Figure 37.9.

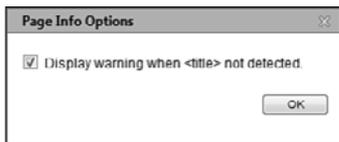


Figure 37.9
The Page Info Options dialog.

Now it's time to write some JavaScript code to implement the functionality for our add-in.

Adding Functionality with JavaScript

There are two facets to adding functionality to an add-in with JavaScript. One is writing code that interacts with elements in your user interface, typically to change the interface based on certain conditions. The other facet is writing code that interacts with the Expression Web interface itself.

To interact with elements in your user interface, the `window` and `document` objects from the browser's *document object model* (DOM) are used. The `window` object refers to the window for your interface, a panel or a dialog box depending on the type of interface you're using. The `document` object refers to the page displayed within the window.

To interact with the Expression Web interface, Microsoft developed a completely new *application programming interface* (API) that makes it easy to interact with the Expression Web application itself and with the document and site that is open within Expression Web.

➔ *For a complete reference on the Expression Web JavaScript API for developing add-ins, see Chapter 39, "Expression Web 4 JavaScript API Reference."*

Let's add some JavaScript code to the add-in's pages that controls the functionality of the add-in.

JavaScript for panel1.htm

Let's review the `<panel>` element from the Page Info add-in manifest.

```
<panel src="panel.htm" id="MainUI" title="Page Info" filetype="HTML-DOM"
activate="ActivatePanel()" deactivate="ClearPanel()" />
```

Remember from the Chapter 36 that the `activate` attribute specifies a JavaScript function that runs when the panel is activated, and the `deactivate` attribute specifies a JavaScript function that runs with the panel is deactivated. In this section, you'll write the `ActivatePanel` and `ClearPanel` JavaScript functions. I cover the `ClearPanel` function first because it's a simpler function.



note

If you click a button in the Page Info Options dialog, you'll get a script error because the JavaScript code that runs when the button is clicked hasn't been added to the page yet.



tip

If you've ever written JavaScript when designing a site, you've certainly used the `window` and `document` objects before. The same techniques used when writing JavaScript code for a site are used when writing code to interact with an add-in user interface.



note

Throughout the rest of this chapter, I often refer you to Chapter 39 for more information on the JavaScript API in Expression Web. This chapter is meant to familiarize you with the concepts needed to develop add-ins and is not meant as a reference for the JavaScript functions.



tip

The JavaScript functions for each page can be placed anywhere on the page within a script block. If you need to see an example of the scripts placed in the page, you can download the completed add-in from www.informit.com/register.

The `ClearPanel` function changes the panel's interface to display an informational message asking the user to open a page to view page information. Listing 37.3 contains the code for the `ClearPanel` function.

Listing 37.3 The `ClearPanel` Function

```
function ClearPanel() {
    var blank = window.document.getElementById("blank");
    var ui = window.document.getElementById("pageInfo");
    var titleWarning = window.document.getElementById("titleWarning");

    blank.innerHTML = "<p>Open a web page to display properties.</p>";
    blank.style.display = "block";
    ui.style.display = "none";
    titleWarning.style.display = "none";
}
```

The first three lines of this function create three variables. Each variable refers to a particular `div` in `panel.htm`. To get a reference to a particular `div`, the `ClearPanel` function uses the `getElementById` JavaScript function and uses the ID of the `div` as an argument to that function. Consider the following `div` from `panel.htm`:

```
<div id="blank" class="centered">
```

To obtain a reference to this `div` in your JavaScript code, you would use the following code:

```
window.document.getElementById("blank");
```

The next line of code sets the `innerHTML` property of the "blank" `div` so that it displays an information message telling the user to open a web page to display properties. Here is that line of code:

```
blank.innerHTML = "<p>Open a web page to display properties.</p>";
```

I've used the `innerHTML` property because the value I'm using contains HTML tags. The `innerHTML` property adds the HTML code that I've specified inside the `div` referenced by "blank". After setting the `innerHTML` property, the code for the `div` is as follows:

```
<div id="blank" class="centered">
  <p>Open a web page to display properties.</p>
</div>
```

The next three lines set the CSS display property for the "blank" `div`, the "ui" `div`, and the "titleWarning" `div`.

note

While developing this add-in, I show you the basics of interacting with web pages using JavaScript. Although a complete discussion on using JavaScript functions to interact with the DOM is outside the scope of this book, a reference is available at [http://msdn.microsoft.com/en-us/library/ms533050\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms533050(v=VS.85).aspx).

The "blank" div contains a message that we want displayed when a page isn't open. The "ui" div contains the user interface used to display page information. When there isn't a page open in Expression Web, we want the "blank" div to be displayed, and we want the "ui" div to be hidden. By setting the CSS `display` property of the "blank" div to `block` and the `display` property of the "ui" div to `none`, the panel is displayed correctly to the user.

The "titleWarning" div displays a warning message to the user when there isn't a title defined for the page that is open in Expression Web. The user can control whether to display this warning using the Options dialog.

CSS Visibility Versus Display

You might be wondering why I chose to use the CSS `display` property rather than the CSS `visibility` property to control the visibility of each div. For example, I could have used the following code to hide the "ui" div.

```
ui.style.visibility = "hidden";
```

When you hide an element using the `visibility` property, the element still takes up space on the page even though it's not visible. For example, if the "ui" div were 100 pixels in height, setting the `visibility` property to `hidden` would result in a 100 pixel high blank spot. However, if you use the `display` property instead, the element is removed completely from the page.

Because of the way I've designed the panel's interface, this distinction isn't important. However, if I had placed the "blank" div under the "ui" div, setting the `visibility` property of the "ui" div to `hidden` would result in a large unwanted empty area above the "blank" div.

There are times when it's preferable to use the `visibility` property. If you want to hide a particular element that appears inline with other elements and you don't want the other elements to shift position, use the `visibility` property.

The `ActivatePanel` function is called when a page is opened that matches the type of file specified in the `filetype` attribute in the manifest. Listing 37.4 shows the code for the `ActivatePanel` function. (Line numbers are included for reference only.)

Listing 37.4 The `ActivatePanel` Function

```
1 function ActivatePanel() {
2     var loc = xweb.document.location;
3     if (loc.protocol == "unsaved:")
4     {
5         ClearPanel();
6         window.document.getElementById("blank").innerHTML =
7         "<p>Save this page to display properties.</p>";
8     }
9     else
```

```
9  {
10     var fileName = xweb.document.name;
11     var pathFromRoot = xweb.document.pathFromSiteRoot;
12     var fileUrl;
13     var blank = window.document.getElementById("blank");
14     var ui = window.document.getElementById("pageInfo");
15     var title = xweb.document.getElementsByTagName("title")[0];
16
17     if (pathFromRoot == undefined || pathFromRoot == "") {
18         fileUrl = fileName;
19     } else {
20         fileUrl = pathFromRoot + "/" + fileName;
21     }
22     blank.style.display = "none";
23     ui.style.display = "block";
24
25     SetLabel("pageUrl", fileUrl);
26     SetLabel("pageCreated", xweb.file.getCreationDate("site:" + fileUrl));
27     SetLabel("pageModified", xweb.file.getModificationDate("site:" +
    fileUrl));
28     SetLabel("fileSize", xweb.file.getSize("site:" + fileUrl));
29
30     if (title != undefined) {
31         SetLabel("pageTitle", title.innerHTML);
32     } else {
33         var ShowWarning = xweb.application.settings.read
    ("WarnOnMissingTitle");
34         if (ShowWarning == undefined || ShowWarning == "true")
35         {
36             titleWarning.style.display = "block";
37         } else {
38             titleWarning.style.display = "none";
39         }
40         window.document.getElementById("pageTitle").innerHTML =
    "[No Title]&nbsp;&nbsp;&nbsp;<a href=\" javascript:void()\" onclick=
    \"SetTitle();\">Click to set title.</a>";
41     }
42 }
43 }
```

This code is a bit more complex than the code for the `ClearPanel` function. Some of the code is similar to what you saw in the `ClearPanel` function, but the `ActivatePanel` function also contains code that uses the Expression Web JavaScript API. Let's look at this function in detail.

In Line 2, a variable called `loc` is declared and set to `xweb.document.location`. `xweb` is the *root namespace* of the Expression Web JavaScript API. Because object names aren't guaranteed to be unique, it's necessary to group objects within a logical container. That logical container is called a *namespace*, and the highest level namespace is called a *root namespace*.

Had I not specified the `xweb` namespace in Line 2, Expression Web would infer that the `document` object refers to `window.document` in the browser's DOM. Because `window.document.location` returns the path to the file displayed in the panel and not the file open in Expression Web, the code would not have worked as expected. By using the `xweb` namespace, I'm letting Expression Web know that "document" refers to the `document` object within the Expression Web API and not the browser's DOM.

The `xweb.document.location` property returns an `IHTMLLocation` object that contains information about the location of the file open in Expression Web. The `protocol` member of the `IHTMLLocation` object contains a string that defines the protocol of the open file. If the file has never been saved, the `protocol` will be "unsaved:" and properties for the file won't be available. In that case, we want to let the user know to save the file to display properties.

Line 3 checks the `protocol` of the `loc` object. If it's "unsaved:", `ClearPanel` is called to clear any existing information, and the "blank" `div` is changed to display a suitable message.

The rest of the function (beginning with Line 10) is the code that runs when the `protocol` of the open file is not "unsaved:". This is the code that uses the Expression Web JavaScript API to obtain properties about the current page and display them in the panel.

Line 10 defines a variable called `fileName` and sets it to `xweb.document.name`. The `xweb.document.name` property returns the name of the file open in Expression Web along with the file extension. The path to the file is not returned by `xweb.document.name`.

Line 11 defines a variable called `pathFromRoot` and sets it to `xweb.document.pathFromSiteRoot`. The `pathFromSiteRoot` property returns the relative path to the open file from the root of the site. For example, if I open `default.html` from inside a folder called `news`, `pathFromSiteRoot` would return `news/`.

Line 12 defines a variable called `fileUrl` with no value. We use this variable later to populate the value we want to display for the current page name.

Lines 13-15 get references to the "blank" `div`, the "pageInfo" `div`, and the `title` tag for the page. The technique used to do so should be familiar to you at this point.

Line 17 checks the value of `pathFromRoot` to determine whether the file that's open is in a subfolder. If it isn't, Line 18 sets the `fileUrl` variable to the value of `fileName`. Otherwise, `fileUrl` is set to a concatenation of `pathFromRoot` and `fileName` in Line 20.

Lines 22 and 23 set the CSS `display` property for the "blank" and "ui" `div`. The technique used should be familiar to you.

Lines 25-28 call the `SetLabel` JavaScript function. (We haven't written this function yet.) The `SetLabel` function takes two arguments: the name of a page element and a value. It then locates the page element passed to it and sets the `innerText` property of the element to the value that is passed to the function. By incorporating this functionality into its own function, it's more convenient to reuse the functionality in other parts of the add-in. It also makes the add-in's code easier to

**tip**

When writing add-in code, follow this simple rule. If you want to refer to the page that contains your add-in's interface, use the `window` object. If you want to refer to the page open in Expression Web, always use objects in the `xweb` namespace.

maintain. If a change is necessary in the code that sets the labels, the code only has to be changed in one place.

Line 25 passes the value of the `fileUrl` variable to the `SetLabel` function.

Lines 26, 27, and 28 use the `xweb.file.getCreationDate`, `xweb.file.getModificationDate`, and `xweb.file.getSize` functions to get information about the current file. Note that the argument to these functions is "site:" followed by `fileUrl`. When you're writing add-ins using the JavaScript API, file paths that refer to files in the site open in Expression Web are preceded by `site:`, and file paths that refer to files in the add-in's folder are preceded by `addin:`.

**note**

We'll write the code for the `SetLabel` function in the next section.

➡ *For more information on the `getCreationDate`, `getModificationDate`, and `getSize` functions, see Chapter 39, "Expression Web 4 JavaScript API Reference."*

Line 30 checks to see whether the `title` variable is defined. The `title` variable is set to the `title` element on the page on Line 15. However, if the `title` element is missing from the page, the `title` variable will be undefined. If the `title` variable is not undefined, `SetLabel` is called (Line 31) and the `innerText` property of the `title` element is passed to it so that we can display the title in the panel. If a `title` element isn't defined, the add-in reads a setting called `WarnOnMissingTitle` (Line 33) and uses that to determine whether to display the `titleWarning` element on the page in Lines 34-39.

➡ *For more information on using the `xweb.application.settings.read` function to read add-in settings, see Chapter 39, "Expression Web 4 JavaScript API Reference."*

Finally, in Line 40, the `innerHTML` property of the `pageTitle` element is set to a value letting the user know that no title exists on the page. A hyperlink is also added that calls the `SetTitle` function. We review the code for that function shortly.

As you saw in the `ActivatePanel` function, the `SetLabel` function is used to set the text displayed in page elements on the panel. The `SetLabel` function takes two arguments. The first is the ID of the page element that is being modified, and the second is the value that should be displayed in that element. Here is the code for the `SetLabel` function:

```
function SetLabel(label, value) {  
    window.document.getElementById(label).innerText = value;  
}
```

This code should be familiar to you by now.

The last function in `panel.htm` is the `SetTitle` function. Remember from Listing 37.4 that if a title isn't specified for the page, the Page Info add-in displays a hyperlink allowing the user to set a title. When this link is clicked, the `SetTitle` function is called. Listing 37.5 shows the code for the `SetTitle` function.

Listing 37.5 The setTitle Function

```
function setTitle() {
    var ret = xweb.application.showModalDialog('addin:settitle.htm',
    'Set Page Title', 'dialogHeight:125;dialogWidth:300;scroll:no');
    var title = xweb.document.getElementsByTagName("title")[0];
    if (ret)
    {
        // update title display
        SetLabel("pageTitle", title.innerHTML);
    }
}
```

This function uses the `xweb.application.showModalDialog` function to display a dialog that prompts the user for a title for the page. The technique used here is one that you'll likely use often in your add-ins. The variable `ret` is set to the value returned by the dialog. (The return value is actually set in the code contained within the `settitle.htm` page, and we look at that code next.) If the value returned by the `settitle.htm` dialog is true, `SetLabel` is called to update the title that is displayed in the panel.

➡ *For more information on the `xweb.application.showModalDialog` function, see Chapter 39, "Expression Web 4 JavaScript API Reference."*

JavaScript for options.htm

The Options dialog contains two JavaScript functions: `SetCheckboxByAppSettings` and `SetWarningOption`. `SetCheckboxByAppSettings` sets the initial state of the check box (shown previously in Figure 37.9) by reading an application setting. The following code listing shows the JavaScript code for this function:

```
function SetCheckboxByAppSettings()
{
    var WarnCheckbox = window.document.getElementsByName("WarnCheckbox")[0];
    var ShowWarning = xweb.application.settings.read("WarnOnMissingTitle");

    if (ShowWarning == false) WarnCheckbox.checked = false;
}
```

The first line of this function gets a reference to the check box control on the dialog. Because the HTML for the check box sets the name attribute to `WarnCheckbox`, the `getElementsByName` function is used to obtain a reference to it. Since the `getElementsByName` function returns an array of elements that match the name passed to it, the code gets the first element returned by using the `[0]` index.

The next line of code reads an application setting called `WarnOnMissingTitle`. (I'll cover the details on reading and writing

**note**

The checked attribute of the input element that defines the check box is set to true. Therefore, the check box is checked by default. However, we need to read the application setting so that the check box reflects the setting as configured by the user.

application settings in Chapter 39, “Expression Web 4 JavaScript API Reference.”) The next line sets the checked property of the check box based on the value of the WarnOnMissingTitle setting.

The WarnOnMissingTitle setting is set by the SetWarningOption function that is called when the OK button is clicked. Here is the JavaScript code for that function:

```
function SetWarningOption()
{
    var WarnCheckbox = window.document.getElementsByName("WarnCheckbox")[0];
    xweb.application.settings.write("WarnOnMissingTitle", WarnCheckbox.checked);
    xweb.application.endDialog(true);
}
```

Just as with the SetCheckboxByAppSettings function, the first line of SetWarningOption gets a reference to the WarnCheckbox check box control. The next line sets the WarnOnMissingTitle application setting based on whether the WarnCheckbox control is checked. It then calls endDialog to close the dialog.

**note**

Application settings are stored in the add-in's folder in a file called user.config. For details on this file and how Expression Web stores settings, see the documentation on xweb.application.settings in Chapter 39.

The Set Page Title Dialog

The Set Page Title dialog is called from the SetTitle function in panel.htm.

The Set Page Title dialog is implemented in settitle.htm. Create a new page and save it as settitle.htm. Once you've done that, switch to Code view and enter the code in Listing 37.6. Line numbers are for reference only.

Listing 37.6 Code for the Set Page Title Dialog

```
1 <html xmlns="http://www.w3.org/1999/xhtml">
2 <head>
3 <meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
4 <style type="text/css">
5 p {
6     font-family: Arial, Helvetica, sans-serif;
7     font-size: .75em;
8 }
9 input {
10    font-family: Arial, Helvetica, sans-serif;
11    font-size: .85em;
12 }
13 </style>
14 <script type="text/javascript">
15
16     function SetTitle() {
17         var newTitle = window.document.getElementsByName("title")[0];
18         var titleEle = xweb.document.getElementsByTagName("title")[0];
19         var headEle = xweb.document.getElementsByTagName("head")[0];
20
```


to the current value of `headEle.innerHTML`. Once Line 22 is executed, the `title` element will be the last child element inside the `head` element. Line 23 gets a reference to the last child element in the `head` element and assigns it to the `titleEle` variable.

Line 26 assigns the value entered into the `title` text box to the `title` element, and Line 27 calls `xweb.application.endDialog` and passes an argument of `true`. Remember that the `SetTitle` function in `panel.htm` checks for the return value from the call to `xweb.application.showModalDialog` to know whether to update the title displayed in the panel. The return value of `showModalDialog` is set by the argument passed to `endDialog`.

The `CancelDialog` function starts on Line 30. It's implemented using one line of JavaScript on Line 31 that calls `endDialog` and passes an argument of `false`. This value is checked in the `SetTitle` function in `panel.htm`. If the user clicks `Cancel` in the `Set Page Title` dialog, the code in `panel.htm` knows that it's not necessary to update the title displayed in the panel.

You now have a completed add-in that displays several properties about the page that is open in Expression Web. It can also warn a user when a title isn't set for a page, and in such cases, it allows the user to easily set a title. While this add-in may not be the most useful add-in, it has illustrated several important concepts regarding add-ins in Expression Web. By building the `Page Info` add-in, you've learned:

- How to create an add-in manifest that defines a menu item, a dialog, and a panel.
- How to use JavaScript to manipulate an add-in's user interface.
- How to use the new JavaScript API in Expression Web.
- How to display dialogs and store application settings.

