Pro C# 5.0 and the .NET 4.5 Framework

Sixth Edition

Andrew Troelsen

Apress®

Pro C# and the .NET 4.5 Framework, Sixth Edition

ISBN 978-1-4302-4233-8

ISBN 978-1-4302-4234-5 (eBook)

President and Publisher: Paul Manning
Lead Editor: Ewan Buckingham
Technical Reviewer: Andy Olsen
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel, Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham, Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh
Coordinating Editors: Jessica Belanger, Christine Ricketts
Copy Editors: Ralph and Vanessa Moore
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global
Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

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

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com. For detailed information about how to locate your book's source code, go to www.apress.com/source-code.

This edition of the text is dedicated to the two most important people in my life. First to my wife Mandy, who was crazy enough to bring up the idea of adoption.

Second, to my son Soren Wade Troelsen. Words can't express how much I love you. However, I will say one thing:

Grrrrrrawwwwhhhh!

You can ask me about this when you get older.

Contents at a Glance

- About the Author lliii
- About the Technical Reviewer liv
- Acknowledgments lv
- Introduction lvi
- Part I: Introducing C# and .NET Platform 1
- Chapter 1: The Philosophy of .NET 3
- Chapter 2: Building C# Applications 39
- Part II: Core C# Programming 71
- Chapter 3: Core C# Programming Constructs, Part I 73
- Chapter 4: Core C# Programming Constructs, Part II 121
- Part III: Object-Oriented Programming with C# 161
- Chapter 5: Understanding Encapsulation 163
- Chapter 6: Understanding Inheritance and Polymorphism 213
- Chapter 7: Understanding Structured Exception Handling 253
- Chapter 8: Working with Interfaces 281
- Part IV: Advanced C# Programming 319
- Chapter 9: Collections and Generics 321
- Chapter 10: Delegates, Events, and Lambda Expressions 359
- Chapter 11: Advanced C# Language Features 399
- Chapter 12: LINQ to Objects 439
- Chapter 13: Understanding Object Lifetime 473
- Part V: Programming with .NET Assemblies 501
- Chapter 14: Building and Configuring Class Libraries 503

Chapter 15: Type Reflection, Late Binding, and Attribute-Based Programming.555
Chapter 16: Dynamic Types and the Dynamic Language Runtime 599
Chapter 17: Processes, AppDomains, and Object Contexts 623
Chapter 18: Understanding CIL and the Role of Dynamic Assemblies 651
Part VI: Introducing the .NET Base Class Libraries 695
Chapter 19: Multithreaded, Parallel, and Async Programming 697
Chapter 20: File I/O and Object Serialization 753
Chapter 21: ADO.NET Part I: The Connected Layer 801
Chapter 22: ADO.NET Part II: The Disconnected Layer 859
Chapter 23: ADO.NET Part III: The Entity Framework 927
Chapter 24: Introducing LINQ to XML 967
Chapter 25: Introducing Windows Communication Foundation 985
Chapter 26: Introducing Windows Workflow Foundation 1047
Part VII: Windows Presentation Foundation 1089
Chapter 27: Introducing Windows Presentation Foundation and XAML 1091
Chapter 28: Programming with WPF Controls 1157
Chapter 29: WPF Graphics Rendering Services 1223
Chapter 30: WPF Resources, Animations, and Styles 1267
Chapter 31: Dependency Properties, Routed Events, and Templates 1301
Part VIII: ASP.NET Web Forms 1335
Chapter 32: Introducing ASP.NET Web Forms 1337
Chapter 33: ASP.NET Web Controls, Master Pages, and Themes 1383
Chapter 34: ASP.NET State Management Techniques 1429
Index 1463

Contents

- **About the Author** lliii
- **About the Technical Reviewer** liv
- **Acknowledgments** lv
- **Introduction** lvi
- **Part I: Introducing C# and .NET Platform** 1
- **Chapter 1: The Philosophy of .NET** 3

