Contents

Debugging ASP.NET Applications

Debugging under classic ASP was a hit-and-miss affair, usually forcing the developer to add *Response.Write* statements through the code until he or she found the failure point. ASP.NET introduces much better debugging, thanks to the .NET Framework and Common Language Runtime (CLR).

Foreword xxv

Chapter 1 Introducing ASP.NET 1

Introduction 2
Learning from the History of ASP 2
  The Origins of ASP 2
    Why ASP Was Needed 3
    Why ASP Was Not Originally Embraced 4
  Developing ASP 1.x 5
  Developing ASP 2.x 6
    Major Changes with ASP 2 6
    Weaknesses in the ASP 2 Model 7
  Developing ASP 3.0 7
    Final Changes to Original ASP Model 8
    Weaknesses in the ASP 3 Model 8
    The Need for a New ASP Model 9
  The ASP Timeline 10
Reviewing the Basics of the ASP.NET Platform 11
  Utilizing the Flexibility of ASP.NET 12
    Converting Code into Multiple Languages 13
  Comparing Improvements in ASP.NET to Previous ASP Models 14
How Web Servers Execute ASP Files 15
  Client-Server Interaction 16
  Server-Side Processing 17
    Compiling and Delivering ASP.NET Pages 18
  Running ASP.NET Web Pages 19
    Obtaining and Installing .NET 19
    Creating Your First ASP.NET Application 20

Upgrading from Classic ASP 26
Taking Security Precautions 28
Summary 29
Solutions Fast Track 29
Frequently Asked Questions 32

Chapter 2 ASP.NET Namespaces 35
Introduction 36
Reviewing the Function of Namespaces 36
Using Namespaces 37
Using the Microsoft.VisualBasic Namespace 38
Understanding the Root Namespace: System 38
Supplied Functionality 38
Integral Numbers 39
Floating-Point Numbers 39
Dates 40
Strings 40
Booleans 40
Objects 40
Grouping Objects and Data Types with the System.Collections Namespace 43
Supplied Functionality 43
Enabling Client/Browser Communication with the System.Web Namespace 45
Supplied Functionality 45
System.Web.UI Namespace Set 46
System.Web.Services Namespace Set 51
Working with Data Sources Using the System.Data Namespace 52
Supplied Functionality 52
Processing XML Files Using the System.XML Namespace 53
Supplied Functionality 53
Summary 55
Solutions Fast Track 56
Frequently Asked Questions 58

Reviewing the Function of Namespaces

To use a namespace in an ASP.NET page, you must use the *Import* directive. Unlike in classic ASP, ASP.NET pages are compiled before they are run. You build ASP.NET pages using a compiled language, such as VB.NET or C#.

Chapter 3 ASP Server Controls 61

Introduction 62
Major Features of ASP.NET Server Controls 62
  Collecting Data Using HTML Forms 63
Server-Side Processing in ASP.NET 65
  A Simple Application Using Conventional HTML Controls 66
  A Simple Application Using ASP Server Controls 68
  Mapping Server Controls and Preserving Their States 69
  Including Scripts in an .aspx File 69
    Loading a List Box via Script 70
  Using the *IsPostBack* Property of a Page 72
  AutoPostBack Attributes of Server Controls 73
    Structure of an ASP.NET Web Form 75
    Page Directives 76
    The Order of Event Execution 77
Code-Behind versus In-Page Coding 77
  Using Code-Behind without Compilation 79
  Using Code Behind with Compilation 81
  Using VS.Net for Developing a Web Application 84
Using HTML Server Controls 87
  Using the *HtmlAnchor* Control 88
  Using the *HtmlTable* Control 88
  Using *HtmlInputText* and *HtmlTextArea* Controls 90
  Using *HtmlButton* and *HtmlImage* Controls 91
  Using the *HtmlInputFileControl* 93
  Using the *HtmlSelect* Control with Data Binding to a *SortedList* Structure 95
    Creating and Loading the *SortedList* 97
  Using *HtmlCheckBox* and *HtmlInputRadioButton* Controls 98
Using ASP.NET Web Controls 100

Developing ASP.NET Web Forms

When you develop an ASP.NET Web form, you can use the following type of controls:

- HTML Server Controls
- Web Server Controls (also known as Web Controls or ASP.NET Web Form Controls)
- Validation Controls
- Custom Controls