The `Page Info` add-in used only a small fraction of the capability provided by the JavaScript API. As you progress to the next chapter, you'll gain in-depth knowledge of the entire JavaScript API so that you can build your own add-ins.

As you build more complex add-ins, it's likely that you'll eventually want to add functionality that the JavaScript API doesn't provide. Fortunately, it's possible to create a class using the .NET Framework that adds additional functionality to an add-in. In the next section, I show you how you can call a .NET Framework class from JavaScript.

Accessing Managed Classes from JavaScript

The functionality provided by the JavaScript API is sufficient for many add-in developers. However, some add-ins require additional capabilities not available in the JavaScript API. In these



tip

If your add-in doesn't work at this point, you can find tips on troubleshooting and debugging it in Chapter 38.



note

Classes built with the .NET Framework are referred to as *managed classes*.



note

Thanks to John Dixon, a developer on the Expression Web team, for his assistance in detailing how to call managed classes from JavaScript add-ins. You can read John's blog post on the topic at <http://blogs.msdn.com/b/jdixon/archive/2010/08/09/calling-managed-code-from-expression-web-4-html-js-extensibility-add-ins.aspx>.

cases, creating a managed class and calling that class from your JavaScript add-in can add significant functionality.

Let's create a simple managed class that we can call from the JavaScript API. You need a version of Visual Studio to complete this exercise. If you don't have a full version of Visual Studio, you can download Visual C#2010 Express from <http://www.microsoft.com/express/Downloads/#2010-Visual-CS>.



note

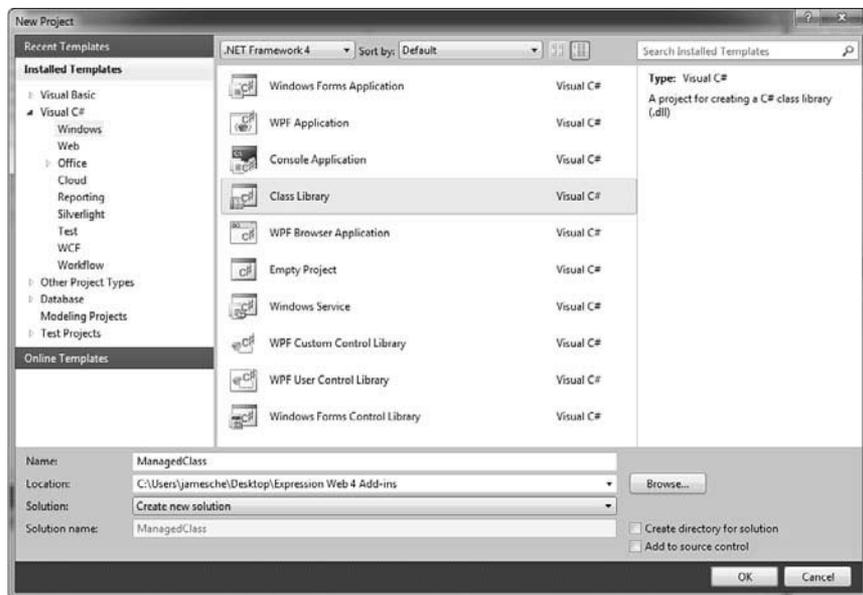
I've chosen C# as the language used for the managed class example because the vast majority of add-in developers are C# developers. The C# syntax is also similar to JavaScript syntax.

Creating a Managed Class

Follow these steps to create a managed class that you can call from JavaScript:

1. Open Visual Studio (or Visual C# 2010 Express) and select File, New Project.
2. In the Installed Templates pane, select Visual C#.
3. If you're using Visual Studio 2010, make sure that .NET Framework 4 is selected in the drop-down at the top of the dialog.
4. Select Class Library from the list of project templates.
5. Enter the location where you want to create your project and enter **ManagedClass** in the Name box as shown in Figure 37.10.
6. Click OK to create the class library project.

Figure 37.10
The New Project dialog in Visual Studio is used to create a class library that you can call from your JavaScript add-in.



Once your class library project is created, the `Class1.cs` code file is displayed. Change the name of your class from `Class1` to `AddinUtility` by changing this code:

```
public class Class1
```

to this:

```
public class AddinUtility
```

Next, select **File, Save Class1.cs As** and save the file as `AddinUtility.cs`.

To use your new class from a JavaScript add-in, add the following code to the top of `AddinUtility.cs`.

```
using System.Runtime.InteropServices;
```

You then need to make your class visible to Expression Web. To do that, change your class declaration from this:

```
public class AddinUtility
```

to this:

```
[ComVisible(true)]  
public class AddinUtility
```

Now you need to add a reference to the extensibility DLL so that you can use functionality provided by that DLL. To do that, follow these steps:

1. Select **View, Solution Explorer** (or **View, Other Windows, Solution Explorer** in Express Edition) to ensure that the Solution Explorer is visible.
2. Right-click on the **References** folder in Solution Explorer and select **Add Reference**.
3. Click the **.NET** tab in the Add Reference dialog.
4. Select **extensibility** from the list of components as shown in Figure 37.11 and click **OK**.

You now need to make another change to the class declaration so that your managed class can make a connection to Expression Web. Change the class declaration from this:

```
public class AddinUtility
```

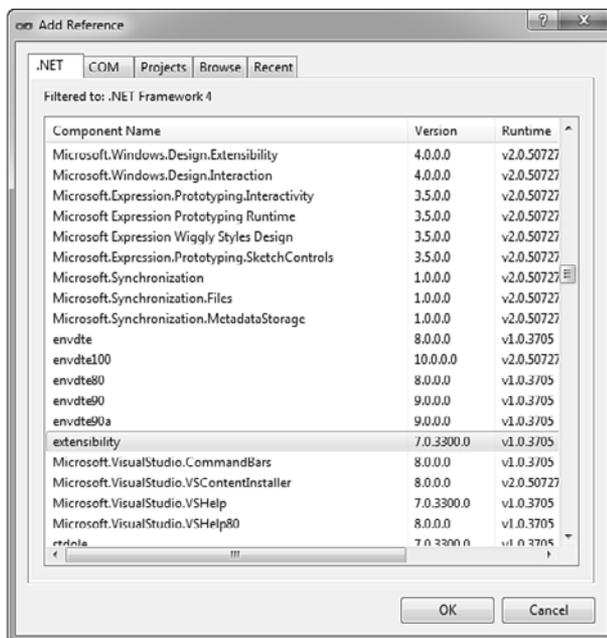
to this:

```
public class AddinUtility : Extensibility.IDTExtensibility2
```

 **note**

In this section, I show you the steps necessary to create a managed class that you can call from a JavaScript add-in. I won't go into detail on some of the concepts presented because creating managed classes is outside the context of this book.

Figure 37.11
Adding a reference to the extensibility DLL.



Don't press Enter after adding this code. Instead, hover over the word "Extensibility" and click the button that appears beneath it. Select Implement Interface 'Extensibility.IDTExtensibility2' from the menu that appears as shown in Figure 37.12. When you do, Visual Studio adds some required code to your class.

Figure 37.12
Implementing the IDTExtensibility interface requires some code to be added to your class.



You may notice that each of the functions that are part of the `IDTExtensibility2` interface contain the following line of code:

```
throw new NotImplementedException();
```

This line causes an error to occur when any of these functions are called. Because Expression Web calls the `OnConnection` function automatically when your add-in loads, you need to remove this line of code from that function. After you do, the `OnConnection` function should contain no code at all.

Now we need to add a function to our managed class that we can call from JavaScript. For this example, we add a function that returns the amount of memory that the Expression Web process is using, something that's not possible using the JavaScript API.

Add the following function to the code for the `AddinUtility` class. Make sure that you add this code immediately after the opening curly bracket for the class.

```
public Int64 GetMemoryUsage()
{
    long totalMemory;
    Int64 privateMemory;

    System.Diagnostics.Process webProcess =
    System.Diagnostics.Process.GetCurrentProcess();
    totalMemory = webProcess.PrivateMemorySize64;
    totalMemory = (totalMemory / 1024) / 1024;

    privateMemory = Convert.ToInt64(totalMemory);

    return privateMemory;
}
```

This code uses the `System.Diagnostics.Process.GetCurrentProcess` function to return a reference to the `ExpressionWeb.exe` process. It then uses the `PrivateMemorySize64` property of the process to find out how much memory is being used by the process. `PrivateMemorySize64` returns the memory usage in bytes, so that value is converted to megabytes and assigned to the `privateMemory` variable. The function then returns the `privateMemory` variable.

Once you've added this code, compile the `AddinUtility` class by selecting **Build, Build ManagedClass**. Visual Studio compiles your DLL and saves it to the `bin\Debug` folder within your project's folder. Copy `ManagedClass.dll` from that folder to your add-in's folder.

Editing the Add-in Manifest to Load the Managed Class

Managed classes are loaded by using the `load` element in the add-in's manifest. To load the `ManagedClass.dll` assembly, add the following code into the `addin.xml` file in your add-in's



note

`IDTExtensibility2` is a special programming construct called an *interface* and using an interface is called *implementing the interface*. An interface is a kind of programming contract, and when you implement an interface, you agree to abide by that contract. The code that Visual Studio adds to your class is part of that contract.



tip

If you want some code to run when your add-in loads, you can put that code into the `OnConnection` function.

folder. For reference, I've included the `addin` element in the following code. However, you'll want to add only the `load` element into your manifest.

```
<addin developer="yes">
  <load type="ManagedClass.AddinUtility,ManagedClass" name="util" />
```

Save your manifest and close and reopen Expression Web if necessary.



note

The details of the `load` element were covered in Chapter 36.

Calling the Managed Class

To call the `GetMemoryUsage` from your JavaScript add-in, open `panel.htm` and locate the following line of code in the `ClearPanel` function:

```
blank.innerHTML = "<p>Open a web page to display properties.</p>";
```

Add the following two lines directly under the existing line:

```
blank.innerHTML += "<p>Memory usage: <span id='memSize'>&nbsp;</span>MB</p>";
SetLabel("memSize", util.GetMemoryUsage());
```

This code defines a new `span` for displaying the memory used. It then calls the `GetMemoryUsage` function in the managed class using the `util` namespace as defined by the `load` element added to the manifest. The `SetLabel` function is used to set the `innerText` of the `span` to the value returned by the `GetMemoryUsage` function.

Save `panel.htm` and right-click within the Page Info panel and select Refresh. If you open any page and then close all pages, you see a message in the panel that tells you how much memory is being used by Expression Web.

In this example, you've seen how you can use a managed class to obtain functionality that's not possible from the JavaScript API. The possibilities available when using managed classes are almost infinite. For more information on what's possible using the .NET Framework, see the .NET Framework Developer Center at <http://msdn.microsoft.com/en-us/netframework/default.aspx>.



note

Don't be alarmed if you see that Expression Web is using more than 100MB of memory. Expression Web is a managed process, and managed processes typically use more memory than other processes.

Summary

In the past two chapters, you learned some of what's possible with the JavaScript API. You also learned how to extend that functionality by creating a managed class and calling functions on that class from your JavaScript add-in.

You now have the basic skills necessary to create your own add-ins for Expression Web. You know how to create a manifest, add menu and toolbar items, create panels, and create dialogs.

In the next chapter, you learn how you can troubleshoot and debug your add-ins as you develop them. You learn about ways you can troubleshoot your add-ins from within Expression Web, and you also learn how you can use Visual Studio's powerful debugging features to debug your add-ins.

This page intentionally left blank

PACKAGING, TESTING, AND DEBUGGING ADD-INS

Creating an Add-in Installation Package

If you're going to write add-ins, you're going to have to deal with software bugs. Even the simplest add-in is likely to contain bugs, so testing and debugging your add-ins is a must. However, before you can test and debug an add-in, you need to install the add-in.

If you used the Add-in Builder to create your add-in manifest, installation of the add-in is done automatically for you. However, you'll still want to create an installation package so that others can install your add-in.

➔ *For more information on using the Add-in Builder, see "Creating the Manifest with the Add-in Builder," p. 662.*

Creating an add-in installation package is easy to do. Simply zip the folder that contains your add-in files and then change the file extension to `.xadd`. (You'll need to ensure that Windows is configured to show file extensions.) You can create a zip file by right-clicking on your add-in's folder and selecting Send To, Compressed (zipped) Folder from



note

A great place to share your add-in is the Microsoft Expression Gallery at <http://gallery.expression.microsoft.com/>.



tip

You cannot double-click on an `.xadd` file to install an add-in. To install an add-in, you need to use the Manage Add-ins dialog in Expression Web.

the menu. Once the folder has been zipped, change the file extension of the Zip file to `.xadd`. Your add-in is now ready for installation.

Testing and Debugging Add-ins

You've spent many hours developing a really cool add-in, and you're eager to upload it to the Expression Gallery so that other Expression Web users can use it. However, before you do, you should spend some time testing and debugging your add-in so that users don't experience problems. Even if your add-in is completely free, if users experience problems, they will complain and make sure to let other users know about their trouble. Testing and debugging are critical to the success of any add-in.

Testing Add-ins

The easiest way to test an add-in is to use it yourself during your normal usage of Expression Web. However, that's not the best way. It's likely that other people who may use your add-in (assuming you decide to share it with others) use Expression Web differently than you do. Subtle changes in the way that an add-in is used can expose serious bugs that you'll want to deal with before you share your add-in with others.

The specific scenarios you use when testing your add-in will differ depending on the complexity of your add-in. For example, testing the Page Info add-in was fairly simple because the act of opening and closing pages causes all the JavaScript functions used by the add-in to execute. However, when you're dealing with a more complex add-in, you may have JavaScript functions that execute only when certain conditions are met. In these situations, it's usually best to build a test page that's designed to call each of your functions. If an error is encountered, you can then debug the error more easily.

I follow some general rules when testing all of my add-ins:

- Test the add-in with disk-based sites, FTP sites, and HTTP sites.
- Test the add-in with pages at numerous folder levels within my site.
- Test the add-in with no sites open in Expression Web.
- If the add-in requires user input, conduct tests with no input and with bogus input.
- Find people willing to test the add-in for me and provide feedback.

I find (and fix) plenty of problems when going through the first four of these rules, but I'm always amazed at how many problems I miss that are uncovered by other people using my add-ins.

Regardless of how much effort you put into predicting how users will use your add-in, you will still encounter a surprising number of unexpected scenarios once your add-in gets into the hands of users.

When you provide your add-in to testers, make sure that you give them some guidelines for reporting bugs. For example, let them know that you need detailed, step-by-step instructions for reproducing



tip

Keep in mind that how any particular function works is often impacted by external factors. Create as many test scenarios as you can when testing your functions. This section gives you some pointers on how to do that.

any problems they encounter. Provide an example of a bug submission so that they'll know what you expect. If you don't, you often won't get enough detail to reproduce a particular problem.

As an example, the following steps are not sufficient to reproduce a problem:

1. Opened my site.
2. Started editing my page.
3. Got the error message.

However, the following steps would likely allow you to reproduce a problem and get it corrected:

1. Opened a site at `http://www.mysite.com`.
2. Opened `aboutme.htm` from the root folder.
3. Switched to Design View.
4. Added the text "Click Here for Info".
5. Selected the text and clicked the Insert Hyperlink button in Expression Web.
6. Added a hyperlink to `moreinfo.htm` in the root folder of the site.
7. Clicked outside that hyperlink and got the error.

If you provide your testers with examples of what you expect, you'll get better information from them.

It's also important that you understand a tester's expectation compared with what was experienced when encountering a problem. Not all problems that testers encounter are actually bugs. When a tester encounters a problem, your add-in may be operating exactly the way you designed it, but your design may not be what the tester expects. For that reason, I like to have testers include steps that were performed, what was expected, and what actually happened.

Once you are able to reproduce a problem, you can then debug it and fix it. Let's review some of the tools available for debugging add-ins.

**tip**

No amount of testing can identify all software bugs. Even simple add-ins are likely to be released with some undiscovered bugs.

Debugging Add-ins Using Expression Web

Debugging is the process of identifying and fixing problems in software. You can debug your add-ins from within Expression Web using a couple of different tools: the Extensibility Tester and debug consoles.

The Extensibility Tester

The Extensibility Tester is an add-in developed by Microsoft for testing the JavaScript API. Using the Extensibility Tester, you can test JavaScript code easily outside your add-in project. For example, suppose your add-in needs to get the path of the file currently open in Expression Web, but you aren't sure whether you want to use `xweb.document.filename` or `xweb.document.location`.

Using the Extensibility Tester, you can evaluate each of these to determine which property is appropriate for your particular situation.

You can download the Extensibility Tester from <http://gallery.expression.microsoft.com/en-us/ExtensibilityTester>. After you download and install it, select Panels, Extensibility Tester to open it. The Extensibility Tester is divided into four windows:

- **Input**—Expressions to evaluate are added here. To evaluate an expression, click the Evaluate button.
- **Output**—If no error occurs when evaluating the expression, the result is displayed here.
- **Error**—If an error occurs when evaluating the expression, the error is displayed here.
- **Events**—Displays events in real-time.

➔ *For more information on events, see Chapter 39, “Expression Web 4 JavaScript API Reference.”*

In Figure 38.1 the expression `xweb.document.filename` is being evaluated. Because a file is not currently open in Expression Web, “Object required” is displayed in the Error window. An “Object required” error means that the JavaScript expression refers to an object that doesn’t exist. In this case, the `xweb.document` object doesn’t exist because a file isn’t open in Expression Web.