An Initial Look at the .NET Platform 3

Some Key Benefits of the .NET Platform 4

Introducing the Building Blocks of the .NET Platform (the CLR, CTS, and CLS) 4

The Role of the Base Class Libraries 5

What C# Brings to the Table 5

Managed vs. Unmanaged Code 7

Additional .NET-Aware Programming Languages 7

Life in a Multilanguage World 8

An Overview of .NET Assemblies 9

The Role of the Common Intermediate Language 10

The Role of .NET Type Metadata 13

The Role of the Assembly Manifest 14

Understanding the Common Type System 15

CTS Class Types 15

CTS Interface Types 16

CTS Structure Types 16
CTS Enumeration Types 17
CTS Delegate Types 17
CTS Type Members 17
Intrinsic CTS Data Types 18
Understanding the Common Language Specification 19
Ensuring CLS Compliance 20
Understanding the Common Language Runtime 21
The Assembly/Namespace/Type Distinction 22
The Role of the Microsoft Root Namespace 25
Accessing a Namespace Programmatically 26
Referencing External Assemblies 27
Exploring an Assembly Using ildasm.exe 28
Viewing CIL Code 29
Viewing Type Metadata 30
Viewing Assembly Metadata (a.k.a. the Manifest) 31
The Platform-Independent Nature of .NET 31
A Brief Word Regarding Windows 8 Applications 33
Building Windows 8 Applications 34
The Role of .NET Under Windows 8 35
Summary 37
Chapter 2: Building C# Applications 39
The Role of the .NET Framework 4.5 SDK 39
The Developer Command Prompt 40
Building C# Applications Using csc.exe 40
Specifying Input and Output Targets 41
Referencing External Assemblies 43

Referencing Multiple External Assemblies 44
Compiling Multiple Source Files 44
Working with C# Response Files 45
Building .NET Applications Using Notepad++ 47
Building .NET Applications Using SharpDevelop 48
Building a Simple Test Project 48
Building .NET Applications Using Visual C# Express 51
Some Unique Features of Visual C# Express 52
Building .NET Applications Using Visual Studio 52
Some Unique Features of Visual Studio 53
Targeting the .NET Framework Using the New Project Dialog Box 54
Using the Solution Explorer Utility 54
The Class View Utility 57
The Object Browser Utility 58
Integrated Support for Code Refactoring 59
Code Snippets and Surround with Technology 62
The Visual Class Designer 63
The Integrated .NET Framework 4.5 SDK Documentation System 67
Summary 70
Part II: Core C# Programming 71
Chapter 3: Core C# Programming Constructs, Part I 73
The Anatomy of a Simple C# Program 73
Variations on the Main() Method 75
Specifying an Application Error Code 76
Processing Command-Line Arguments 77
Specifying Command-Line Arguments with Visual Studio 78
An Interesting Aside: Some Additional Members of the System.Environment Class 79

The System.Console Class 81
Basic Input and Output with the Console Class 82
Formatting Console Output 83
Formatting Numerical Data 84
Formatting Numerical Data Beyond Console Applications 85
System Data Types and Corresponding C# Keywords 86
Variable Declaration and Initialization 87
Intrinsic Data Types and the new Operator 89
The Data Type Class Hierarchy 89
Members of Numerical Data Types 91
Members of System.Boolean 92
Members of System.Char 92
Parsing Values from String Data 92
System.DateTime and System.TimeSpan 93
The System.Numerics.dll Assembly 93
Working with String Data 95
Basic String Manipulation 96
String Concatenation 97
Escape Characters 97
Defining Verbatim Strings 98
Strings and Equality 99
Strings Are Immutable 100
The System.Text.StringBuilder Type 101
Narrowing and Widening Data Type Conversions 102
The checked Keyword 105
Setting Project-Wide Overflow Checking 106
The unchecked Keyword 107
Understanding Implicitly Typed Local Variables 108

Restrictions on Implicitly Typed Variables 109
Implicit Typed Data Is Strongly Typed Data 110
Usefulness of Implicitly Typed Local Variables 111
C# Iteration Constructs 112
The for Loop 112
The foreach Loop 113
The while and do/while Looping Constructs 114
Decision Constructs and the Relational/Equality Operators 114
The if/else Statement 115
Equality and Relational Operators 115
Conditional Operators 116
The switch Statement 116
Summary 118
Chapter 4: Core C# Programming Constructs, Part II 121
Methods and Parameter Modifiers 121
The Default by Value Parameter-Passing Behavior 122
The out Modifier 123
The ref Modifier 125
The params Modifier 126
Defining Optional Parameters 127
Invoking Methods Using Named Parameters 129
Understanding Method Overloading 130
Understanding C# Arrays 133
C# Array Initialization Syntax 134
Implicitly Typed Local Arrays 134
Defining an Array of Objects 135
Working with Multidimensional Arrays 136
Arrays As Arguments or Return Values 137