Basic Web Controls 101
Using *Labels*, *TextBoxes*, *RadioButtons*, *CheckBoxes*, and *DropDownLists* 103
Using the *ListControl* Abstract Class 106
Using *HyperLink* Controls 110
Binding a *ListControl* to an *ArrayList* 111
Validation Controls 113
The *RequiredFieldValidator* Control 114
The *RegularExpressionValidator* Control 115
The *CompareValidator* Control 117
The *RangeValidator* Control 118
The *CustomValidator* Control 118
CustomValidator with Explicit Client-Side Validation Function 120
Displaying the Error Message with Style 122
The *ValidationSummary* Control 123
Validating Patterned Strings, Passwords, and Dates 126
</form></body></html> The *Databound ListControls* Family 130
Using the *Repeater* Server Control 132
Using the *DataList* Control 139
Using the *DataGrid* Control 144
Providing Paging in *DataGrid* 152
Navigating to a Selected Page 154
Providing Data Editing Capability in a *DataGrid* Control 157
Creating Custom ASP Server User Controls 161
Creating a Simple Web User Control 161
Exposing Properties of a User Control 163
Developing the Payroll User Control 164
Consuming the Payroll User Control 166
Summary 168
Solutions Fast Track 168
Frequently Asked Questions 171

Chapter 4 Configuring ASP.NET **173**
Introduction 174
Overview of ASP.NET Configuration 174
Uses for a Configuration File 177
Application Configuration 179
Setting Static Variables Using the <appSettings> Tag 179
Providing Global Support Using the <globalization> Tag 180
Configuring Application Identity Using the <identity> Tag 181
Setting Page-Specific Attributes Using the <pages> Tag 181
Configuring the Tracing Service Using the <trace> Tag 183
System Configuration 184
Determining Client Capabilities Using the <browserCaps> Tag 184
Setting Compilation Options Using the <compilation> Tag 187
Controlling Connections Using the <connectionManagement> Tag 190
Defining Custom Errors Using the <customErrors> Tag 191
Mapping Requests Using the <httpHandlers> Tag 192
Configuring HTTP Modules Using the <httpModules> Tag 193
Setting Runtime Options Using the <httpRuntime> Tag 194
Setting Process Model Options Using the <processModel> Tag 195
Configuring the Session State Using the <sessionState> Tag 200
Configuring Request Modules Using the <webRequestModule> Tag 202

SECURITY ALERT!

With the standard ASP.NET *machine.config* file, all configuration files are secured and cannot be downloaded by a client system. This allows for some protection of critical information such as user IDs and passwords for DSN sources, but keep in mind that any system can be hacked with enough time and effort. Always keep security in mind when planning your Web application.

Configuring Web Services Using the <webServices> Tag 203
Security 204
Authenticating Users Using the <authentication> Tag 205
Configuring Security Modules Using the <authenticationModules> Tag 207
Controlling Access Using the <authorization> Tag 208
Configuring Encryption Keys Using the <machineKey> Tag 209
Mapping Security Policies Using the <securityPolicy> Tag 210
Applying Trust Levels Using the <trust> Tag 211
Anatomy of a Configuration File 211
Creating a Configuration File 215
Retrieving Settings 220
Summary 223
Solutions Fast Track 223
Frequently Asked Questions 224

Chapter 5 An ASP.NET Application 227
Introduction 228
Understanding ASP.NET Applications 228
Managing State 229
Analzying Global.asax 231
Understanding Application State 232
Using Application State 232
Application Cache Object 233
Static Variables 234
State Example 234
Using Application Events 236
Supported Application Events 236
More Events 237
Working with Application Events 238
Threading Use 239

Working with Application Events

To use application events in your project, you must do the following:

- Create a Web application folder using the MMC.
- Create a file called Global.asax in the directory you marked as an application.
- Within the Global.asax, enter script tags with the language you are using (e.g., VB).
- Insert subroutines using the name of the event you wish to use. Any code you add to this subroutine will run when the event fires.

Understanding Session State 240
Configuring Sessions 241
Using Session Events 243
Working with Session Events 245
Comparing Application and Session States 246
Static Values 249
Caching Data 252
Expiring the Cache 258
Summary 259
Solutions Fast Track 259
Frequently Asked Questions 262

Chapter 6 Optimizing Caching Methods 265
Introduction 266
Caching Overview 266
Output Caching 269
Using the @ *OutputCache* Directive 269
Using the *HttpCachePolicy* Class 275
Advantages of Using Output Caching 276
Fragment Caching 277
Advantages of Using Fragment Caching 281
Data Caching 281
Using the Cache Method 282
Using the *cache.add* and *cache.insert* Methods 285
Using the Dependency Option 285
Using the Expiration Policy Option 287
Using the Priority Options 288
Using the *CacheItemRemovedCallback* Delegate 289
</HTML>Using the *Cache.Remove* Method 292
Advantages of Using Data Caching 292
Best Uses for Caching 293
Output Caching 294
Fragment Caching 294
Data Caching 294

Answers to Your Frequently Asked Questions

Q: I have been asked to migrate an application from ASP to ASP.NET. In the ASP application, several third-party utilities have been used to provide for caching. Should I use these or use ASP.NET's internal caching?