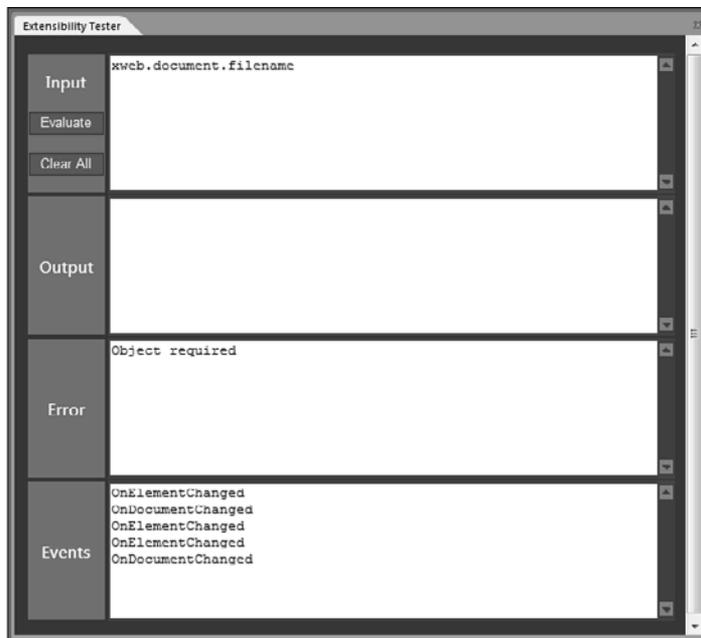
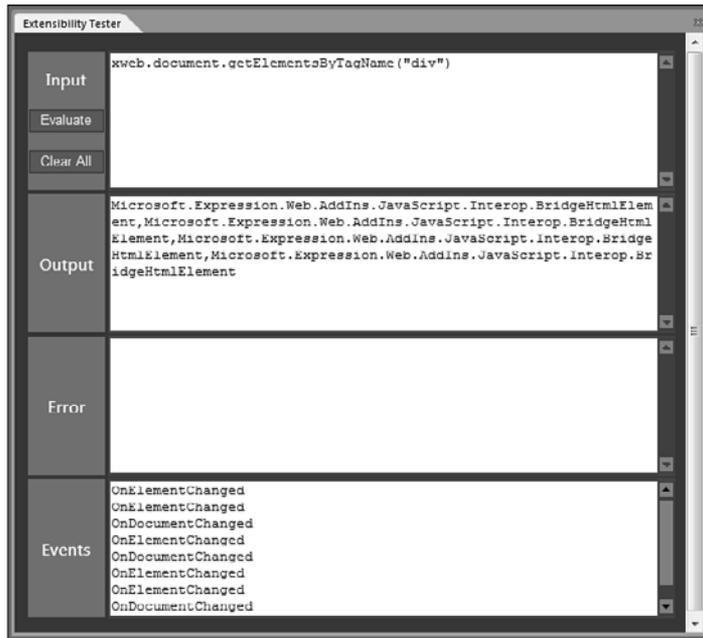


Figure 38.1
The Extensibility Tester add-in makes it possible to test JavaScript expressions easily.

In some cases, the output you get from the Extensibility Tester may not be sufficient to resolve a problem. For example, in Figure 38.2, the following code is being evaluated:

```
xweb.document.getElementsByTagName("div")
```

Figure 38.2
The output from the Extensibility Tester isn't always as helpful as you might want it to be.



The Output window displays the value `Microsoft.Expression.Web.AddIns.JavaScript.Interop.BridgeHtmlElement` four times. (The `BridgeHtmlElement` object is an internal object that Expression Web uses to identify an HTML element.) If I want to find out additional information about a particular `div`, I need to change my input to refer to a particular element. The `getElementsByTagName` method returns a zero-based array of elements, so I can use the following code to get a reference to the first `div` on the page:

```
xweb.document.getElementsByTagName("div")[0]
```

Figure 38.3 shows the output obtained when evaluating the `innerHTML` property of the first `div` on the page.



tip

The Extensibility Tester is a great way to experiment with the JavaScript API and learn more about how to develop add-ins.



Figure 38.3

By using a more specific input string, I am able to get useful information about a particular div.

Debug Consoles

If your add-in is in developer mode, you can use the debug console to aid in debugging your add-in. To access the debug console for your add-in, select it from the Panels menu in Expression Web. The menu item for your add-in's debug console will be named using the name of your add-in's folder. Therefore, if your add-in is installed into the `C:\Users\jim\AppData\Roaming\Microsoft\Expression\Web 4\Addins\MyAddin` folder, the debug console can be opened by selecting Panels, MyAddin (console).

The debug console displays some error information automatically. For example, if your add-in's manifest contains errors, the debug console may display helpful information about the error. However, you can also use the `xweb.developer` object to write directly to the debug console.



tip

As discussed in Chapter 36, "Expression Web 4 Add-in Basics," to specify that an add-in run in developer mode, set the `developer` attribute in the manifest's `addin` element to `yes`.



tip

If an error is encountered in your manifest (and when some other errors occur), Expression Web creates a file called `errors.txt` in the root of your site that contains the error message.

The `xweb.developer` object has two methods that can be used to write content to the debug console:

- `xweb.developer.write(string)`—Displays the value passed as a string in the debug console. The string value can either be an explicit string or an expression that evaluates to a string.
- `xweb.developer.writeLine(string)`—Equivalent to the `write` method except that `writeLine` includes a carriage return at the end of the string so that each string appears on a new line.

Consider the following code snippet:

```
var loc = xweb.document.location;  
var linkDiv = xweb.document.getElementById("link");  
linkDiv.innerHTML = "<a href=\"" + loc.href + "\">Click Here</a>";
```

Suppose that when this code runs, the hyperlink created within the `div` points to a URL that's not what I expect. By adding a call to `xweb.developer.write`, you can easily determine the value of `loc.href` before you use it in your code. Here's the same code with an additional line that writes the value of `loc.href` to the debug console.

```
var loc = xweb.document.location;  
xweb.developer.writeLine("loc.href = " + loc.href);  
var linkDiv = xweb.document.getElementById("link");  
linkDiv.innerHTML = "<a href=\"" + loc.href + "\">Click Here</a>";
```

Figure 38.4 shows the contents of the debug console when this code is run.

Figure 38.4
The debug console displaying the value of `loc.href`.



Debugging Add-ins Using Visual Studio

The debug console and Extensibility Tester are nice tools for troubleshooting add-ins, but if you really want a full-featured debugger, Visual Studio is your best choice.

To debug your add-in in Visual Studio, follow these steps:

1. Open Internet Explorer and select Tools, Internet Options.
2. Click the Advanced tab.
3. Uncheck Disable Script Debugging (Internet Explorer).



tip

To clear the contents of the debug console, close and reopen Expression Web.

4. Uncheck **Disable Script Debugging (Other)**. Your Internet Options dialog should look like the one shown in Figure 38.5.
5. Click **OK**.
6. Ensure that all panels and dialogs for your add-in are closed and close Expression Web.
7. Launch Visual Studio and open the file that contains the script you want to debug.
8. Right-click on the line where you want a breakpoint and select **Breakpoint, Insert Breakpoint**.
9. Launch Expression Web, but do not open your add-in's panel or dialog box.
10. Switch back to Visual Studio and select **Debug, Attach to Process**.
11. Locate **ExpressionWeb.exe** in the Available Processes window and make sure that **Script** appears in the **Type** column as shown in Figure 38.6. If it does, proceed to step 15.
12. If **Script** does not appear in the **Type** column, click the **Select** button.
13. Select the **Debug These Code Types** radio button and check the **Script** check box.
14. Click **OK**.
15. Select **ExpressionWeb.exe** from the Available Processes list and click the **Attach** button.
16. Switch back to Expression Web and access your add-in.

At this point, Visual Studio should break into the process when your breakpoint is hit and you'll have access to all of the debugging tools available in Visual Studio.

Using this method, you can set a breakpoint and debug from that point. However, you can also use Visual Studio to debug unexpected errors that may occur while testing your add-in.

Once you have script debugging enabled in Internet Explorer, when a script error is encountered, you are asked whether you want to debug the file containing the script as shown in Figure 38.7. If you click **Yes**, the Visual Studio Just-In-Time Debugger dialog displays and you can choose a debugger as shown in Figure 38.8. Choose **New Instance of Visual Studio 2010** and click **Yes** to debug your add-in. (Your version of Visual Studio may differ.)

 **note**

Debugging add-ins in Expression Web requires a full version of Visual Studio. You cannot debug Expression Web add-ins with any of the Express editions.

 **note**

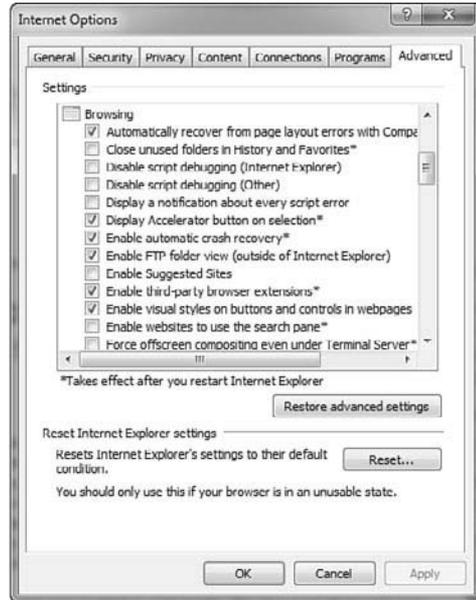
Expression Web add-ins actually run inside an Internet Explorer control. That's why you need to ensure that script debugging is enabled in Internet Explorer before you can debug your add-ins in Visual Studio.

 **note**

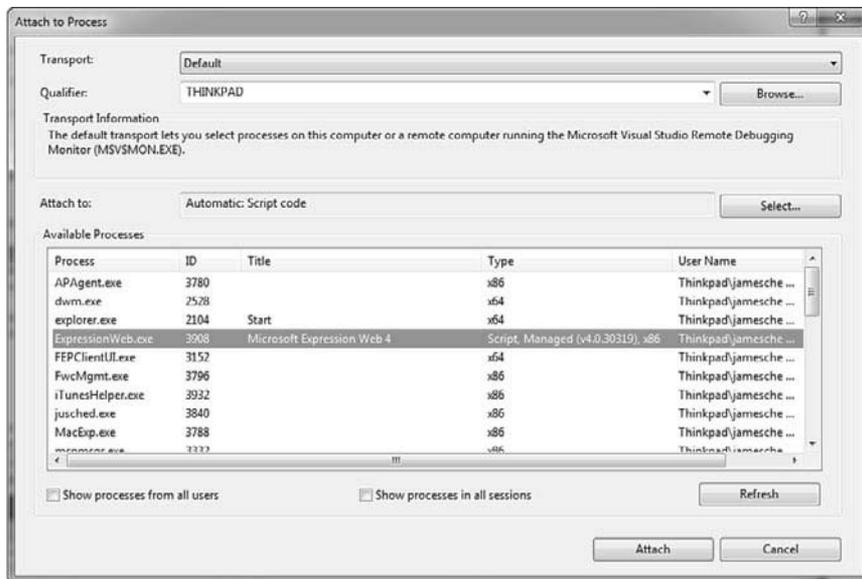
Thanks to John Dixon on the Expression Web product team for documenting details on how to do this on his blog. You can read John's blog post at <http://blogs.msdn.com/b/jdixon/archive/2010/09/02/debugging-javascript-add-ins-within-expression-web-4.aspx>.

Figure 38.5

You need to uncheck both Disable Script Debugging check boxes in Internet Explorer before you can debug your add-in.

**Figure 38.6**

Make sure that Visual Studio is configured to debug scripts.



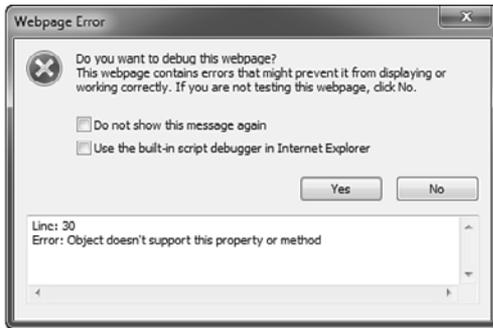


Figure 38.7
Internet Explorer displays a script error dialog when script debugging is enabled.

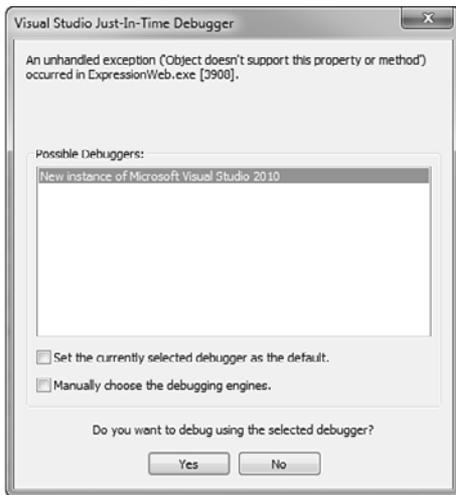


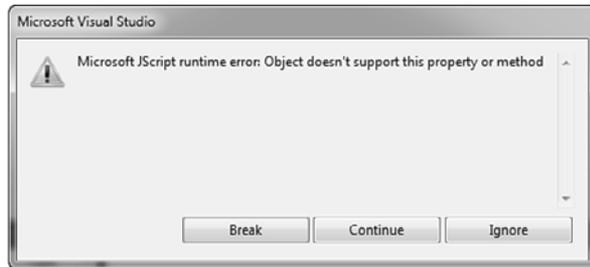
Figure 38.8
The Visual Studio JIT debugger can debug scripts that generate errors.

When Visual Studio launches, it automatically attaches the debugger to Expression Web, and the script error is displayed as shown in Figure 38.9. Click Break and Visual Studio breaks at the point where the error occurred.

note

You can read details on all of Visual Studio's debugging tools at <http://msdn.microsoft.com/en-us/library/sc65sadd.aspx>.

Figure 38.9
The script error is displayed within Visual Studio. Clicking Break allows you to debug it.



Perpetual Script Errors

Once you enable script debugging in Internet Explorer, an error message is displayed each time a script error is encountered. If you're developing a panel for your add-in and you have an error in your script that runs when the panel is activated or deactivated, it's possible that a script error will be displayed each time you open a page or close a page. If an error occurs when you are closing a page and you click No to the prompt asking whether you want to debug the error, Expression Web will not close the page.

To correct such a situation, you need to first correct the script error that is causing your problem. Once you do that and you save the file containing the script, right-click on your panel and select Refresh. Your panel then runs the corrected script and you should be able to open and close pages without an error message.

You should also know that when script debugging is enabled, you may see errors occur on many sites as you browse the Internet. For that reason, it's a good idea to disable script debugging again after you've finished debugging your Expression Web add-in.

Summary

In this chapter, you familiarized yourself with the tools available for troubleshooting and debugging your add-ins. You now have all the skills necessary for developing and debugging add-ins in Expression Web. However, we've only touched on the capabilities of the JavaScript API.

In the next chapter, you'll find a complete reference on the Expression Web 4 JavaScript API. The API reference is designed to be a valuable tool for looking up information on the API while you're developing add-ins. However, it's also helpful to read the reference so that you can familiarize yourself with all the capabilities of the JavaScript API.

This page intentionally left blank

EXPRESSION WEB 4 JAVASCRIPT API REFERENCE

Conventions Used in this Reference

This reference covers the objects, properties, and methods (functions) in the Expression Web 4 JavaScript API. To avoid any confusion as you are using this reference, the following conventions are followed.

Each object in the JavaScript API has its own heading that identifies the object name. Even though all objects in the Expression Web JavaScript API share the `xweb` root namespace, the `xweb` namespace is included in code samples to avoid any confusion with objects in the browser's DOM.

Properties and methods of each object are documented in subheadings of each object. When code samples are given, the following syntax is used.

- Samples of method calls use the syntax `object.method(param)`. Optional parameters appear in square brackets as in `object.method(param1, [param2])`.
- When a parameter has more than one possible explicit value, each possible value is separated by a vertical bar as in `value1|value2`.

Properties contain syntax, return types, and often an Information section with more information about the property. Methods contain syntax, parameters (if applicable), return value, and often contain an Information section with more information about the method.

Properties of an object are listed first, followed by methods. Properties and methods appear in alphabetical order.

As with the other chapters of this book, the reference contains numerous notes and tips. Cautions are also included when warranted.

This reference does not include material on the DOM's `window` or `document` objects. For information on the browser's DOM, see Microsoft's Scripting Internet Explorer guide located at [http://msdn.microsoft.com/en-us/library/dd464669\(v=VS.85\).aspx](http://msdn.microsoft.com/en-us/library/dd464669(v=VS.85).aspx).

Code examples in this chapter will often refer to explicit file names. This was done to clarify the examples, but in most cases, your add-in will get file names from user input or from programmatically accessing the current site.

XWEB.APPLICATION Object

The `xweb.application` object refers to the Expression Web application itself. In addition to being used for manipulating files and common dialogs within the Expression Web application, it is also used when opening and closing modal add-in dialogs.

xweb.application.version Property

The `version` property returns a string that identifies the version of Expression Web.

Syntax

```
xweb.application.version;
```

Returns

The full version number is returned as a string.

Information

The `version` property is read-only.

Example

The following code example checks the version of Expression Web. If the version of Expression Web isn't what's required for the add-in, an alert is displayed notifying the user to download and install the latest service pack.

```
// This add-in requires at least version 4.0.1234.
// This function checks for that version.

// get application version as a split array
var currVersion = xweb.application.version.split('.');

// check build.
if (currVersion[0].valueOf() == 4 && currVersion[2].valueOf() < 1234)
```

```
{
    alert("This add-in requires at least service pack 1 of Expression Web 4." +
        "\n\nPlease download and install the service pack before continuing.");
} else
{
    // perform a function requiring service pack 1
}
```

xweb.application.chooseFile Method

The `chooseFile` method displays a Browse dialog (commonly referred to as a file picker).

Syntax

```
xweb.application.chooseFile(label, [filter]);
```

Parameters

The `label` parameter is a string that specifies the text to display on the filter drop-down in the Browse dialog. The `filter` parameter allows you to specify which file types are displayed in the browse dialog.

Returns

The full path and filename of the file that is selected in the dialog as a string. If no file is selected or if Cancel is clicked in the Browse dialog, returns `null`.

Information

The syntax for the `filter` parameter is `*.ext`, and multiple filters can be applied by separating each with a comma.

Example

The following example displays a Browse dialog configured for selecting a web file.

```
var webFile = xweb.application.chooseFile("Web Files",
    "*.htm,*.html,*.asp,*.aspx,*.php");
if(webFile) alert("Full path: " + webFile);
```

Figure 39.1 shows the appearance of the Browse dialog that is launched by this code. Notice that the label on the filter drop-down reflects the value of the `label` parameter.



tip

The first time that `chooseFile` is called during an Expression Web session, it opens at an initial location of `%windir%\System32`. Thereafter, it opens in the same directory that was displayed when it was last closed until Expression Web is closed and reopened.

There is no way to specify the initial directory displayed in the Browse dialog when using the `chooseFile` method.