The System.Array Base Class 138
Understanding the enum Type 140
Controlling the Underlying Storage for an enum 141
Declaring enum Variables 142
The System.Enum Type 143
Dynamically Discovering an enum's Name/Value Pairs 143
Understanding the Structure Type 146
Creating Structure Variables 147
Understanding Value Types and Reference Types 149
Value Types, References Types, and the Assignment Operator 150
Value Types Containing Reference Types 151
Passing Reference Types by Value 153
Passing Reference Types by Reference 155
Final Details Regarding Value Types and Reference Types 156
Understanding C# Nullable Types 157
Working with Nullable Types 158
The ?? Operator 159
Summary 160
Part III: Object-Oriented Programming with C# 161
Chapter 5: Understanding Encapsulation 163
Introducing the C# Class Type 163
Allocating Objects with the new Keyword 166
Understanding Constructors 166
The Role of the Default Constructor 167
Defining Custom Constructors 167
The Default Constructor Revisited 169
The Role of the this Keyword 170

Chaining Constructor Calls Using this 172
Observing Constructor Flow 174
Revisiting Optional Arguments 176
Understanding the static Keyword 177
Defining Static Field Data 178
Defining Static Methods 179
Defining Static Constructors 181
Defining Static Classes 183
Defining the Pillars of OOP 184
The Role of Encapsulation 185
The Role of Inheritance 185
The Role of Polymorphism 187
C# Access Modifiers 188
The Default Access Modifiers 189
Access Modifiers and Nested Types 189
The First Pillar: C#'s Encapsulation Services 190
Encapsulation Using Traditional Accessors and Mutators 191
Encapsulation Using .NET Properties 193
Using Properties Within a Class Definition 196
Read-Only and Write-Only Properties 198
Revisiting the static Keyword: Defining Static Properties 199
Understanding Automatic Properties 199
Interacting with Automatic Properties 201
Regarding Automatic Properties and Default Values 201
Understanding Object Initialization Syntax 203
Calling Custom Constructors with Initialization Syntax 204
Initializing Inner Types 206
Working with Constant Field Data 207

Understanding Read-Only Fields 208
Static Read-Only Fields 209
Understanding Partial Types 210
Summary 211
Chapter 6: Understanding Inheritance and Polymorphism 213
The Basic Mechanics of Inheritance 213
Specifying the Parent Class of an Existing Class 214
Regarding Multiple Base Classes 216
The sealed Keyword 216
Revising Visual Studio Class Diagrams 218
The Second Pillar of OOP: The Details of Inheritance 220
Controlling Base Class Creation with the base Keyword 221
Keeping Family Secrets: The protected Keyword 223
Adding a Sealed Class 224
Programming for Containment/Delegation 225
Understanding Nested Type Definitions 226
The Third Pillar of OOP: C#'s Polymorphic Support 228
The virtual and override Keywords 229
Overriding Virtual Members Using the Visual Studio IDE 231
Sealing Virtual Members 232
Understanding Abstract Classes 233
Understanding the Polymorphic Interface 235
Understanding Member Shadowing 239
Understanding Base Class/Derived Class Casting Rules 240
The C# as Keyword 242
The C# is Keyword 243
The Master Parent Class: System.Object 243

Overriding System.Object.ToString() 247
Overriding System.Object.Equals() 247
Overriding System.Object.GetHashCode() 248
Testing Your Modified Person Class 249
The Static Members of System.Object 250
Summary 250
Chapter 7: Understanding Structured Exception Handling 253
Ode to Errors, Bugs, and Exceptions 253
The Role of .NET Exception Handling 254
The Building Blocks of .NET Exception Handling 255
The System.Exception Base Class 255
The Simplest Possible Example 257
Throwing a General Exception 259
Catching Exceptions 260
Configuring the State of an Exception 261
The TargetSite Property 261
The StackTrace Property 262
The HelpLink Property 263
The Data Property 264
System-Level Exceptions (System.SystemException) 266
Application-Level Exceptions (System.ApplicationException) 266
Building Custom Exceptions, Take One 267
Building Custom Exceptions, Take Two 269
Building Custom Exceptions, Take Three 269
Processing Multiple Exceptions 271
General catch Statements 274
Rethrowing Exceptions 274