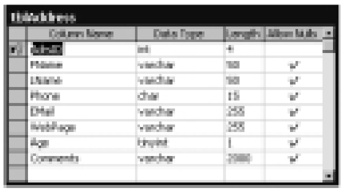
A: Use ASP.NET's caching when possible. With automatic scavenging features and integrated memory management, ASP.NET provides a more tightly integrated caching system than existing third-party utilities.

Summary 295
Solutions Fast Track 296
Frequently Asked Questions 297

Chapter 7 Introduction to ADO.NET: A Simple Address Book 299
Introduction 300
Understanding the Changes in ADO.NET 300
Supported Connectivity 305
The *System.Data* Namespace 305
The *System.Data.Common* Namespace 307
The *System.Data.OleDb* Namespace 307
The *System.Data.SqlClient* Namespace 308
The *System.Data.SqlTypes* Namespace 308
Creating Connection Strings 310
Where to Put the Connection String 312
Creating an Address Book Application 314
Connecting to a Database: Exercise 319
Browsing a Database: Exercise 323
Adding to a Database: Exercise 330
Updating Data in a Database: Exercise 335
Deleting from a Database: Exercise 339
Summary 342
Solutions Fast Track 343
Frequently Asked Questions 345
Frequently Asked Questions 345

Chapter 8 Using XML in the .NET Framework 347
Introduction 348
An Overview of XML 348
What Does an XML Document Look Like? 349
Creating an XML Document 350
Creating an XML Document in VS.NET XML Designer 351
Components of an XML Document 352
Well-Formed XML Documents 355

The tblAddress Layout



tblAddress

Column Name	Data Type	Length	Allow Nulls
[illegible]	int	4	
FName	varchar	50	✓
LName	varchar	50	✓
Phone	char	15	✓
EMail	varchar	255	✓
WebPage	varchar	255	✓
Age	tinyint	1	✓
Comments	varchar	2000	✓

Schema and Valid XML Documents 356
Structure of an XML Document 360
Processing XML Documents Using .NET 361
Reading and Writing XML Documents 362
Storing and Processing XML Documents 363
Reading and Parsing Using the XmlTextReader Class 364
Parsing an XML Document: 365
Navigating through an XML Document to Retrieve Data 367
Writing an XML Document Using the XmlTextWriter Class 370
Generating an XML Document Using *XmlTextWriter* 370
Exploring the XML Document Object Model 373
Navigating through an *XmlDocument* Object 374
Parsing an XML Document Using the *XmlDocument* Object 376
Using the *XmlDataDocument* Class 378
Loading an *XmlDocument* and Retrieving the Values of Certain Nodes 379
Using the Relational View of an *XmlDataDocument* Object 381
Viewing Multiple Tables of a *XmlDataDocument* Object 383
Querying XML Data Using XPathDocument and XPathNavigator 388
Using *XPathDocument* and *XPathNavigator* Objects 390
Using *XPathDocument* and *XPathNavigator* Objects for Document Navigation 392
Transforming an XML Document Using XSLT 396
Transforming an XML Document to an HTML Document 397

Exploring the Components of an XML Document

An XML document contains a variety of constructs. Some of the frequently used ones are as follows:

- **Declaration**
- **Comment**
- **Schema or Document Type Definition (DTD)**
- **Elements**
- **Root Element**
- **Attributes**

Transforming an XML Document into Another XML Document 400
Working with XML and Databases 405
Creating an XML Document from a Database Query 406
Reading an XML Document into a DataSet 408
Summary 410
Solutions Fast Track 410
Frequently Asked Questions 414

Chapter 9 Debugging ASP.NET 417
Introduction 418
Handling Errors 418
Syntax Errors 419
Compilation Errors 419
Runtime Errors 420
Unstructured Error Handling 421
Structured Error Handling 423
Logic Errors 426
Page Tracing 426
Using the *Trace* Class 427
Sorting the *Trace* Information 430
Writing the *Trace* Information to the Application Log 432
Application Tracing 432
Using Visual Studio .NET Debugging Tools 434
Setting Breakpoints 434
Enabling and Disabling Debug Mode 435
Viewing Definitions Using the Object Browser 436
Using the Class Viewer 436
Summary 438
Solutions Fast Track 438
Frequently Asked Questions 439

Properties in the Trace Class

Property	Description
IsEnabled	Indicates whether tracing is enabled for the current request.
TraceMode	Sets the trace mode: sortByCategory or sortByTime.