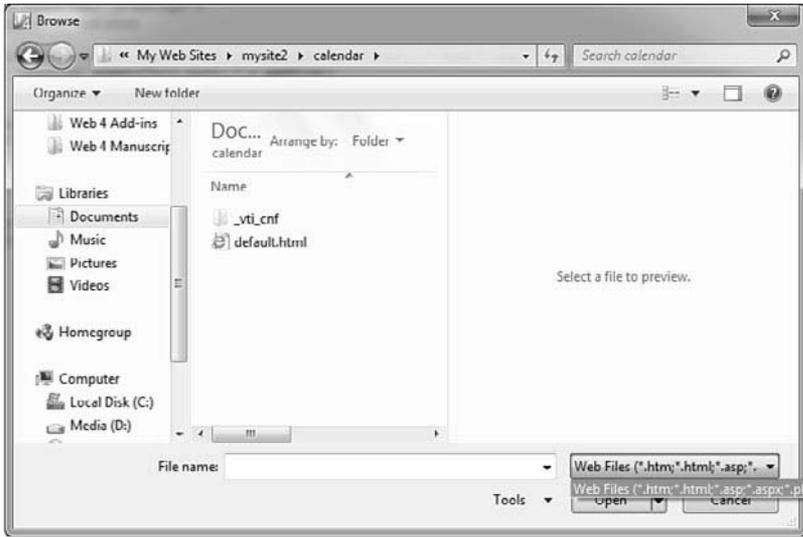


Figure 39.1
The Browse dialog shown here was launched using the `chooseFile` method.

xweb.application.endDialog Method

The `endDialog` method closes a modal dialog that was opened using the `showModalDialog` method.

Syntax

```
xweb.application.endDialog(boolValue);
```

Parameters

The `boolValue` parameter is passed back to the `showModalDialog` method that launched the dialog and is set as the return value of that method. It can either be a Boolean (`true` or `false`) or a numeric value. Numeric values will be implicitly converted to a Boolean. A value of `0` represents `false`, and any other number (including negative numbers) represents `true`.

Returns

Nothing.

Information

If no value is provided for the `boolValue` parameter, an invalid procedure call error occurs.

Example

```
<input type="button" id="btnOK" value="OK" onclick="ProcessDialog(false)">
<input type="button" id="btnCancel" value="Cancel"
  onclick="ProcessDialog(true)">
<script type="text/javascript">
function ProcessDialog(canceled)
{
  // check to see if Cancel was clicked
  if(!canceled) {
    // Cancel not clicked. Handle dialog logic.
  }
  // Cancel was clicked.
  xweb.application.endDialog(canceled);
}
</script>
```

In the example, the call to `showModalDialog` returns `false` if the user clicks OK and `true` if the user clicks Cancel. See the code example for `showModalDialog` for more information.

xweb.application.handleEvent Method

The `handleEvent` method tells the Expression Web application to handle the event that is passed to it. This method is commonly used with add-ins that provide hotkey functionality in order to hand off the handling of the `keydown` event to Expression Web.

Syntax

```
xweb.application.handleEvent(event);
```

Parameters

The `event` parameter is a JavaScript event (`IHTMLElementObj`) object.

Returns

Returns `true` if the event is handled by Expression Web. Otherwise, `false`.

Information

The `handleEvent` method only handles the `keydown` event.

Example

```
<script type="text/javascript">
function HandleEvent(e)
{
    if (e.ctrlKey)
    {
        // handle shortcut key
        switch (e.keyCode)
        {
            case 75: //k
                // do something on a Ctrl-K
                break;
            // other case statements for other hotkeys
            default:
                // Hotkey not used by the add-in.
                // Pass it off to Expression Web to handle.
                xweb.application.handleEvent(e);
        }
    }
}
</script>
<input type="text" name="textinput" onkeydown="HandleEvent(event)">
```

In the example code, the `HandleEvent` function executes when a key is pressed while inside the text box. The `HandleEvent` checks to see whether the Ctrl key is pressed and then handles the event based on which key was pressed. If the key that was pressed is not a hotkey handled by the add-in, the event is passed to Expression Web so that it can handle it. Without the call to `xweb.application.handleEvent`, the add-in would have swallowed the event without Expression Web having an opportunity to handle it.

`xweb.application.newDocument` Method

The `newDocument` method performs the same action as clicking the New Document button in the Expression Web interface. However, you can specify a particular Doctype declaration by passing in an optional parameter.

The `newDocument` method sets the focus to the newly created document.

Syntax

```
xweb.application.newDocument([doctype]);
```

Parameters

The optional `doctype` parameter is a string that specifies the doctype for the new document.

Returns

Returns an `xweb.document` object representing the newly created page.

Information

The value of the `doctype` parameter can be one of the following:

- HTML 4.01 Strict
- HTML 4.01 Transitional
- XHTML 1.0 Frameset
- XHTML 1.0 Strict
- XHTML 1.0 Transitional
- XHTML 1.1

If you don't specify a `doctype` parameter, Expression Web creates the page using the Document Type Declaration setting on the Authoring tab of the Page Editor Options dialog.

If the value that you specify for `doctype` is not one of the values previously stated, document creation fails and `newDocument` returns `null`.

➔ *For more information on the `xweb.document` object, see “`xweb.document` Object,” p. 715, in this chapter.*

Example

The following code creates a new file with an XHTML 1.0 Transitional doctype. Before it does so, it determines the root-relative path of the active document and uses that path to reactivate the document after the new file is created.

```
// get the root-relative path of the active document
var docPath = xweb.document.pathFromSiteRoot + '/' + xweb.document.filename;

// create a new file with XHTML 1.0 Transitional doctype
xweb.application.newDocument("XHTML 1.0 Transitional");

// reactive the previously active document.
xweb.application.openDocument("site:" + docPath);
```

This code relies on the fact that when `openDocument` is called on a document that is already open, the document is activated.



note

Microsoft's documentation on the `newDocument` method states that all the Doctype values that appear on the Authoring tab of the Page Editor Options dialog are valid for the `doctype` parameter. However, this is not true. You must use one of the values previously stated.

xweb.application.openDocument Method

The `openDocument` method opens a web page.

Syntax

```
xweb.application.openDocument(URI);
```

Parameters

The `URI` parameter is the URI of the document to be opened.

Returns

Returns an `xweb.document` object.

Information

The `URI` parameter must be preceded with `site:`. If the `URI` parameter points to a file that is not a web page (for example, a `.css`, `.xml`, `.txt` file and so forth), a valid `xweb.document` will not be returned. However, the open operation will succeed. Therefore, an add-in should not assume a valid `xweb.document` object has been returned by `openDocument`.

The following example illustrates the recommended way to use the `openDocument` method.

Example

The following example opens a file and checks to ensure that the file is a web page.

```
var doc = xweb.application.openDocument("site:" + fileURI);
if (doc.name)
{
    // perform some action on the xweb.document object assigned to doc
}
```

In the preceding example, if `fileURI` contains a URI to a file that is not a web page, `doc.name` returns `null`. In such a case, even though the file will be opened in Expression Web, the `doc` variable will not contain a valid `xweb.document` object.

xweb.application.refreshFileListing Method

The `refreshFileListing` method refreshes the Folder List in Expression Web.



tip

If the `fileURI` refers to a file that is already open in Expression Web, the tab for the already opened file will be activated.

Syntax

```
xweb.application.refreshFileListing();
```

Parameters

None.

Returns

If successful in refreshing the Folder List, `refreshFileListing` returns `true`. Otherwise, it returns `false`.

Information

The `refreshFileListing` method only works correctly if the Folder List has the focus when it is called. Therefore, the recommended way of refreshing the Folder List using an add-in is to enable legacy mode in the add-in's manifest and call `xweb.legacyapp.ActiveWeb.Refresh(true)`.

Example

The following example refreshes the Folder List. Note that the first example only works when the Folder List has focus, so you'll likely use the second example in most cases.

```
// works only when Folder List has focus
xweb.application.refreshFileListing();

// works regardless of focus, but requires 'legacy="yes"' in manifest
xweb.legacyapp.ActiveWeb.Refresh(true);
```

xweb.application.setActiveDocument Method

The `setActiveDocument` method activates the `xweb.document` that is passed to it.

Syntax

```
xweb.application.setActiveDocument(doc);
```

Parameters

The `doc` parameter is the `xweb.document` object to activate.

Returns

Returns `true` if the document is activated. Otherwise, returns `false`.

Example

The following example opens two documents. It then activates the first document that was opened.

```
var doc1 = xweb.application.openDocument("site:products.htm");
var doc2 = xweb.application.openDocument("site:default.htm");
xweb.application.setActiveDocument(doc1);
```

xweb.application.setPanelVisibility Method

The `setPanelVisibility` method allows you to show or hide panels within the Expression Web interface.

Syntax

```
xweb.application.setPanelVisibility(panel, "hidden"|"visible");
```

Parameters

The `panel` parameter is a case-sensitive string that specifies the ID of the panel that is being shown or hidden. The second parameter sets the visibility of the panel and is either "hidden" or "visible".

Returns

Returns `true` if the panel was found. Otherwise, returns `false`.

Information

An add-in's panel ID is defined in the add-in's manifest. Table 39.1 lists the IDs of Expression Web's built-in panels.

Example

The following example sets the visibility of the Manage Styles panel.

```
xweb.application.setPanelVisibility("TaskPane_ManageStyles", "visible");
```

The `setPanelVisibility` method does not affect the focus of a panel. If the panel indicated by the `panel` parameter is docked with other panels but is not the active panel, making the panel visible with `setPanelVisibility` will not activate the panel's tab.

When docked panels are shown using the `setPanelVisibility` method, they appear in the same position they were in before being hidden.



tip

The return value of `setPanelVisibility` does not reflect the state of the panel. If a panel is already visible and `setPanelVisibility` makes the panel visible, it still returns `true` even though visibility wasn't changed.



tip

If more than one panel shares the same ID, `setPanelVisibility` sets the visibility of the first panel encountered with the specified ID.

Table 39.1 Panel IDs for Built-In Panels

Panel Name	Panel ID
Folder List	TaskPane_FolderList
Tag Properties	TaskPane_TagProperties
CSS Properties	TaskPane_CssProperties
Apply Styles	TaskPane_ApplyStyles
Manage Styles	TaskPane_ManageStyles
Behaviors	TaskPane_Behaviors
Layers	TaskPane_Layers
Toolbox	TaskPane_Toolbars
Data Source Library	TaskPane_DataSourceLibrary
Data Source Details	TaskPane_DataSourceDetails
Conditional Formatting	TaskPane_ConditionalFormatting
Find 1	TaskPane_Find1
Find 2	TaskPane_Find2
Accessibility	TaskPane_Accessibility
Compatibility	TaskPane_Compatibility
SEO	TaskPane_Seo
Hyperlinks	TaskPane_Hyperlinks
CSS Reports	TaskPane_CssReports
Publishing Status	TaskPane_PublishingStatus

xweb.application.showModalDialog Method

The `showModalDialog` method opens a modal dialog.

Syntax

```
xweb.application.showModalDialog(dialogURI, dialogTitle, [dialogOptions]);
```

Parameters

The `dialogURI` parameter refers to the web page to be displayed in the dialog and is relative to the root of the add-in's folder. The `dialogTitle` parameter is a string displayed in the title bar of the dialog. `dialogOptions` is an optional parameter that defines the features of the dialog window.



tip

You can also set the visibility of a panel added by an add-in. You'll need to check the manifest of the add-in to determine the panel's ID.

Returns

A Boolean. The return value for `showModalDialog` is set by the `endDialog` method.

Information

It is not necessary to precede `dialogURI` with `addin:`.

The `dialogTitle` parameter is not optional, but you may specify either an empty string or `null`, in which case the title bar of the dialog will be blank.

The `dialogOptions` parameter includes an optional list of semicolon-delimited options for the dialog in the format `option:value`. The following options are available:

- `dialogHeight`—The height of the dialog as a numeric value in pixels. Do not include a unit of measure.
- `dialogWidth`—The width of the dialog as a numeric value in pixels. Do not include a unit of measure.
- `resizable`—Determines whether the dialog can be resized by dragging the edges of the window. To make the dialog resizable, valid values are `yes`, `1`, `true`, or `on`. To prevent the window from being resized, valid values are `no`, `0`, `false`, or `off`. Dialogs are not resizable by default.
- `scroll`—Specifies whether the dialog should have scrollbars. Valid values are `yes`, `1`, `true`, or `on` to enable scrollbars and `no`, `0`, `false`, or `off` to disable scrollbars. Scrollbars are enabled by default.



tip

Vertical scrollbars are always visible. Horizontal scrollbars are only visible when the width of the dialog's content exceeds the width of the dialog.

Example

The following example opens a modal dialog displaying the content of `dialog.htm`. The dialog has a title of `Options Dialog`, is 200 pixels high, is 400 pixels wide, and has no scroll bars.

```
var ret = xweb.application.showModalDialog("dialog.htm", "Options Dialog",  
"dialogHeight:200;dialogWidth:400;scroll:no");
```

XWEB.APPLICATION.SETTINGS Object

The `settings` object is used to store and retrieve user-specific settings for an add-in. Settings are stored in a `user.config` file located inside the add-in's folder.

xweb.application.settings.read Method

The `read` method reads a setting.

Syntax

```
xweb.application.settings.read(setting);
```

Parameters

The `setting` parameter is a string that identifies the setting to be retrieved.

Returns

Returns the value of the specified setting.

Information

If the setting you attempt to read does not exist, the `read` method returns undefined.

Example

The following example reads a setting called `ConfirmOnClose`. If the setting is true, a confirmation message is displayed before a document is closed. Otherwise, the document is closed without a prompt.

```
var ConfirmOnClose = xweb.application.settings.read("ConfirmOnClose");
if (ConfirmOnClose)
{
    if (confirm("Are you sure you want to close?"))
    {
        xweb.document.close();
    }
} else
{
    xweb.document.close();
}
```

xweb.application.settings.write Method

The `write` method stores a setting and value.

Syntax

```
xweb.application.settings.write(setting, value);
```

Parameters

The `setting` parameter is a string that identifies the setting. The `value` parameter is the value to be stored.

Returns

Nothing.

Information

The `value` parameter must be a primitive data type such as a number, a Boolean value, or a string.

The `write` method has no return value. Therefore, if you want to ensure that the value was written, use the `read` method to check the value.



tip

If the setting specified by the `setting` parameter already exists, the existing value is overridden by the new value.

Example

The following example writes a setting called `ConfirmOnClose` and sets it based on a checkbox called `chkConfirm`.

```
var chkConfirm = document.getElementById("chkConfirm");  
xweb.application.settings.write("ConfirmOnClose", chkConfirm.checked);
```

XWEB.DEVELOPER Object

The `xweb.developer` object contains methods that write information to the debug console for add-ins.

`xweb.developer.write` Method

The `write` method writes a value to the debug console for an add-in.

Syntax

```
xweb.developer.write(value);
```

Parameters

The `value` parameter is the value to be written to the debug console.



tip

For methods of the `developer` object to succeed, you must set the `developer` attribute of the `addin` element in the add-in manifest to `yes`.

Returns

Undefined.

Information

Subsequent calls to the `write` method append new values to the existing value. If a line break is desired between values, use the `writeLine` method instead.

Example

The following example uses the `xweb.developer.write` method to log output to the debug console.

```
// open a document
var doc = xweb.application.openDocument("site:" + fileUrl);

// write the location of the document to the debug console
xweb.developer.write(doc.location);
```

xweb.developer.writeLine Method

The `writeLine` method is equivalent to the `write` method except that a line break is added after each value.

Syntax

```
xweb.developer.writeLine(value);
```

Parameters

The `value` parameter is written to the debug console, followed by a line break.

Returns

Undefined.

XWEB.DOCUMENT Object

The `xweb.document` object represents a web page in Expression Web. The properties and methods of the `xweb.document` object provide the capability to get information about the associated page and to manipulate its content.

note

The debug console can be accessed from the Panels menu. A debug console is available for each add-in running in developer mode. The name of the debug console panel reflects the name of the add-in's folder and not the name of the add-in.

caution

The `xweb.document` object refers to a web page with a valid DOM. Other files types (such as `.css`, `.xml`, and so forth) do not have a valid `xweb.document` object associated with them.

Don't confuse `xweb.document` with `window.document`. The `xweb.document` object represents a web page open in Expression Web. The `window.document` object represents the document in an add-in's panel or dialog.

➔ *For more information on the `xweb.document.synchronizeViews` method, see "xweb.document.synchronizeViews Method," p. 733, in this chapter.*

The `xweb.document` object refers to the active document in Expression Web. However, an `xweb.document` object can also be referenced by a variable. It is possible to access properties and methods of the `xweb.document` object without that `xweb.document` being the active page in Expression Web. The code in Listing 39.1 uses two variables, each of which contains an `xweb.document` object, and code is modified in the document that is not active in Expression Web.

Listing 39.1 Example of `xweb.document` Usage

```
// The openDocument method returns an xweb.document object and makes
// that document the active document in Expression Web.
var newsDoc = xweb.application.openDocument("site:news/news.htm");
var faqDoc = xweb.application.openDocument("site:faq.htm");

// faq.htm is now the active document in Expression Web.

// add a script reference to news.htm which is NOT the active document.
newsDoc.appendScriptReference("addin:scripts/jquery.js", "javascript");
```

When the code in Listing 39.1 runs, the `news.htm` file is modified by adding a new script reference. However, the `faq.htm` file remains the active document in Expression Web. In other words, it's not necessary to activate a document to manipulate it using the properties and methods of the `xweb.document` object. The `xweb.document` object references the active document in memory. If anything causes the active document in Expression Web to change, the `xweb.document` object reference will change as well. The following example illustrates this concept.

```
xweb.application.openDocument("site:default.htm");
var doc = xweb.document;
// doc.filename is now "default.htm"

// The following call changes the document referenced by "doc"
xweb.application.openDocument("site:panel.htm");
// doc.filename is now "panel.htm"
```



tip

When used without explicitly specifying the `xweb` namespace, the `document` object refers to `window.document`.



caution

Properties and methods of the `xweb.document` object may fail or return incorrect values if the document contains unsaved changes. To prevent this, always call `synchronizeViews` on the `document` object prior to accessing properties and methods of the `xweb.document` object.

xweb.document.anchors Property

The `anchors` property locates all anchors on the document.

Syntax

```
xweb.document.anchors;
```

Returns

An array of `xweb.document.htmlElement` objects is returned. If no anchors are defined in the page, the `anchors` property returns `null`.

Information

The `anchors` property is read-only.

Example

The following example gets a reference to the first anchor in the active document and displays the `name` attribute of that anchor in an alert.

```
// get a reference to the first anchor in the document
var anchor = xweb.document.anchors[0];

// show an alert with the anchor name
alert(anchor.getAttribute("name"));
```

**tip**

An anchor is an element without an `href` attribute.

xweb.document.applets Property

The `applets` property locates all applet elements in the document.

Syntax

```
xweb.document.applets;
```

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing an applet element in the document. If no applet elements are present in the document, the `applets` property returns `null`.

Information

The `applets` property is read-only.

xweb.document.embeds Property

The `embeds` property locates all `embed` elements in the document.