Inner Exceptions 275
The finally Block 276
Who Is Throwing What? 277
The Result of Unhandled Exceptions 277
Debugging Unhandled Exceptions Using Visual Studio 278
Summary 280
Chapter 8: Working with Interfaces 281
Understanding Interface Types 281
Interface Types vs. Abstract Base Classes 282
Defining Custom Interfaces 284
Implementing an Interface 286
Invoking Interface Members at the Object Level 288
Obtaining Interface References: The as Keyword 289
Obtaining Interface References: The is Keyword 290
Interfaces As Parameters 291
Interfaces As Return Values 293
Arrays of Interface Types 294
Implementing Interfaces Using Visual Studio 295
Explicit Interface Implementation 296
Designing Interface Hierarchies 299
Multiple Inheritance with Interface Types 300
The IEnumerable and IEnumerator Interfaces 302
Building Iterator Methods with the yield Keyword 305
Building a Named Iterator 306
The ICloneable Interface 308
A More Elaborate Cloning Example 310

The IComparable Interface 313
Specifying Multiple Sort Orders with IComparer 316
Custom Properties and Custom Sort Types 317
Summary 318
Part IV: Advanced C# Programming 319
Chapter 9: Collections and Generics 321
The Motivation for Collection Classes 321
The System.Collections Namespace 323
A Survey of System.Collections.Specialized Namespace 325
The Problems of Nongeneric Collections 326
The Issue of Performance 326
The Issue of Type Safety 330
A First Look at Generic Collections 333
The Role of Generic Type Parameters 334
Specifying Type Parameters for Generic Classes/Structures 335
Specifying Type Parameters for Generic Members 336
Specifying Type Parameters for Generic Interfaces 337
The System.Collections.Generic Namespace 338
Understanding Collection Initialization Syntax 340
Working with the List<T> Class 341
Working with the Stack<T> Class 342
Working with the Queue<T> Class 343
Working with the SortedSet<T> Class 345
The System.Collections.ObjectModel Namespace 346
Working with ObservableCollection<T> 347
Creating Custom Generic Methods 349
Inference of Type Parameters 351

Creating Custom Generic Structures and Classes 353
The default Keyword in Generic Code 354
Constraining Type Parameters 355
Examples Using the where Keyword 356
The Lack of Operator Constraints 357
Summary 358
Chapter 10: Delegates, Events, and Lambda Expressions 359
Understanding the .NET Delegate Type 359
Defining a Delegate Type in C# 360
The System.MulticastDelegate and System.Delegate Base Classes 362
The Simplest Possible Delegate Example 364
Investigating a Delegate Object 365
Sending Object State Notifications Using Delegates 367
Enabling Multicasting 370
Removing Targets from a Delegate's Invocation List 371
Method Group Conversion Syntax 372
Understanding Generic Delegates 374
The Generic Action<> and Func<> Delegates 375
Understanding C# Events 378
The C# event Keyword 379
Events Under the Hood 380
Listening to Incoming Events 381
Simplifying Event Registration Using Visual Studio 383
Creating Custom Event Arguments 384
The Generic EventHandler<T> Delegate 386
Understanding C# Anonymous Methods 387
Accessing Local Variables 389

Understanding Lambda Expressions 390
Dissecting a Lambda Expression 393
Processing Arguments Within Multiple Statements 394
Lambda Expressions with Multiple (or Zero) Parameters 395
Retrofitting the CarEvents Example Using Lambda Expressions 396
Summary 397
Chapter 11: Advanced C# Language Features 399
Understanding Indexer Methods 399
Indexing Data Using String Values 401
Overloading Indexer Methods 403
Indexers with Multiple Dimensions 403
Indexer Definitions on Interface Types 404
Understanding Operator Overloading 404
Overloading Binary Operators 405
And What of the += and –+ Operators? 408
Overloading Unary Operators 408
Overloading Equality Operators 409
Overloading Comparison Operators 410
Final Thoughts Regarding Operator Overloading 411
Understanding Custom Type Conversions 412
Recall: Numerical Conversions 412
Recall: Conversions Among Related Class Types 412
Creating Custom Conversion Routines 413
Additional Explicit Conversions for the Square Type 416
Defining Implicit Conversion Routines 417
Understanding Extension Methods 418
Defining Extension Methods 418
Invoking Extension Methods 420