Chapter 10 Web Services 441
Introduction 442
Understanding Web Services 443
Communication between Servers 448
.asmx Files 450
WSDL 455
Using XML in Web Services 460
An Overview of the System.Web.Services Namespace 461
The *System.Web.Services.Description* Namespace 461
The *System.Web.Services.Discovery* Namespace 461
The *System.Web.Services.Protocols* Namespace 462
Type Marshalling 464
Using DataSets 466
Summary 469
Solutions Fast Track 469
Frequently Asked Questions 471

Chapter 11 Creating an XML.NET Guestbook 473
Introduction 474
Functional Design Requirements of the XML Guestbook 475
Constructing the XML 476
Adding Records to the Guestbook 478
Understanding the *pnlAdd* Panel 482
Adding a Thank-You Panel with *PnlThank* 484
Exploring the Submit Button Handler Code 484
Viewing the Guestbook 488
Displaying Messages 488
Advanced Options for the Guestbook Interface 490
Manipulating Colors and Images 491
Modifying the Page Output 495

Understanding Web Services

Web Services are objects and methods that can be invoked from any client over HTTP. Web Services are built on the Simple Object Access Protocol (SOAP) which enables messaging over HTTP on port 80 (for most Web servers) and uses a standard means of describing data.

Summary 498
Solutions Fast Track 498
Frequently Asked Questions 500

Chapter 12 Creating an ADO.NET Shopping Cart 501

Introduction 502
Setting Up the Database 502
Setting Up the Table "Books" 505
Setting Up the Table "Categories" 505
Setting Up the Table "Customer" 505
Setting Up the Table "Orders" 505
Setting Up the Table "BookOrders" 506
Creating an Access Database 506
SQL Server Database 510
Creating the Stored Procedures 512
Creating the Web Services 518
Overview of the Book Shop Web Services 518
Creating the Data Connection 520
Creating a Web Service 521
Testing a Web Service 527
Using WSDL Web References 531
Building the Site 533
Site Administration 533
Creating the Administration Login (adminLogin.aspx) 535
Creating the Administrator Page (adminPage.aspx) 537
Retrieving the Data: Creating the *getBooks.AllBooks* Web Method 537
Displaying the Data: Binding a *DataGrid* to the *DataSet* 540
Adding New Books to the Database: Creating the *allBooks.addItem* Web Method 541
Deleting Books: Deleting from the *DataGrid* and the Database 541

Using WSDL Web References

- Disco, or vsdisco, written in WSDL, enables access to all Web Services and methods for that site. This provides a one-stop shop, if you will, into the server's cupboards.
- Proxy classes can easily be generated using WSDL, which enables code to access remote services as if they were local classes.

Updating Book Details: Updating the *DataGrid* and the Database 542
Creating the *addBook* Page (addBook.aspx) 543
Customer Administration 543
Creating the Customer Admin Section 543
Creating the *loginCustomer* Page 544
Creating the *updateCustomerInfo* Page 545
Creating an ADOCatalog 547
Creating the BookCatalog Class 548
Creating the *CreateSummaryTable* Method 549
Creating the *InitCatalog* Method 550
Creating the *Catalog* Method 550
Creating the *catalogItemDetails*, *catalogRange*, and *catalogByCategory* Methods 550
Creating the *catalogRangeByCategory* Method 551
Building an XMLCart 553
Creating the User Interface 556
Creating the start.aspx Page 556
Rendering the Catalog 558
Rendering the Cart 559
Creating the Code 559
Summary 562
Solutions Fast Track 562
Frequently Asked Questions 566

Chapter 13 Creating a Message Board with ADO and XML 567

Introduction 568
Setting Up the Database 568
MSAccess Database 569
SQL Server Database 572
Designing Your Application 576
Designing Your Objects 579
Creating Your Data Access Object 579

Designing the *User* Class 581
Designing the *Board* Class 591
Designing the *ThreadList* Class 599
Designing the *Thread* class 603
Designing the *PostList* Class 606
Designing the *Post* Class 608
Designing the *MessageBoard* Class 611
Designing the User Interface 612
Setting Up General Functions 614
Building the Log-In Interface 621
Designing the Browsing Interface 628
Board Browsing 628
Thread Browsing 631
Message Browsing 635
Creating the User Functions 638
Editing the Member Profile 638
Creating Threads and Posts 641
Building the Administrative Interface 645
Summary 658
Solutions Fast Track 658
Frequently Asked Questions 661
Index 663

Setting Up the Database

Setting up the database is one of the most important parts of any application. How do you represent your ideas in a structured, well-formed way? The first and most important step is to break down what you know you want your application to do, analyze those tasks, and then extract the important parts.