Syntax

```
xweb.document.embeds;
```

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing an `embed` element in the document. If no `embed` elements are present in the document, the `embeds` property returns `null`.

Information

The `embeds` property is read-only.

xweb.document.filename Property

The `filename` property returns the filename and extension of the document.

Syntax

```
xweb.document.filename;
```

Returns

The filename and extension as a string.

Information

The `filename` property does not return any part of the path of the document. Only the filename and extension are returned by the `filename` property.

If a site is not open in Expression Web, the `filename` property returns `undefined`.

To obtain a relative path to a document, append the value of `filename` to the value returned by the `pathFromSiteRoot` property.

**tip**

The `filename` property always returns `undefined` when the site open in Expression Web is not a disk-based site.



For more information on the `xweb.document.pathFromSiteRoot` property, see “`xweb.document.pathFromSiteRoot` Property,” p. 722, in this chapter.

The `filename` property is read-only.

Example

The following example determines the relative path of the active document from the root of the site.

```
var relPath = xweb.document.pathFromSiteRoot + '/' + xweb.document.filename;
```

xweb.document.forms Property

The `forms` property locates all form elements in the document.

Syntax

```
xweb.document.forms;
```

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing a form element in the document. If no form elements are present in the document, the `forms` property returns `null`.

Information

The `forms` property is read-only.

xweb.document.frames Property

The `frames` property locates all frame elements in the document.

Syntax

```
xweb.document.frames;
```

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing a frame element in the document. If no frame elements are present in the document, the `frames` property returns `null`.

Information

The `frames` property is read-only.



The `frames` property does not return any `iframe` elements.

xweb.document.images Property

The `images` property locates all `img` elements in the document.

Syntax

```
xweb.document.images;
```

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing an `img` element in the document. If no `img` elements are present in the document, the `images` property returns `null`.

Information

The `images` property is read-only.

xweb.document.isXHTML Property

The `isXHTML` property determines whether the document is an XHTML document.

Syntax

```
xweb.document.isXHTML;
```

Returns

Returns a Boolean value indicating whether the document contains an XHTML doctype. If no doctype exists in the document, `isXHTML` returns `false`.

Information

The `isXHTML` property is read-only.

xweb.document.links Property

The `links` property locates all hyperlinks in the document.

Syntax

```
xweb.document.links;
```



tip

You may see this documented as `isXhtml`. However, the Expression Web JavaScript API doesn't pay attention to case, so `isXhtml` is equivalent to `isXHTML`.

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing an a element with an `href` element in the document. If no a elements with `href` attributes are present in the document, the `links` property returns `null`.

Information

The `links` property is read-only.

`xweb.document.location` Property

The `location` property returns the location of the document.

Syntax

```
xweb.document.location;
```

Returns

The absolute file path of the document as a string. If the active site is a disk-based site, `location` returns `file:///` followed by the path and filename. If the site is a server-based site, the `location` property returns the path preceded by `http://` or `ftp://` depending on the type of site.

Information

The `location` property actually returns an `IHTMLLocation` object and not a string. Therefore, it's possible to obtain much more useful information than simply the path of a page using the `location` element. For example, `location.host` returns the host (for example, `www.jimcobook.com`) when the document is part of a server-based site.

For more information on the `IHTMLLocation` object, see [http://msdn.microsoft.com/en-us/library/aa209080\(office.11\).aspx](http://msdn.microsoft.com/en-us/library/aa209080(office.11).aspx).

The `location` property is read-only.

Example

The following example checks the `location` property to determine if the active document is unsaved. If it is, the user is prompted to save the document.

```
var unsaved = false;

// check to see if the active document is unsaved
```



tip

Don't confuse anchors with links. If an a element doesn't contain an `href` attribute, it's an anchor and is returned by the `anchors` property. If it does contain an `href` attribute, it's a link and is returned by the `links` property.



tip

The `location` property is the easiest way to determine whether a document is an unsaved file. If the document has never been saved, `location.protocol` returns `unsaved:`. The code example illustrates this.

```
if (xweb.document.location.protocol == "unsaved:") unsaved = true;

if (unsaved)
{
    alert("Please save the document.");
}
```

xweb.document.name Property

The `name` property returns the name of the document.

Syntax

```
xweb.document.name;
```

Returns

The name of the document, along with the file extension, as a string.

Information

This property is identical to the `filename` property. The `name` property is read-only.

xweb.document.pathFromSiteRoot Property

The `pathFromSiteRoot` property is used to determine where the document exists relative to the root of the site.

Syntax

```
xweb.document.pathFromSiteRoot;
```

Returns

Returns the relative path from the root of the site to the document as a string. If the document exists in the root of the site, `pathFromSiteRoot` returns an empty string.

Information

Use the `pathFromSiteRoot` property to construct relative paths to web pages.

The `pathFromSiteRoot` property is read-only.

Example

The following example computes the relative path from the site root for the active document.

```
var pathFromRoot = xweb.document.pathFromSiteRoot;
var name = xweb.document.fileName;

// compute the relative path for the active document
var relPath = pathFromRoot + '/' + name;
```

xweb.document.scripts Property

The `scripts` property locates all `script` elements in the document.

Syntax

```
xweb.document.scripts;
```

Returns

Returns an array of `xweb.document.htmlElement` objects, each representing a `script` element in the document. If no `script` elements are present in the document, the `scripts` property returns `null`.

Information

The `scripts` property is read-only.

xweb.document.selection Property

The `selection` property returns whatever is selected in the document.

Syntax

```
xweb.document.selection;
```

Returns

A selection object.

➔ *For more information on the selection object, see “xweb.document.selection Object,” p. 751, in this chapter.*

Information

The selection property is read-only.

Example

The following example wraps the selection in the active document within a div element.

```
var pw = xweb.legacyapp.ActivePageWindow;
var doc = xweb.document;

// Check for a dirty page.
// If page is dirty and we're in Code view, flip to Design and back to Code.
// This works around a Permission Denied error that occurs otherwise.
if (pw.IsDirty && pw.ViewMode == 2)
{
    pw.ViewMode = 1;
    pw.ViewMode = 2;
}

doc.selection.wrap("<div>", "</div>");
```

The example code uses the legacy `ActivePageWindow` object to check for unsaved changes via the `IsDirty` property. It also uses the `ViewMode` property to work around a problem that prevents programmatic manipulation of code on a dirty page.

`xweb.document.appendScriptReference` Method

The `appendScriptReference` method adds a script reference before the closing head element.

Syntax

```
xweb.document.appendScriptReference(path, type);
```

Parameters

The `path` parameter is a string and uses the syntax `addin:script.js`. It can also include a relative path such as `addin:scripts/script.js`.

The `type` parameter is a string that identifies the MIME type of script that is being referenced. Typical values would be `javascript`, `ecmascript`, or `vbscript`.

Returns

Returns `true` if the script reference was added successfully and `false` otherwise.

Information

The script reference must be a script file that is located within the add-in's folder. If a script reference is added to a script within the site, the path to the script is added as an absolute path that can cause problems after a site is published.

Example

The following code adds a script reference immediately before the closing head element of the active document.

```
var doc = xweb.document;  
doc.appendScriptReference("addin:scripts/jquery.js", "javascript");
```

After this code runs, the following script reference is added.

```
<script type="text/javascript" src="jquery.js"></script>
```

The `appendScriptReference` method adds the script reference with an absolute path. When the affected page is saved, Expression Web will display the Save Embedded Files dialog so that the user can specify where to save the embedded script file.



tip

If you attempt to add a reference to a script that is already referenced, the `appendScriptReference` method will not add a reference and will return `false`.

xweb.document.appendStyleReference Method

The `appendStyleReference` method adds a style sheet reference before the closing head element.

Syntax

```
xweb.document.appendStyleReference(path);
```

Parameters

The `path` parameter refers to a style sheet in the add-in's folder and is in the format `addin:style.css`. A relative path can also be included as in `addin:styles/style.css`.

Returns

Returns `true` if the style sheet reference was successfully added and `false` otherwise.

Information

The style sheet must exist within the add-in's folder. Style sheet references are added in the following format.

```
<style rel="stylesheet" type="text/css" href="style.css" />
```



tip

If you attempt to add a style sheet reference that already exists, nothing will be added and `appendStyleReference` will return `false`.

Example

The following example links a style sheet called `styles.css` to the active document. The `styles.css` file is located in the `styles` folder within the add-in's folder.

```
var doc = xweb.document;  
doc.appendChildReference("addin:styles/styles.css");
```

The `appendChildReference` method adds the style sheet link with an absolute path. When the affected page is saved, Expression Web will display the Save Embedded Files dialog so that the user can specify where to save the embedded style sheet.

`xweb.document.close` Method

The `close` method closes the document.

Syntax

```
xweb.document.close();
```

Parameters

None.

Returns

Returns `true` if the document is successfully closed and `false` otherwise.

`xweb.document.getElementById` Method

The `getElementById` method returns the first element in the DOM of the document that contains an `id` attribute that matches the argument passed to it.

Syntax

```
xweb.document.getElementById(id);
```

Parameters

The `id` parameter is the `id` attribute of the element you want to return. If more than one element is found with the specified `id`, the first element encountered is returned.

Returns

The `getElementById` method returns an `htmlElement` object.

➔ For more information on the `htmlElement` object, see “`htmlElement` Object,” p. 742, in this chapter.

Example

The following example gets a reference to an element with an `id` attribute value of `banner` and sets the HTML code inside of the element.

```
var ele = xweb.document.getElementById("banner");
ele.innerHTML = "<img src=\"images/banner.png\" alt=\"Advertisement\">";
```

`xweb.document.getElementsByAttributeName` Method

The `getElementsByAttributeName` method returns elements that contain the specified attribute.

Syntax

```
xweb.document.getElementsByAttributeName(attrib);
```

Parameters

The `attrib` parameter is a string that specifies the attribute name.

Returns

Returns an array of `htmlElement` objects, each representing an element with the specified attribute. If no elements are found with the specified attribute, returns an empty array.

Example

The following code returns an array with all elements that have a `id` attribute. It then iterates through the array and populates a text area on the page with each ID.

```
var selID = xweb.document.getElementsByTagName("textarea")[0];
var arrID = xweb.document.getElementsByAttributeName("id");

for (var i = 0; i < arrID.length; ++i)
{
    selID.innerHTML += arrID[i].id + "<br />";
}

xweb.document.synchronizeViews;
```

xweb.document.getElementsByTagName Method

The `getElementsByTagName` method returns elements with a tag name that matches the specified tag name.

Syntax

```
xweb.document.getElementsByTagName(tag);
```

Parameters

The `tag` parameter is a string that specifies an HTML element name.

Returns

An array of `htmlElement` objects with the specified tag name. If no elements are found with the specified tag name, an empty array is returned.

Example

For an example of using the `getElementsByTagName` method, see the code example for the `getElementsByTagName` method.

xweb.document.getScriptElementByCode Method

The `getScriptElementByCode` method searches for code within `script` blocks in the document and returns the first `script` element that contains the specified code.

Syntax

```
xweb.document.getScriptElementByCode(code);
```

Parameters

The `code` parameter is a case-insensitive string that specifies the script code that the `getScriptElementByCode` method is to find.

Returns

The first `script` element with matching code is returned as an `htmlElement` object. If no `script` elements are found containing the specified code, returns `null`.

Example

The following example uses the `getScriptElementByCode` method to determine if a specific script has already been inserted onto the page.

```
var scriptBlock;
scriptBlock = xweb.document.getScriptElementByCode("function doSomething()");
if (!scriptBlock)
{
    // add script to page
}
```

xweb.document.getScriptElementByFile Method

The `getScriptElementByFile` method searches script references in the document for script elements with an `src` attribute that matches a particular pattern.

Syntax

```
xweb.document.getScriptElementByFile(pattern);
```

Parameters

The `pattern` parameter can be any part of the path specified by the `src` attribute of the script element.

Returns

The first script element that matches the pattern is returned as an `htmlElement`. If no script element is found, `getScriptElementByFile` returns `null`.

Information

If more than one script element matches the specified pattern, only the first element is returned.

Consider the following script reference:

```
<script src="scripts/jscript.js" type="text/javascript" />
```

The following values for the `pattern` parameter are examples of patterns that match this script reference:

```
scripts
jscript
jscript.js
pts/jsc
```

xweb.document.getStyleElementByCode Method

The `getStyleElementByCode` method searches `style` elements in the document for a specified code pattern.

Syntax

```
xweb.document.getStyleElementByCode(pattern);
```

Parameters

The `pattern` parameter is a case-insensitive string that contains the text for which to search inside `style` elements.

Returns

The first `style` element found that contains the specified string is returned as an `htmlElement` object. If no `style` elements are found, returns `null`.



tip

The `getStyleElementByCode` method searches not only CSS code, but also CSS comments.

Example

The following example returns the first `style` element that defines a `first-child` pseudo class.

```
var styleEle;
styleEle = xweb.document.getStyleElementByCode(":first-child");
if (styleEle)
{
    // do something with style element
}
```

xweb.document.getStyleElementByFile Method

The `getStyleElementByFile` method searches for a filename pattern in `style` sheet links.

Syntax

```
xweb.document.getStyleElementByFile(pattern);
```

Parameters

The `pattern` parameter is a case-insensitive string that contains the pattern to find.

Returns

Returns the first `link` element that matches the pattern as an `htmlElement` object. If no match is found, returns `null`.

Information

The `getStyleElementByFile` method searches the `href` attribute of all `link` elements with a `rel` attribute set to `stylesheet`.

xweb.document.insertBeforeHtml Method

The `insertBeforeHtml` method inserts text at the top of the document immediately before the opening `html` element.

Syntax

```
xweb.document.insertBeforeHtml(text);
```

Parameters

The `text` parameter contains the text to be inserted.

Returns

Returns `true` if the text was inserted and `false` otherwise.

Information

The `insertBeforeHtml` method can be used to insert comments, a doctype, ASP.NET or PHP information, or any other information that needs to be inserted before the opening `html` element.

The `insertBeforeHtml` method does not check to see whether the text you are inserting has already been inserted. If you call it multiple times with the same argument, it adds the same text to the document multiple times.

Example

The following example inserts a copyright block at the top of the active document.

```
var copyright;
copyright = "<!-- Copyright 2011, Jimco Software //-->\n";
// ensure views are in sync
xweb.document.synchronizeViews();
// add the copyright
xweb.document.insertBeforeHtml(copyright);
```

xweb.document.save Method

The `save` method saves the document. This method is equivalent to selecting File, Save in Expression Web.

Syntax

```
xweb.document.save([path]);
```

Parameters

The optional `path` parameter can be used to save a file under a new name.

Returns

Returns `true` if the document is successfully saved. Otherwise, it returns `false`.

Information

The path must be preceded by `site:` and it must be relative to the root of the site.

Example

The following example saves the existing file as `newfile.htm`.

```
xweb.document.save("site:newfile.htm");
```

xweb.document.saveAs Method

The `saveAs` method opens the Save As dialog so that the document can be saved under a different name.

Syntax

```
xweb.document.saveAs();
```

Parameters

None.

Returns

Returns `true` if the document is successfully saved and `false` otherwise.

Information

If you want to programmatically save the document under a new name without user interaction, use the `xweb.legacyapp.ActivePageWindow.SaveAs` method.

Example

The following example shows how to save a document under a new name with and without user interaction.

```
// save the document under a new name with user interaction
xweb.document.saveAs();

// save the document under a new name without user interaction
xweb.legacyapp.ActivePageWindow.SaveAs("folder/file.htm");
```

xweb.document.synchronizeViews Method

The `synchronizeViews` method synchronizes the Code and Design views in Expression Web.

Syntax

```
xweb.document.synchronizeViews();
```

Parameters

None.

Returns

Returns `true` if views are successfully synchronized and `false` otherwise.

Information

When changes are made to a page programmatically, Expression Web may not recognize that change automatically. Also, changes made while in Code view may not be reflected in Design view.

To ensure that the views in Expression Web are always up to date, call the `synchronizeViews` method before and after making any programmatic change to a document.

XWEB.FILE Object

The `file` object refers to a file within either a site or within an add-in's folder. The `file` object is used to interact with files that do not have a DOM. To interact with files that have a DOM, use the `xweb.document` object.

xweb.file.copy Method

The `copy` method copies a file from one location to another.

Syntax

```
xweb.file.copy(source, dest, overwrite);
```

Parameters

The `source` parameter specifies the source file's URI. The `source` parameter's value should be preceded by `site:` if the file is in the active site and `addin:` if the file is in the add-in's folder.

The `dest` parameter specifies the destination URI. The value of the `dest` parameter should be preceded by `site:`.

The `overwrite` parameter is a Boolean value that specifies whether to overwrite the destination file if it already exists. If the `overwrite` parameter is `true`, the file will be overwritten if it already exists.

Returns

The `copy` method returns `true` if the file is successfully copied. Otherwise, it returns `false`.

Information

You cannot use the `copy` method to copy a file to the add-in's folder.

The Folder List in Expression Web is not automatically refreshed after the `copy` method is executed. Therefore, it's advisable to use the following method call after a call to the `copy` method:

```
xweb.legacyapp.ActiveWeb.Refresh(true);
```



tip

The preceding code requires that the `legacy` attribute be set to `true` for the `addin` element in your manifest.

Example

The following example copies a style sheet from the add-in folder to the active site.

```
if (!xweb.file.exists("site:styles/styles.css"))
{
    xweb.file.copy("addin:styles.css", "site:styles/styles.css");
    xweb.legacyapp.ActiveWeb.Refresh(true);
}
```

xweb.file.createFile Method

The `createFile` method creates a file at the location specified.

Syntax

```
xweb.file.createFile(path);
```

Parameters

The `path` parameter is a string that specifies the location where the file is to be created relative to the root of the site. It must be preceded by `site:` and includes the filename and extension. Any extension is valid.

Returns

Returns `true` unless an error occurs. If an error occurs, returns `false`.

Information

If the file specified by the `path` parameter already exists, `createFile` does nothing, but it still returns `true`.

Files created by `createFile` are empty. Therefore, it's best to programmatically add the necessary base code (such as `html`, `head`, and `body` elements) using the `xweb.document.insertBeforeHtml` method after creating the file.



caution

If the path specified matches an existing file, the existing file will be replaced with the new file. Therefore, you should always check for an existing file using the `xweb.file.exists` method as shown in the code example.

Example

The following example creates a new HTML file and adds the necessary base code.

```
if (!xweb.file.exists("site:about.htm"))
{
    if (xweb.file.createFile("site:about.htm"))
    {
        xweb.legacyapp.ActiveWeb.Refresh(true);
        xweb.application.openDocument("site:about.htm");
        xweb.document.insertBeforeHtml("<html>\n<body>\n</body>\n</html>");
    }
}
```

xweb.file.createFolder Method

The `createFolder` method creates a folder in the current site at the path specified.