Importing Extension Methods 421
The IntelliSense of Extension Methods 421
Extending Types Implementing Specific Interfaces 422
Understanding Anonymous Types 423
Defining an Anonymous Type 424
The Internal Representation of Anonymous Types 425
The Implementation of ToString() and GetHashCode() 426
The Semantics of Equality for Anonymous Types 427
Anonymous Types Containing Anonymous Types 429
Working with Pointer Types 429
The unsafe Keyword 431
Working with the * and & Operators 433
An Unsafe (and Safe) Swap Function 434
Field Access via Pointers (the -> Operator) 435
The stackalloc Keyword 435
Pinning a Type via the fixed Keyword 436
The sizeof Keyword 437
Summary 438
Chapter 12: LINQ to Objects 439
LINQ-Specific Programming Constructs 439
Implicit Typing of Local Variables 440
Object and Collection Initialization Syntax 441
Lambda Expressions 441
Extension Methods 442
Anonymous Types 443
Understanding the Role of LINQ 443
LINQ Expressions Are Strongly Typed 444
The Core LINQ Assemblies 444

Applying LINQ Queries to Primitive Arrays 445
Once Again, Without LINQ 447
Reflecting over a LINQ Result Set 447
LINQ and Implicitly Typed Local Variables 448
LINQ and Extension Methods 449
The Role of Deferred Execution 450
The Role of Immediate Execution 452
Returning the Result of a LINQ Query 452
Returning LINQ Results via Immediate Execution 454
Applying LINQ Queries to Collection Objects 455
Accessing Contained Subobjects 455
Applying LINQ Queries to Nongeneric Collections 456
Filtering Data Using OfType<T>() 457
Investigating the C# LINQ Query Operators 457
Basic Selection Syntax 459
Obtaining Subsets of Data 460
Projecting New Data Types 461
Obtaining Counts Using Enumerable 462
Reversing Result Sets 462
Sorting Expressions 463
LINQ As a Better Venn Diagramming Tool 463
Removing Duplicates 464
LINQ Aggregation Operations 465
The Internal Representation of LINQ Query Statements 466
Building Query Expressions with Query Operators (Revisited) 466
Building Query Expressions Using the Enumerable Type and Lambda Expressions 467
Building Query Expressions Using the Enumerable Type and Anonymous Methods 468
Building Query Expressions Using the Enumerable Type and Raw Delegates 469

Summary 470

Chapter 13: Understanding Object Lifetime 473

Classes, Objects, and References 473

The Basics of Object Lifetime 475

The CIL of new 475

Setting Object References to null 477

The Role of Application Roots 477

Understanding Object Generations 479

Concurrent Garbage Collection Under .NET 1.0–3.5 480

Background Garbage Collection Under .NET 4.0 and Greater 481

The System.GC Type 481

Forcing a Garbage Collection 482

Building Finalizable Objects 485

Overriding System.Object.Finalize() 486

Detailing the Finalization Process 488

Building Disposable Objects 489

Reusing the C# using Keyword 491

Building Finalizable and Disposable Types 493

A Formalized Disposal Pattern 493

Understanding Lazy Object Instantiation 496

Customizing the Creation of the Lazy Data 498

Summary 499

Part V: Programming with .NET Assemblies 501

Chapter 14: Building and Configuring Class Libraries 503

Defining Custom Namespaces 503

Resolving Name Clashes with Fully Qualified Names 505

Resolving Name Clashes with Aliases 506
Creating Nested Namespaces 508
The Default Namespace of Visual Studio 509
The Role of .NET Assemblies 510
Assemblies Promote Code Reuse 510
Assemblies Establish a Type Boundary 510
Assemblies Are Versionable Units 510
Assemblies Are Self-Describing 511
Assemblies Are Configurable 511
Understanding the Format of a .NET Assembly 511
The Windows File Header 512
The CLR File Header 513
CIL Code, Type Metadata, and the Assembly Manifest 514
Optional Assembly Resources 514
Building and Consuming Custom Class Library 514
Exploring the Manifest 518
Exploring the CIL 520
Exploring the Type Metadata 521
Building a C# Client Application 522
Building a Visual Basic Client Application 524
Cross-Language Inheritance in Action 525
Understanding Private Assemblies 526
The Identity of a Private Assembly 526
Understanding the Probing Process 526
Configuring Private Assemblies 527
The Role of the App.Config File 529
Understanding Shared Assemblies 531
The Global Assembly Cache 532