Syntax

```
xweb.file.createFolder(path);
```

Parameters

The `path` parameter is a string specifying the root relative path for the new folder. It must be preceded with `site:.`

Returns

Returns `true` if the folder is successfully created and `false` otherwise. If the folder specified by the `path` parameter already exists, `createFolder` returns `false`.

Information

The Folder List is not automatically refreshed after the new folder is created, so it's advised that you programmatically refresh the Folder List after calling this method.

xweb.file.deleteFile Method

The `deleteFile` method deletes the file that is specified.

Syntax

```
xweb.file.deleteFile(file);
```

Parameters

The `file` parameter is a string specifying the location of the file to delete relative to the root of the current site. It must be preceded by `site:.` Any file type is valid.

Returns

Returns `true` if the file is successfully deleted and `false` otherwise.

Information

You cannot delete folders using the `deleteFile` method.

It is possible to delete a file that is currently open in Expression Web.

The Folder List is not automatically refreshed after the new folder is created, so it's advised that you programmatically refresh the Folder List after calling this method.



caution

The `deleteFile` method deletes files without any warning. Be careful when using this method.

xweb.file.exists Method

The `exists` method checks to see whether the specified file exists in the current site.

Syntax

```
xweb.file.exists(path);
```

Parameters

The `path` parameter is a string that specifies the path and name of the file relative to the root of the site. It must be preceded by `site:`.

Returns

Returns `true` if the file exists and `false` if it doesn't.

xweb.file.getAttributes Method

The `getAttributes` method returns attributes of the specified file or folder.

Syntax

```
xweb.file.getAttributes(path);
```

Parameters

The `path` parameter is a string specifying the path to the file or folder relative to the root of the site. It must be preceded by `site:`.

Returns

Returns a string indicating the attributes of the file or folder. If the file or folder doesn't exist, returns `null`.

Information

The following is a list of possible attributes that are returned:

- R—Read-only
- D—Folder
- H—Hidden
- S—System

If the file specified by the `path` parameter is hidden and read-only, the following is returned by `getAttributes`.

RH

xweb.file.getCreationDate Method

The `getCreationDate` method returns the date that the specified file was created.

Syntax

```
xweb.file.getCreationDate(path);
```

Parameters

The `path` parameter is a string that specifies the path to the file relative to the root of the site. It must be preceded by `site:.`

Returns

Returns a JavaScript `Date` object. If the specified file is not found, returns `undefined`.

Information

Because `getCreationDate` returns a `Date` object, you can call any of the `Date` object's methods and properties for additional functionality. The code example illustrates this concept.

Example

The following example uses the `getCreationDate` method to obtain time and date information about a file:

```
var d = xweb.file.getCreationDate("site:styles/styles.css");  
// show the file creation day - example: "24"  
alert("File was created on day " + d.getDate());  
// show the year - example: "2011"  
alert("File was created in " + d.getFullYear());  
// show UTC time - example: "Sun, 2 Jan 2011 15:23:54 UTC"  
alert("UTC creation: " + d.toUTCString());
```

xweb.file.getModificationDate Method

The `getModificationDate` method returns the last modified date of the specified file.



note

For a full reference on the JavaScript `Date` object, see http://www.w3schools.com/jsref/jsref_obj_date.asp.

Syntax

```
xweb.file.getModificationDate(path);
```

Parameters

The `path` parameter is a string that specifies the path of the file relative to the root of the site. It must be preceded with `site:`.

Returns

Returns a JavaScript `Date` object. If the specified file is not found, returns `undefined`.

Information

See `getCreationDate` for more information on the JavaScript `Date` object.

xweb.file.getSize Method

The `getSize` method gets the file size of the specified file.

Syntax

```
xweb.file.getSize(path);
```

Parameters

The `path` parameter is a string that specifies the path to the file relative to the root of the site. It must be preceded with `site:`.

Returns

Returns an integer representing the size of the specified file in bytes.

Information

The size returned is the size on disk and doesn't take unsaved modifications into account.

xweb.file.listFolder Method

The `listFolder` method returns a list of files or folders in the site that match a specified pattern.

Syntax

```
xweb.file.listFolder(path, type);
```

Parameters

The `path` parameter is a string that specifies the path of the file or folder relative to the root of the site. It must be preceded with `site:.`

The `type` parameter specifies whether to return files or folders that match the `path` parameter. The following values are valid for the `type` parameter.

- `files`—Returns files that match the specified path.
- `directories`—Returns folders that match the specified path.

Returns

Returns an array of strings containing the name(s) of files or folders that match the specified path.

Information

You can use the following wildcard characters to define a matching pattern for the `path` parameter:

- `*`—Matches one or more instances of any character.
- `?`—Matches a single instance of a character.

The `listFolder` method is not recursive. Only the files and folders in the folder specified by the `path` parameter are returned.

Example

The following example matches all files with an “i” as the second character in the root folder of the site:

```
xweb.file.listFolder("site:i*", "files");
```

xweb.file.read Method

The `read` method reads the specified file.

Syntax

```
xweb.file.read(path);
```

Parameters

The `path` parameter is a string that specifies the path to the file relative to the root of the site. It must be preceded by `site:`.

Returns

Returns a string that contains the entire contents of the file.

Information

While the `read` method can be used on non-text files (such as binary files), it's only useful for textual files.

xweb.file.setAttributes Method

The `setAttributes` method sets attributes on the specified file or folder.

Syntax

```
xweb.file.setAttributes(path, attribute);
```

Parameters

The `path` parameter is a string that specifies the path to the file or folder relative to the site root. It must be preceded by `site:`.

The `attribute` parameter is a case-sensitive string and must be one of the following:

- `R`—Makes the file or folder read-only.
- `W`—Makes the file or folder read/write.
- `H`—Makes the file or folder hidden.
- `V`—Makes the file or folder visible.
- `RH`—Makes the file read-only and hidden.
- `RV`—Makes the file read-only and visible.
- `WH`—Makes the file read/write and hidden.
- `WV`—Makes the file read/write and visible.

Returns

Returns `true` if the attribute was successfully set. Otherwise, returns `false`.

Example

The following code example makes the `scripts/myscript.js` file read-only and hidden.

```
xweb.file.setAttributes("site:scripts/myscript.js", "RH");
```

xweb.file.write Method

The `write` method writes the specified string to the specified file.

Syntax

```
xweb.file.write(path, string);
```

Parameters

The `path` parameter is a string that specifies the path to the file or folder relative to the site root. It must be preceded by `site:`.

The `string` parameter is the string to be written to the file.

Returns

Returns `true` if the string was successfully written to the file. Otherwise, returns `false`.

Information

It is possible to use the `write` method to write string content to a binary file. However, doing so will almost certainly corrupt the file.

HTMLELEMENT Object

The `htmlElement` object represents an HTML element in a document. The properties and methods of the `htmlElement` object can be used to examine and manipulate the content in a document.



tip

The properties and methods of the `xweb.document` object can be used to obtain a reference to an `htmlElement` object.

htmlElement.childNodes Property

The `childNodes` property returns an array of an element's first-level child nodes. If the element has no child nodes, an empty array is returned.

Syntax

```
htmlElement.childNodes;
```

Returns

The `childNodes` property only returns first-level children of the element.

Information

The `childNodes` property is read-only.

Example

In the following example code, only the `ul` element is returned by the `childNodes` property. None of the `li` elements is returned.

```
<div id="ListDiv">
  <ul>
    <li>Item 1</li>
    <li>Item 2</li>
  </ul>
</div>
<script type="text/javascript">
var ele = xweb.document.getElementById("ListDiv");
var children = ele.childNodes; // only the ul element is returned.
</script>
```

htmlElement.className Property

The `className` property returns the value of the `class` attribute of the element.

Syntax

```
htmlElement.className;
```

Returns

Returns the value of the `class` attribute as a string. If the element does not have a `class` element, `className` returns `undefined`.

Information

The `className` property is read-only.

htmlElement.id Property

The `id` property returns the value of the `id` attribute of the element.

Syntax

```
htmlElement.id;
```

Returns

Returns the value of the `id` attribute as a string. If the element does not have an `id` attribute, `id` returns `undefined`.

Information

The `id` property is read-only.

htmlElement.innerHTML Property

The `innerHTML` property returns or sets the HTML code that is between the start and end tags of the element.

Syntax

```
htmlElement.innerHTML;
```

Returns

Returns the HTML code of the element as a string. If the element contains no HTML code, an empty string is returned.

When used to set the inner HTML of an element, `innerHTML` returns the newly set HTML as a string.

Information

The `innerHTML` property is a read/write property.



caution

HTML code that you add to a document programmatically is not checked for validity.

Example

The following code example illustrates the use of the `innerHTML` property.

```
<div id="content">  
  <h1>Welcome</h1>  
  <p>Welcome to our site.</p>  
</div>  
<script type="text/javascript">  
  var contentDiv = xweb.document.getElementById("content");  
  alert(contentDiv.innerHTML);  
  contentDiv.innerHTML = "<h1>Welcome Home</h1>";
```

```
    alert(contentDiv.innerHTML);  
</script>
```

When the preceding code runs, the first alert displays the following HTML:

```
<h1>Welcome</h1>  
<p>Welcome to our site.</p>
```

The `innerHTML` property is then set to new HTML. The second alert displays the following:

```
<h1>Welcome Home</h1>
```

After this code runs, the `p` element is removed from the page, and the text displayed inside the `h1` heading is changed.

htmlElement.innerHTML Property

The `innerHTML` property is similar to the `innerHTML` property, but the `innerHTML` property returns text only and not HTML.

Syntax

```
htmlElement.innerHTML;
```

Returns

Returns the inner text of the element as a string. If the element has no inner text, an empty string is returned.

When used to set the inner text of an element, `innerHTML` returns the newly set inner text as a string.

Information

If you use the `innerHTML` property to assign HTML code to an element, the HTML code is encoded and appears as regular text in the document.

The `innerHTML` property is a read/write property.

Example

The following code example illustrates the use of the `innerHTML` property.

```
<div id="content">  
  <h1>Welcome</h1>  
</div>  
<script type="text/javascript">  
  var contentDiv = xweb.document.getElementById("content");
```

```
alert(contentDiv.innerText);
contentDiv.innerText = "Welcome Home";
alert(contentDiv.innerHTML);
contentDiv.innerText = "<h1>Welcome Home</h1>";
alert(contentDiv.innerText);
</script>
```

In this example, the first alert displays the following:

```
Welcome
```

Only the text of the `h1` element is returned. The actual HTML is ignored.

After the `innerText` property is set to `Welcome Home`, an alert is used to display the `innerHTML` property of the `div`. That alert displays the following.

```
Welcome Home
```

By setting the `innerText` property of the `div`, the `h1` element has been removed. The `innerText` property does not recognize HTML code.

Next, the `innerText` property is set using `<h1>Welcome Home</h1>`. The `innerText` property of the `div` is then displayed in an alert. That alert displays the following:

```
&lt;h1&gt;Welcome Home&lt;/h1&gt;
```

htmlElement.nextSibling Property

The `nextSibling` property returns the next `htmlElement` at the same DOM level as the element.

Syntax

```
htmlElement.nextSibling;
```

Returns

Returns an `htmlElement` representing the next element in the DOM that is at the same level. If there are no other elements at the same level in the DOM, `nextSibling` returns `undefined`.

Information

Experienced JavaScript developers may have encountered situations where using the `nextSibling` property in the DOM outside Expression Web returns whitespace that exists between elements. The `htmlElement.nextSibling` property in Expression Web's JavaScript API does not consider whitespace to be elements.

The `nextSibling` property is read-only.

Example

In the following code example, `nextSibling` returns the link element.

```
<title>My Page</title>
<link rel="stylesheet" type="text/css" href="styles.css" />
<script type="text/javascript">
function DoSomething()
{
  var title = xweb.document.getElementsByTagName("title")[0];
  var sib = title.nextSibling; // returns link element
  // do something with elements
}
</script>
```

htmlElement.outerHtml Property

The `outerHtml` property returns or sets the HTML code of the element.

Syntax

```
htmlElement.outerHtml;
```

Returns

Returns the HTML code for the element, including the HTML elements that comprise the start and end tags.

When used to set the outer HTML for an element, returns the newly set HTML code as a string.

Information

The `outerHtml` property is a read/write property.

Example

The following example illustrates the use of the `outerHtml` property.

```
<div id="content">
  <h1>Welcome</h1>
  <p>Welcome to our site.</p>
</div>
<script type="text/javascript">
  var contentDiv = xweb.document.getElementById("content");
  alert(contentDiv.outerHtml);
  // change the HTML
  contentDiv.outerHtml = "<div id=\"content\">\n<h1>Welcome!</h1>\n</div>";
  alert(contentDiv.outerHtml);
```

The first alert from the preceding code example displays the following:

```
<div id="content">
  <h1>Welcome</h1>
  <p>Welcome to our site.</p>
</div>
```

Notice that the `outerHtml` property contains the entire HTML content, including the start and end tags. After the `outerHtml` property is set to a new value, the complete `div` tag is replaced with the new content.

htmlElement.parentNode Property

The `parentNode` property returns parent node of the element.

Syntax

```
htmlElement.parentNode;
```

Returns

Returns the parent node of the element as an `htmlElement` object. If the element has no parent nodes, returns `undefined`.

Information

The `parentNode` property is read-only.

Example

In the following code example, the `parentNode` property returns the `div` element.

```
<div>
  <p id="heading">Conference Dates</p>
</div>
<script type="text/javascript">
  var ele = xweb.document.getElementById("heading");
  var parent = ele.parentNode;
</script>
```

htmlElement.previousSibling Property

The `previousSibling` property returns the previous `htmlElement` at the same DOM level as the element. The `previousSibling` is used in the same fashion as the `nextSibling` method.

Syntax

```
htmlElement.previousSibling;
```

Returns

Returns the previous element at the same level in DOM as an `htmlElement` object.

If the element has no previous siblings, `previousSibling` returns `null`.

Information

The `previousSibling` property is read-only.

htmlElement.tagName Property

The `tagName` property returns the element's HTML tag name.

Syntax

```
htmlElement.tagName;
```

Returns

Returns the name of the element's tag as a string.

Information

The string that is returned does not contain the opening or closing brackets. Only the tag name is returned. For example, if the element is an anchor element, the `tagName` property returns `a`.

The `tagName` property is read-only.

htmlElement.getAttribute Method

The `getAttribute` method returns the value of the element's specified attribute.

Syntax

```
htmlElement.getAttribute(attribName);
```

Parameters

The `attribName` parameter specifies the name of the attribute to return.

Returns

Returns the value of the specified attribute as a string. If the specified attribute does not exist, returns undefined.

Example

```
<div id="content" style="font-size:.75em">
  <p>Hello</p>
</div>
<script type="text/javascript">
  var ele = xweb.document.getElementById("content");
  alert(ele.getAttribute("style"));
</script>
```

When the preceding code example runs, the alert box displays the following:

```
font-size:.75em
```

htmlElement.removeAttribute Method

The `removeAttribute` method removes the specified attribute from the element.

Syntax

```
htmlElement.removeAttribute(attribName);
```

Parameters

The `attribName` parameter specifies the attribute name to remove from the element.

Returns

The `removeAttribute` method returns `true` if the attribute is found and removed and returns `false` otherwise.

htmlElement.setAttribute Method

The `setAttribute` method sets an attribute name and value of the element.

Syntax

```
htmlElement.setAttribute(name, value);
```

Parameters

The `name` parameter specifies the name of the attribute to set. The `value` parameter specifies the value for the attribute. The `value` parameter can be an empty string or `null`.

Returns

The `setAttribute` returns `true` if the attribute is set. Otherwise, it returns `false`.

Information

If the specified attribute already exists, the value of the existing attribute will be changed to the value specified by the `value` parameter. If the specified attribute doesn't exist, it will be added to the element.

XWEB . DOCUMENT . SELECTION Object

The `selection` object represents a selection within a document. The `selection` object exposes properties and methods that allow for interacting and modifying the document.

`selection.end` Property

The `end` property identifies the end of the selection in the document.

Syntax

```
selection.end;
```

Returns

Returns an integer that represents the number of characters from the first character in the document to the last character in the selection.

Information

The `end` property is read-only.

`selection.start` Property

The `start` property identifies the start of the selection in the document.

Syntax

```
selection.start;
```

Returns

Returns an integer that represents the number of characters from the first character in the document to the first character in the selection.

Information

The `start` property is read-only.

`selection.text` Property

The `text` property retrieves or sets the text of the selection.

Syntax

```
selection.text;
```

Returns

Returns the selected text as a string. If nothing is selected, returns an empty string.

Information

The `text` property is read/write.

Example

The following example illustrates the use of the `text` property. In the example, the following HTML code is in the document.

```
<!-- Content Paragraph -->  
<p>This is content.</p>
```

The word "Content" in the HTML comment is selected.

```
alert(xweb.document.selection.text); // displays "Content"  
xweb.document.selection.text = "Main";  
// replaces "Content" with "Main" and returns "Main"
```

`selection.append` Method

The `append` method adds text to the end of the selection.

Syntax

```
selection.append(text);
```

Parameters

The `text` parameter is the text to add to the end of the selection.

Returns

Returns `true` if the text was successfully added. Otherwise, returns `false`.

Information

The `append` method can be used to add text, HTML code, JavaScript code, and so on.

`selection.insert` Method

The `insert` method inserts text in front of the selection.

Syntax

```
selection.insert(text);
```

Parameters

The `text` parameter represents the string to be inserted at the beginning of the current selection. If nothing is selected, the text is inserted at the insertion point.

Returns

Returns `true` if the text was successfully inserted. Otherwise, returns `false`.

Information

Like the `append` method, the `insert` method can be used to insert text or code.

`selection.set` Method

The `set` method sets the start and end position of the selection.

Syntax

```
selection.set(start, end);
```

Parameters

The `start` parameter is an integer that represents the number of characters from the beginning of the document to the start of the selection.

The `end` parameter represents the number of characters from the beginning of the document to the end of the selection.

Returns

Returns `true` if the selection was successfully set. Otherwise, returns `false`.

Information

If the value of the `end` parameter is greater than the number of characters in the document, the selection will be set from the character indicated by the `start` parameter to the end of the document.

`selection.remove` Method

The `remove` method removes the selection from the document.

Syntax

```
selection.remove();
```

Parameters

None.

Returns

Returns `true` if the selection was successfully removed. Otherwise, returns `false`.

`selection.replace` Method

The `replace` method replaces the current selection with the specified text.

Syntax

```
selection.replace(text);
```

Parameters

The `text` parameter is a string that represents the new text for the selection.

Returns

Returns `true` if the selection was successfully replaced. Otherwise, returns `false`.

Information

The `replace` method performs the same operation as setting the `text` property of the selection.

selection.wrap Method

The `wrap` method wraps the selection with the specified start and end text.

Syntax

```
selection.wrap(startText, endText);
```

Parameters

The `startText` parameter is the text to be inserted before the selection.

The `endText` parameter is the text to be inserted after the selection.

Returns

Returns `true` if the selection was successfully modified. Otherwise, returns `false`.

Information

The `wrap` method is often used to wrap a selection with specific HTML tags.

Example

This code example makes the selection a hyperlink:

```
selection.wrap("<a href=\"http://www.jimcobooks.com\">", "</a>");
```

This page intentionally left blank

INDEX

_vti folders, displaying, 26-27
<div> tag, adding to ASP.NET
Ajax pages, 549
<table> tag
align attribute, 88-90
cellpadding attribute, 91-92
cellspacing attribute, 92-93
frame attribute, 93
<td> tag, colspan attribute,
94-95

A

absolute positioning, 284
layers, 378-380
AccessDataSource control
(ASP.NET), 590-591
accessibility, 199
frames, 204
hyperlinks, 202-203
tables, 203
WCAG, 202
Accessibility Checker,
16, 205-206
Accessibility panel, 206-207
accessibility reports, 208-209
accessing Page Editor Options
dialog, 185
Add-In Builder, creating add-in
manifest, 662-666
add-in manifest, 618-622
creating with Add-In
Builder, 662-666
editing with Add-In Builder,
667
adding
Ajax functionality to ASP.
NET pages, 543-548
interactivity to layers,
388-393
add-ins, 23, 613-616
adding functionality with
JavaScript, 673-682
commands, 622-624
composition of, 616-617
debugging, 691-698
debug consoles, 694-695
Extensibility Tester,
691-693
Visual Studio, 695-698
dialog boxes, 622-624
installation package,
creating, 687-689
JavaScript, 616-617
managed classes,
accessing, 682-687
menus, 624-627
internal ID values, 631
panels, 627-630
planning, 659-661
testing, 689-691
toolbars, 624-627
internal ID values, 650
user interfaces, creating,
667-673
adjusting
code formatting, 81-83
page size, 78-79
table width, 103-104
Adobe Photoshop files,
importing, 167-168
AdRotator control (ASP.NET),
419-426
Advanced tab (Page Properties
dialog), 62
Advanced tab, Site Settings
dialog, 257-258
Ajax, 538
Ajax Control Toolkit, 543
align attribute, <table> tag,
88-90
aligning
cell content, 100-103
aligning cell content, 95-96
Apply Styles panel, 14-15
CSS styles, applying,
292-293, 304-307
applying
HTML rules to find and
replace, 179-180
multiple CSS classes,
289-290
arranging CSS styles, 315-316
ASP.NET, 22, 410-412
Ajax functionality, adding,
543-548
contact forms, creating,
606-608
Content Pages, 464-467
controls, 414-418, 419
AdRotator control,
419-426
Calendar control, 426-430
CreateUserWizard,
488-489
formatting, 439
Login control, 478-481
LoginStatus control,
482-487
LoginView control,
489-491
Menu control, 444-450

PasswordRecovery control, 487-488
properties, 415-418
SiteMapPath control, 457-459
TreeView control, 451-456
user controls, 475-476
Wizard control, 431-439

data, displaying in tabular form, 592-596

data source controls, 589-592

detail view, creating, 601-603

editing data with, 597-599

email
confirmation pages, displaying, 613
contact forms, 602-605
sending, 608-613
validation controls, 608-610

GridView control, editing data, 597-599

history of data access, 585-587

language, selecting, 412

login system, creating, 476-477

Master Pages, 459-463

master view, creating, 600-601

Menu control, 444-450

pages, creating, 49-50, 412-413

sitemap files, 588
creating, 443-444

user controls, creating, 524-525

validated forms, creating, 515-521

validation controls, 510-512
properties, 513-514

validation groups, 521-522

Web Parts, 522

Web Parts controls, editing, 537-538

Web Parts page
creating, 526-528
display modes, 529-534

ASP.NET Ajax, 538-543

Ajax Control Toolkit, 543

client-side Ajax, 542

<div> tag, adding, 549

client library, creating, 549-550

client script, adding to *ScriptManager* control, 550-552

ScriptManager control, adding, 546-548

server-side Ajax, 542-543

UpdatePanel control, adding, 548

attaching Dynamic Web Templates

to existing page, 322-323

to new page, 324

attributes (tags), 126

setting, 128-130

attributes, add-in manifest, 618-622**Authoring tab (Page Editor Options dialog), 193-194****auto thumbnails, configuring, 155****AutoThumbnail tab (Page Editor Options dialog), 191**

B

behaviors, 19, 344

adding within paragraphs, 344-345

Call Script behavior, 345-346

Change Property behavior, 346-349

Change Property Restore behavior, 349

Go To URL, 349-350

Jump Menu behavior, 350

Jump Menu Go behavior, 350-351

layer properties, configuring visibility, 387-389

Open Browser Window behavior, 351-352

Popup Message behavior, 353

Preload Images behavior, 353

Set Text behavior, 354-356

Swap Image behavior, 357

Swap Image Restore behavior, 357-358

when to use, 358

Behaviors panel, 341**Berners-Lee, Tim, 209****bookmarks, 84****Bradbury, Nick, 132****bread crumb navigation system, 443****browser compatibility, 209-210****Browser Size drop-down (SuperPreview), 224****building layouts with SuperPreview, 230-232****Button tab (Interactive Button dialog), 334-335**

buttons, interactive buttons, 19, 332-334
 editing, 340

C

Calendar control (ASP.NET), 426-430**Call Script behavior, 345-346****cascading order, 300****cellpadding attribute, <table> tag, 91-92****cells**

content, aligning, 100-103

merging, 99-101

cellspacing attribute

<table> tag, 92-93

content, aligning, 95-96

centering DIV with CSS, 292**Change Property behavior, 346-349****Change Property Restore behavior, 349****checking accessibility, 205-206**

- child elements, 617**
- child layers, 383-386**
- classes, CSS, 287-289**
 - multiple, applying, 289-290
- client library, creating for ASP.NET Ajax pages, 549-550**
- client script, adding to ScriptManager control, 550-552**
- client-side Ajax, 542**
- Code Block (PHP scripting), 575-576**
- Code Formatting tab (Page Editor Options dialog), 191**
- code snippets, 68**
 - inserting, 65
- Code Snippets tab (Page Editor Options dialog), 195**
- Code View, 79-84**
 - bookmarks, 84
 - code formatting, adjusting, 81-83
 - Dynamic Web Templates
 - code, viewing, 331-332*
 - modifying attached pages, 325-327*
 - IntelliSense, 84
 - PHP, syntax highlighting, 568-569
 - Quick Tags tools, 83-84
- Color Coding tab (Page Editor Options dialog), 193**
- colspan attribute, <td> tag, 94-95**
- COM (Component Object Model), 613**
- command prompt, starting Microsoft Expression Development Server, 585**
- commands, 622-624**
- Comment item (PHP scripting), 576-577**
- Common Language Runtime, 581**
- comparing**
 - CSS and HTML, 274-276
 - Master Pages and Dynamic Web Templates, 466
- comparison browser selectors (SuperPreview), 222-223**
- compatibility, 209**
 - browser compatibility, 209-210
 - Expression Web features
 - code problems, identifying, 210-213*
 - marking invalid code, 213 reports, 214-216*
- Compatibility Checker, 16**
- compliance with web standards, 131-132**
- composition of add-ins, 616-617**
- configuring**
 - auto thumbnails, 155
 - file editors, 65-67
 - frames, 112-114
 - inline frames, 117-118
 - Microsoft Expression Development Server, 560-561
 - PHP, IntelliSense, 570-572
 - publishing destinations, 241-243
 - reports, 261-263
 - spell checking, 59
- Confirmation Page tab (Form Properties dialog), 404-406**
- confirmation pages, displaying with ASP.NET, 613**
- contact forms, ASP.NET, 602-605**
 - creating, 606-608
- content**
 - adding to layers, 381-386
 - formatting with CSS, 278-284
 - positioning with CSS, 284-287
 - publishing, 232-236
 - server options, 236-240*
- Content Pages (ASP.NET), 464-467**
- context menu (Code View), 84**
- controlling positioning, 142**
- controls**
 - ASP.NET, 414-418, 419
 - AdRotator control, 419-426*
 - Calendar control, 426-430*
 - formatting, 439*
 - Menu control, 444-450*
 - properties, 415-418*
 - SiteMapPath control, 457-459*
 - TreeView control, 451-456*
 - Wizard control, 431-439*
 - Login control, 478-481
 - LoginStatus control, 482-487
 - LoginView control, 489-491
 - PasswordRecovery control, 487-488
 - user controls, 475-476
 - creating, 524-525*
 - Web Parts controls, 524-526
 - editing, 537-538*
- converting image formats, 151-153**
- Cookie variable PHP scripting, 574-575**
- CreateUserWizard control (ASP.NET), 488-489**
- creating**
 - add-in manifest with Add-In Builder, 662-666
 - add-ins, user interface, 667-673
 - ASP.NET contact forms, 606-608
 - ASP.NET pages, 412-413
 - child layers, 383-386
 - CSS style sheets, 279
 - Dynamic Web Templates,
 - layout page, 319
 - forms, 396-400
 - framesets, 110-112
 - image maps, 156-157
 - login systems, 476-477
 - managed classes, 683-686
 - Master Page site, 459
 - pages

- ASP.NET pages, 49-50
 - CCS layouts, 49-51
 - frame pages, 51
 - general pages, 46-49
 - hyperlinks, 55-58
 - sitemap files, 443-444
 - sites, 9-10
 - with SSL, 35-38
 - style sheets, 51
 - thumbnails, 153-155
 - user controls (ASP.NET), 524-525
 - Web Packages, 264-270
 - Web Parts page, 526-528
- CSS, 272-274**
- cascading order, 300
 - classes, 287-289
 - multiple, applying, 289-290
 - content
 - formatting, 278-284
 - positioning, 284-287
 - display property, comparing with visibility property, 675
 - DIV, centering, 292
 - embedded style sheets, 239-240
 - errors, checking for, 313-315
 - external style sheets, 277
 - IDs, 300
 - inheritance, 300
 - inline styles, 278
 - layouts, creating, 49-51
 - pseudo-classes, 290-291
 - pseudo-elements, 291
 - purpose of, 274-276
 - styles, applying
 - Apply Styles panel, 292-293, 304-307
 - Link Style Datasheet dialog, 295
 - Manage Styles panel, 293-295, 299-305
 - Style Builder, 297-298, 310-312
 - text, formatting, 52-55
- CSS Properties panel, 307-310**
- CSS styles, applying, 295
- CSS reports, 17-18, 313-315**

- CSS tab (Page Editor Options dialog), 191-193**
- CSS tools (Style Builder), 13-14**
- custom panel page size, creating, 668**
- Custom tab (Page Properties dialog), 62**
- customizing tables, 97-99**

D

- data access**
 - history of, 587
 - technologies in Expression Web, 588
- data access features, 23**
- data source controls (ASP.NET), 589-592**
- databases**
 - membership database, publishing with Web Deploy, 507-510
 - saving form results to, 407-410
- debug consoles, debugging add-ins, 694-695**
- debugging**
 - add-ins, 691-698
 - debug consoles, 694-695
 - Extensibility Tester, 691-693
 - Visual Studio, 695-698
 - JavaScript, 377-378
- Deep Zoom images, 21**
 - inserting, 161-162
- Default Fonts tab (Page Editor Options dialog), 191**
- deleting**
 - frames, 114
 - recent searches, 183-185
 - rows from tables, 106
- deprecated HTML, 90**
- design surface, 8**
- Design View, 70-79**
 - images, tracing, 76-78
 - page size, adjusting, 78-79
 - PHP, displaying formatting marks, 566-567
 - Ruler and Grid feature, 74-77
 - visual aids, 70-74
- destination**
 - for imported sites, specifying, 44-45
 - publishing, configuring, 241-243
- detaching Dynamic Web Templates, 330**
- detail view, creating with ASP.NET, 601-603**
- dialog boxes, 622-624**
- disk-based sites, 29-32**
- display modes for Web parts page, 529-534**
- display property, comparing with visibility property, 675**
- displaying**
 - _vti folders, 26-27
 - ASP.NET data in tabular form, 592-596
 - PHP information in Design View, 577-578
- document object (DOM), 365-366**
- DOM (Document Object Model), 363-366**
 - document object, 365-366
 - window object, 364-365
- DOM Highlighting, 220-221**
- dynamic content warning, troubleshooting Microsoft Expression Development Server, 582-583**
- Dynamic DNS, 254**
- Dynamic Web Templates, 11, 316-318**
 - attached pages, modifying in Code View, 325-327
 - attaching
 - to existing page, 322-323
 - to new page, 324
 - code, viewing in Code View, 331-332

detaching, 330
 editable regions, adding,
 327-328
 layout page
 creating, 319
 editable regions, adding,
 319
 modifying, 324-325

E

editable regions

adding to Dynamic Web
 Templates, 327-328
 renaming, 328-329

editing

add-in manifest with Add-In
 Builder, 667
 ASP.NET data, 597-599
 interactive buttons, 340
 page content with Quick Tag
 Editor, 136-142
 recent searches, 183-185
 tags, 137-138
 properties, 142
 Web Parts controls, 537-538

elements of add-in manifest, 618-622

email, ASP.NET

confirmation pages,
 displaying, 613
 contact forms,
 602-605, 606-608
 sending, 608-613
 validation controls, 608-610

Email Results tab (Form Properties dialog), 402-404

embedded style sheets, 239-240

tables, formatting, 103

enabling PHP for IIS using FastCGI, 561-562

errors (CSS), checking for, 313-315

events, 126, 130-131

Expression Media Encoder, 21

Extensibility Tester, debugging add-ins, 691-693

external style sheets, 277

F

FastCGI, 557

enabling PHP for IIS, 561-562
 installing
 in Vista and Windows,
 558-559
 in Windows Server 2008,
 559

features

add-ins, 23
 data access, 23
 PHP support, 22

file editors, configuring, 65-67

File Results tab (Form Properties dialog), 401-402

file system method, importing sites, 42

files, importing, 51

Find and Replace tool, 11

find and replace tool, 169-171

HTML rules, applying,
 179-180
 HTML tags, finding and
 replacing, 181-182
 queries, saving, 182-183
 recent searches, removing,
 183-185
 regular expressions, 172-173
 text
 finding, 173-176
 replacing, 176-178
 troubleshooting, 174-175

Flash movies, inserting, 157-159

Folder list, 7-8

Font Families tab (Page Editor Options dialog), 197

Font tab (Interactive Button dialog), 335-337

fonts, formatting, 54-55

form controls, 396

form field validation, JavaScript, 373-377

Form Properties dialog

Confirmation Page, 404-406
 Email Results tab, 402-404
 File Results tab, 401-402
 Saved Fields tab, 406-407

Form variable, PHP scripting, 572-573

formatting

ASP.NET controls, 439
 of code, adjusting, 81-83
 content with CSS, 278-284
 images, 148-151
 tables, embedded style
 sheets, 103
 text, 52-55
 fonts, 54-55

formatting marks (PHP), displaying in Design View, 566-567

Formatting tab (Page Properties dialog), 62

forms

creating, 396-400
 hidden fields, 410
 HTML forms, 393-396
 results
 saving to database,
 407-410
 saving to file, 400-401
 validated forms, creating,
 515-521

frame attribute, <table> tag, 93

frame pages, creating, 51

frames

accessibility, 204
 alternative content, creating,
 114-115
 borders, 119
 breaking out of, 121
 configuring, 112-114
 deleting, 114
 inline frames, inserting,
 117-118
 resizable, 120-121
 splitting, 113-114

targeting, 115-116
 when not to use, 110
 when to use, 107

framesets, creating, 110-112

FrontPage Server Extension, 237

sites, importing, 40-42

FTP, importing sites, 41-39

FTP sites, 32-34

FTPS, importing sites, 41-39

functions, PHP, 556-557

G

general pages, creating, 46-49

General tab

Page Editor Options dialog, 185-190

Page Properties dialog, 60-62

Site Settings dialog, 255-256

generating previews, 225-226

with SuperPreview, 218-219

GIF format, 143-146

transparency, setting, 152

Go To URL behavior, 349-350

GridView control (ASP.NET),

editing data with, 597-599

H

hidden form fields, 410

hierarchical menu systems, 420

history of browser scripting, 358-360

history of data access, 585-587

HomeSite, 132

hotspots, creating on images, 156

HTML

<table> tag

align attribute, 88-90

cellpadding attribute,

91-92

cellspacing attribute, 92-93

frame attribute, 93

<td> tag, colspan attribute, 94-95

versus CSS, 274-276

deprecated HTML, 90

inserting, 140

optimizing during

publishing, 244-246

publishing, troubleshooting, 246-253

tables

customizing, 97-99

inserting, 97

rows, adding and

deleting, 106

tags, finding and replacing, 181-182

HTML bookmarks, 57-58

HTML forms, 393-396

HTML rules

applying to find and replace, 179-180

htmlElement object

(JavaScript), 742-751

HTTP method, importing sites, 42-43

HTTP sites, 35-38

hyperlinks

accessibility, 202-203

creating, 55-58

parameters, 56-58

screentips, creating, 58

targeting, 56

I

IIS 5, configuring membership site, 492-495

IIS 6, configuring membership site, 492-495

IIS 7

installing in Vista and Windows, 558-559

membership site, configuring, 494-502

image maps, creating, 156-157

Image tab, Interactive Button dialog, 337-339

images

Adobe Photoshop files, importing, 167-168

format, converting, 151-153

formats, 143-146

inserting, 146-148

properties, changing, 149-151

resizing, 148

thumbnails, creating, 153-155

tracing, 76-78

Import Site Wizard

destination location,

specifying, 44-45

import method, selecting, 38-43

importing

Adobe Photoshop files, 167-168

files, 51

sites

file system method, 42

with FrontPage Server

Extension, 40-42

with FTP, 41-39

with FTPS, 41-39

HTTP method, 42-43

with SFTP, 41-39

with WebDAV, 42

Web Packages, 270-271

improving navigation with Master Pages, 459

Include Once script (PHP), 574-575

inline frames, inserting, 117-118

inline styles, 278

inserting

code snippets, 65

HTML, 140

images, 146-148

inline frames, 117-118

multimedia

Deep Zoom images, 161-162

Flash movies, 157-159
Silverlight applications,
 159-160
Windows Media, 163-167
 tables, 97

**installation packages, creating
 for add-ins**, 687-689

installing
 FastCGI in Vista and
 Windows, 558-559
 IIS 7 in Vista and Windows,
 558-559
 PHP, 557-562

IntelliSense, 84
 with PHP, 568-571

**IntelliSense tab (Page Editor
 Options dialog)**, 197

Interactive Button dialog
 Button tab, 334-335
 Font tab, 335-337
 Image tab, 337-339

interactive buttons, 19, 332-334
 editing, 340
 practical uses for, 341
 saving, 339-340

interactivity, adding to layers,
 388-393

interface, 9
 design surface, 8
 Folder list, 7-8
 panels, 5-7
 status bar, 9

internal ID values
 of add-in menus, 631
 of add-in toolbars, 650

**ISAPI (Internet Server
 Application Program
 Interface)**, 557

J

JavaScript, 363
 accessing and changing
 attributes, 371-373
 adding functionality to
 add-ins, 673-682
 adding to a page, 361-362

add-ins, 616-617
 conventions used in this
 book, 699-702
 debugging, 377-378
 form field validation, 373-377
 htmlElement object, 742-751
 linking to external script file,
 362-363
 managed classes, accessing,
 682-687
 page elements, showing
 and hiding, 366-371
 xweb.application object,
 702-712
 xweb.application.settings
 object, 712-714
 xweb.developer object,
 714-715
 xweb.document object,
 715-733
 xweb.document.selection
 object, 751-755
 xweb.file object, 733-742

JPEG format, 146

Jump Menu behavior, 350

Jump Menu Go behavior,
 350-351

L

**Language tab (Page Properties
 dialog)**, 63-65

layers, 19, 378-380
 child layers, 383-386
 content, adding, 381-386
 interactivity, adding, 388-393
 positioning, 386-387
 resizing, 382-383
 visibility, setting, 387-389
 z-order, 393

Layout Modes (SuperPreview),
 221

layout page
 adding editable regions, 319
 creating for Dynamic Web
 Templates, 319

layouts
 building with SuperPreview,
 230-232
 previewing with
 SuperPreview, 224-229

Licklider, J.C.K., 410

**limitations of Microsoft
 Expression Development
 Server**, 583-585

**Link Style Datasheet dialog,
 applying CSS styles**, 295

**linking JavaScript to external
 script file**, 362-363

Login control (ASP.NET),
 478-481

login systems, creating,
 476-477

LoginStatus control (ASP.NET),
 482-487

M

Manage Styles panel, 14-15
 CSS styles, applying,
 293-295, 299-305

managed classes
 accessing from JavaScript,
 682-687
 creating, 683-686

Master Page site, creating, 459

Master Pages, 459-463
 navigation, improving, 459

**master view, creating with
 ASP.NET**, 600-601

**membership database,
 publishing with Web Deploy**,
 507-510

membership site, creating,
 491-507

Menu control (ASP.NET),
 444-450

menus, 624-627
 add-ins, internal ID values,
 631

merging table cells, 99-101

metadata, 23-27**methods**

htmlElement object, 742-751
JavaScript

xweb.application object,
702-712
xweb.application.settings
object, 712-714
xweb.developer object,
714-715

xweb.document object,
715-733

xweb.document.selection
object, 751-755

xweb.file object, 733-742

Microsoft Expression**Development Server, 578-580**

configuring, 560-561
dynamic content warning,
troubleshooting, 582-583
how to use, 580-583
limitations of, 583-585
starting from command
prompt, 585

**mismatched editable regions,
resolving, 329-330****multimedia**

Deep Zoom images,
inserting, 161-162
Flash movies, inserting,
157-159
Silverlight applications,
inserting, 159-160
Windows Media, inserting,
163-167

**multiple CSS classes, applying,
289-290****N****navigational systems, 420-443**

ASP.NET, creating sitemap
files, 443-444

.NET Framework, 581**non-executing PHP code,
troubleshooting, 566****O****Open Browser Window
behavior, 351-352****optimization, SEO reports,
263-264****optimizing HTML during
publishing, 244-246****origins of tables, 86-87****P****Page Editor Options dialog**

accessing, 185
Authoring tab, 193-194
AutoThumbnail tab, 191
Code Formatting tab, 191
Code Snippets tab, 195
CSS tab, 191-193
Default Fonts tab, 191
General tab, 185-190
IntelliSense tab, 197
Picture tab, 195-196

**page elements, showing and
hiding with JavaScript,
366-371****Page Properties dialog**

Advanced tab, 62
Custom tab, 62
Formatting tab, 62
General tab, 60-62
Language tab, 63-65

page size, adjusting, 78-79**page transitions, 168-169****page views, 50**

Code View, 79-84
bookmarks, 84
code formatting,
adjusting, 81-83
context menu, 84
IntelliSense, 84
Quick Tags tools, 83-84

Design View, 70-79

image, tracing, 76-78
page size, adjusting,
78-79

Ruler and Grid feature,
74-77

visual aids, 70-74

Split View, 85-86

pages

ASP.NET, creating, 412-413

ASP.NET pages, creating,
49-50

CCS layouts, creating, 49-51

frame pages, creating, 51

general pages, creating,
46-49

hyperlinks, creating, 55-58
style sheets, creating, 51

panels, 5-7

Accessibility panel, 206-207
add-ins, 627-630

Apply Styles panel, 14-15
CSS styles, applying,
292-293, 304-307

Behaviors panel, 341

CSS Properties panel,
307-310

CSS styles, applying, 295

custom page size, creating,
668

interface, creating, 669-671

Manage Styles panel, 14-15
CSS styles, applying,
293-295, 299-305

SnapShot panel, 229-230
Tag Properties panel, 12,
121-126

events, 130-131

tag attributes, setting,
128-130

tag properties, viewing,
126-127

**paragraphs, adding behaviors,
344-345****parameters for hyperlinks,
56-58****PasswordRecovery control
(ASP.NET), 487-488**

perpetual script errors, debugging, 698

PHP

- in Code View, syntax highlighting, 568-569
- comments, 554
- formatting marks, displaying in Design View, 566-567
- functions, 556-557
- information, displaying, 577-578
- installing, 557-562
- IntelliSense, 568-572
 - configuring, 570-572*
- Microsoft Expression Development Server
 - configuring, 560-561*
- non-executing code, troubleshooting, 566
- pages, previewing, 563-566
- program flow, 555-556
- scripting
 - Code Block, 575-576*
 - Comment item, 576-577*
 - Cookie variable, 574-575*
 - Form variable, 572-573*
 - Include Once script, 574-575*
 - Session variable, 573-574*
 - URL variable, 573*
- support for, 22
- syntax, 552-557
- variables, 555

Picture tab (Page Editor Options dialog), 195-196

planning add-ins, 659-661

PNG format, 146

pointer modes (SuperPreview), 220

Popup Message behavior, 353

positioning

- content with CSS, 284-287
- controlling, 142
- layers, 386-387

Preload Images behavior, 353

Preview tab, Site Settings dialog, 256-257

Preview URL (SuperPreview), 221

previewing

- layout with SuperPreview, 224-229
- PHP pages, 563-566

previews

- generating, 225-226
- generating with SuperPreview, 218-219

program flow, PHP, 555-556

properties

- of ASP.NET controls, 415-418
- of Calendar control, 427-430
- htmlElement object, 742-751
- of images, changing, 149-151
- of validation controls, 513-514
- xweb.document object, 715-733
- xweb.document.selection object, 751-755

pseudo-classes (CSS), 290-291

pseudo-elements (CSS), 291

publishing, 232-236

- destination, configuring, 241-243
- files, synchronizing, 244
- HTML, optimizing, 244-246
- membership database with Web Deploy, 507-510
- server options, 236-240
- troubleshooting, 246-253

publishing sites, 10-11

Publishing tab, Site Settings dialog, 258-259

purpose of CSS, 274-276

Q

queries (find and replace tool), saving, 182-183

Quick Tag Editor

- page content, editing, 136-142
- when to use, 142-143

Quick Tag Selector, selecting elements, 134-135

Quick Tags tool, 13, 83-84, 132-134

- Quick Tag Editor
 - page content, editing, 136-142*
 - when to use, 142-143*
- Quick Tag Selector, selecting elements, 134-135

R

recent searches, editing, 183-185

regular expressions, find and replace tool, 172-173

remote browsers, 230-232

removing tags, 138-140

renaming editable regions, 328-329

rendering problems, troubleshooting in SuperPreview, 225-229

replacing text, 176-178

reports

- accessibility reports, 208-209
- configuring, 261-263
- CSS reports, 313-315
- saving, 263
- SEO reports, 263-264
- site reports, 10, 259-263

resizable frames, 120-121

resizing

- images, 148
- layers, 382-383

resolving mismatched editable regions, 329-330

rows, adding and deleting, 106

rowspan attribute, <td> tag, 94-95

Ruler and Grid feature (Design View), 74-77

Ruler and Grid tab (Page Editor Options dialog), 195-196

S

Saved Fields tab (Form Properties dialog), 406-407**saving**

- find and replace queries, 182-183
- form results to a database, 407-410
- form results to a file, 400-401
- interactive buttons, 339-340
- reports, 263

screeintips, hyperlink screeintips, 58**scripting**

- DOM, 363-366
 - document object*, 365-366
 - window object*, 364-365
- history of, 358-360
- JavaScript
 - accessing and changing attributes*, 371-373
 - adding functionality to add-ins*, 673-682
 - adding to a page*, 361-362
 - debugging*, 377-378
 - form field validation*, 373-377
 - htmlElement object*, 742-751
 - page elements, showing and hiding*, 366-371
 - xweb.application object*, 702-712
 - xweb.application.settings object*, 712-714
 - xweb.developer object*, 714-715
 - xweb.document object*, 715-733
 - xweb.document.selection object*, 751-755
 - xweb.file object*, 733-742
- perpetual script errors, debugging, 698
- PHP
 - Code Block*, 575-576
 - Comment item*, 576-577
 - Cookie variable*, 574-575

- Form variable*, 572-573
- Include Once script*, 574-575
- Session variable*, 573-574
- URL variable*, 573
- scripts, writing, 366-377

ScriptManager control

- adding to ASP.NET Ajax pages, 548
- client script, adding, 550-552

Section 508, 199**selecting**

- ASP.NET language, 412
- elements with Quick Tag Selector, 134-135
- type of site, 45-46

sending email with ASP.NET, 608-613**SEO Checker, 17****SEO reports, 263-264****server-side Ajax, 542-543****Session variable, PHP scripting, 573-574****Set Text behavior, 354-356****SFTP, importing sites, 41-39****Silverlight applications, inserting, 159-160****site optimization**

- Accessibility Checker, 16
- Compatibility Checker, 16
- CSS Reports, 17-18
- SEO Checker, 17
- SuperPreview, 17-18

site reports, 10, 259-263**Site Settings dialog**

- Advanced tab, 257-258
- General tab, 255-256
- Preview tab, 256-257
- Publishing tab, 258-259

sitemap files, 588

- creating, 443-444

SiteMapDataSources control (ASP.NET), 591**SiteMapPath control (ASP.NET), 457-459****sites, 23**

- creating, 9-10
 - with SSL*, 35-38
- disk-based, 29-32
- frames
 - when not to use*, 110
 - when to use*, 107
- FTP, 32-34
- HTTP, 35-38
- importing
 - file system method*, 42
 - with FrontPage Server Extension*, 40-42
 - with FTP*, 41-39
 - with FTPS*, 41-39
 - HTTP method*, 42-43
 - with SFTP*, 41-39
 - with WebDAV*, 42
- Master Page site, creating, 467-475
- membership site, creating, 491-507
- metadata, 23-27
- publishing sites, 10-11
- subsites, 27
- templates, 27-29
- type of, selecting, 45-46

SnapShot panel, 229-230**spell checking, 59****Split View, 85-86****splitting frames, 113-114****SqlDataSource control (ASP.NET), 591****SSL, creating sites, 35-38****standards, web standards compliance, 131-132****starting Microsoft Expression Development Server from command prompt, 585****status bar, 9****Style Builder, 13-14**

- CSS styles, applying, 297-298, 310-312

style sheets

- creating, 51, 279
- embedded style sheets, formatting tables, 103

subsites, 27

SuperPreview, 17-18, 216-218

- Browser Size drop-down, 224
- comparison browser selectors, 222-223
- DOM tab, 222-223
- interface
 - DOM Highlighting, 220-221*
 - Layout Modes, 221*
 - pointer modes, 220*
 - UI Helpers, 220*
- layouts
 - building, 230-232*
 - previewing, 224-229*
- Preview URL, 221
- previews, generating, 218-219, 225-226
- remote browsers, 230-232
- SnapShot panel, 229-230

Swap Image behavior, 357

Swap Image Restore behavior, 357-358

synchronizing published files, 244

syntax (PHP), highlighting in Code View, 568-569

T

tables

- accessibility, 203
- borders, 90-91
- cells
 - content, aligning, 95-96*
 - merging, 99-101*
- customizing, 97-99
- inserting, 97
- origins of, 86-87
- rows, adding and deleting, 106
- width, adjusting, 103-104

tabular form, displaying ASP.NET data in, 592-596

tag properties

- attributes, 126
- events, 126

viewing in Tag Properties panel, 126-127

Tag Properties panel, 12, 121-126

- events, 130-131
- web standards compliance, 131-132

tags

- attributes, setting, 128-130
- editing, 137-138
- PHP, 554
- properties, editing, 142
- removing, 138-140
- wrapping, 141-142

targeting

- frames, 115-116
- hyperlinks, 56

templates

- Dynamic Web Templates, 11, 316-318
 - attached pages, modifying in Code View, 325-327*
 - attaching to existing page, 322-323*
 - attaching to new page, 324*
- site templates, 27-29

testing add-ins, 689-691

text

- finding and replacing, 172-178
- formatting, 52-55
- fonts, 54-55

thumbnails, creating, 153-155

toolbars, 624-627

- add-ins, internal ID values, 650

Toolbox

- ASP.NET controls, 22
- form controls, 396
- Web Parts controls, 524-526

tracing images, 76-78

transparency of GIFs, setting, 152

TreeView control (ASP.NET), 451-456

troubleshooting

- find and replace tool, 174-175
- HTML publishing, 246-253
- Microsoft Expression Development Server, dynamic content warning, 582-583
- non-executing PHP code, 566
- page transitions, 169
- rendering problems in SuperPreview, 225-229
- unavailable Quick Tag tools, 134

type of site, selecting, 45-46

U

UI Helpers (SuperPreview), 220

unavailable Quick Tag tools, troubleshooting, 134

UpdatePanel control, adding to ASP.NET Ajax pages, 548

URL variable, PHP scripting, 573

user controls (ASP.NET), 475-476

user controls, creating, 524-525

user interfaces, creating, 667-673

V

validated forms, creating, 515-521

validation controls, ASP.NET, 510-512, 608-610

- properties, 513-514

validation groups, 521-522

variables, PHP, 555

video, inserting Silverlight video, 160-162

viewing

- Dynamic Web Template code, 331-332
- tag properties, 126-127

Vischeck, 216
visibility of layers, setting,
387-389
visibility property (CSS),
comparing with display
property, 675
Vista, installing IIS 7, 558-559
visual aids (Design View),
70-74
Visual Studio, debugging
add-ins, 695-698

W

WCAG (Web Content
Accessibility Guidelines), 202
web browsers, browser
compatibility, 209-210
Web Deploy, publishing
membership database,
507-510
web hosting, 253-254
Web Packages, 264
creating, 264-270
importing, 270-271
Web Parts, 522
Web Parts catalog, 534-536
Web Parts controls, 524-526
editing, 537-538
Web Parts page
creating, 526-528
display modes, 529-534
web standards compliance,
131-132
web-based forums, 272
WebDAV, importing sites, 42
width of tables, adjusting,
103-104
window object (DOM), 364-365
Windows Media, inserting,
163-167
Windows Server 2008,
installing IIS 7, 559

Wizard control (ASP.NET),
431-439
wrapping tags, 141-142
writing scripts, 366-377
WYSIWYG web design
products, 50

X

XML, 617-618
add-in manifest, 618-622
child elements, 617
XmlDataSource control
(ASP.NET), 592
XMLHttpRequest, 538
xweb.application object
(JavaScript), 702-712
xweb.application.settings
object (JavaScript), 712-714
xweb.developer object
(JavaScript), 714-715
xweb.document object,
715-733
xweb.document.selection
object (JavaScript), 751-755
xweb.file object, 733-742

Z

z-order of layers, 393

This page intentionally left blank

QUEPUBLISHING.COM

Your Publisher for Home & Office Computing

Quepublishing.com includes all your favorite—and some new—Que series and authors to help you learn about computers and technology for the home, office, and business.

Looking for tips and tricks, video tutorials, articles and interviews, podcasts, and resources to make your life easier? Visit quepublishing.com.

- **Read the latest articles and sample chapters** by Que's expert authors
- **Free podcasts** provide information on the hottest tech topics
- **Register your Que products** and receive updates, supplemental content, and a coupon to be used on your next purchase
- **Check out promotions and special offers** available from Que and our retail partners
- **Join the site** and receive members-only offers and benefits



QUE NEWSLETTER

quepublishing.com/newsletter



twitter.com/quepublishing



facebook.com/quepublishing



youtube.com/quepublishing



quepublishing.com/rss

Try Safari Books Online FREE for 15 days

Get online access to Thousands of Books and Videos



Safari[®]
Books Online

FREE 15-DAY TRIAL + 15% OFF*
informit.com/safaritrial

➤ Feed your brain

Gain unlimited access to thousands of books and videos about technology, digital media and professional development from O'Reilly Media, Addison-Wesley, Microsoft Press, Cisco Press, McGraw Hill, Wiley, WROX, Prentice Hall, Que, Sams, Apress, Adobe Press and other top publishers.

➤ See it, believe it

Watch hundreds of expert-led instructional videos on today's hottest topics.

WAIT, THERE'S MORE!

➤ Gain a competitive edge

Be first to learn about the newest technologies and subjects with Rough Cuts pre-published manuscripts and new technology overviews in Short Cuts.

➤ Accelerate your project

Copy and paste code, create smart searches that let you know when new books about your favorite topics are available, and customize your library with favorites, highlights, tags, notes, mash-ups and more.

* Available to new subscribers only. Discount applies to the Safari Library and is valid for first 12 consecutive monthly billing cycles. Safari Library is not available in all countries.



Adobe Press



Cisco Press



IBM Press

Microsoft Press



O'REILLY



SAMS



WILEY