Understanding Strong Names 534
Generating Strong Names at the Command Line 535
Generating Strong Names Using Visual Studio 537
Installing Strongly Named Assemblies to the GAC 539
Consuming a Shared Assembly 541
Exploring the Manifest of SharedCarLibClient 543
Configuring Shared Assemblies 543
Freezing the Current Shared Assembly 544
Building a Shared Assembly Version 2.0.0.0 544
Dynamically Redirecting to Specific Versions of a Shared Assembly 547
Understanding Publisher Policy Assemblies 548
Disabling Publisher Policy 549
Understanding the <codeBase> Element 550
The System.Configuration Namespace 552
The Configuration File Schema Documentation 553
Summary 554
Chapter 15: Type Reflection, Late Binding, and Attribute-Based Programming 555
The Necessity of Type Metadata 555
Viewing (Partial) Metadata for the EngineState Enumeration 556
Viewing (Partial) Metadata for the Car Type 557
Examining a TypeRef 559
Documenting the Defining Assembly 559
Documenting Referenced Assemblies 559
Documenting String Literals 560
Understanding Reflection 560
The System.Type Class 561
Obtaining a Type Reference Using System.Object.GetType() 562

Obtaining a Type Reference Using typeof() 563
Obtaining a Type Reference Using System.Type.GetType() 563
Building a Custom Metadata Viewer 564
Reflecting on Methods 564
Reflecting on Fields and Properties 565
Reflecting on Implemented Interfaces 565
Displaying Various Odds and Ends 566
Implementing Main() 566
Reflecting on Generic Types 568
Reflecting on Method Parameters and Return Values 568
Dynamically Loading Assemblies 569
Reflecting on Shared Assemblies 572
Understanding Late Binding 574
The System.Activator Class 574
Invoking Methods with No Parameters 576
Invoking Methods with Parameters 577
Understanding the Role of .NET Attributes 578
Attribute Consumers 579
Applying Attributes in C# 579
C# Attribute Shorthand Notation 581
Specifying Constructor Parameters for Attributes 581
The Obsolete Attribute in Action 582
Building Custom Attributes 583
Applying Custom Attributes 583
Named Property Syntax 584
Restricting Attribute Usage 584
Assembly-Level Attributes 585
The Visual Studio AssemblyInfo.cs File 586

Reflecting on Attributes Using Early Binding 588
Reflecting on Attributes Using Late Binding 589
Putting Reflection, Late Binding, and Custom Attributes in Perspective 591
Building an Extendable Application 591
Building CommonSnappableTypes.dll 592
Building the C# Snap-In 593
Building the Visual Basic Snap-In 593
Building an Extendable Windows Forms Application 594
Summary 598
Chapter 16: Dynamic Types and the Dynamic Language Runtime 599
The Role of the C# dynamic Keyword 599
Calling Members on Dynamically Declared Data 601
The Role of the Microsoft.CSharp.dll Assembly 603
The Scope of the dynamic Keyword 604
Limitations of the dynamic Keyword 605
Practical Uses of the dynamic Keyword 605
The Role of the Dynamic Language Runtime (DLR) 606
The Role of Expression Trees 606
The Role of the System.Dynamic Namespace 607
Dynamic Runtime Lookup of Expression Trees 607
Simplifying Late-Bound Calls Using Dynamic Types 608
Leveraging the dynamic Keyword to Pass Arguments 609
Simplifying COM Interoperability Using Dynamic Data 612
The Role of Primary Interop Assemblies (PIAs) 613
Embedding Interop Metadata 614
Common COM Interop Pain Points 615
COM Interop Using C# Dynamic Data 616

COM interop Without C# Dynamic Data 620
Summary 621
Chapter 17: Processes, AppDomains, and Object Contexts 623
The Role of a Windows Process 623
The Role of Threads 624
Interacting with Processes Under the .NET Platform 625
Enumerating Running Processes 628
Investigating a Specific Process 629
Investigating a Process's Thread Set 629
Investigating a Process's Module Set 632
Starting and Stopping Processes Programmatically 633
Controlling Process Startup Using the ProcessStartInfo Class 634
Understanding .NET Application Domains 635
The System.AppDomain Class 636
Interacting with the Default Application Domain 638
Enumerating Loaded Assemblies 639
Receiving Assembly Load Notifications 640
Creating New Application Domains 641
Loading Assemblies into Custom Application Domains 643
Programmatically Unloading AppDomains 644
Understanding Object Context Boundaries 646
Context-Agile and Context-Bound Types 646
Defining a Context-Bound Object 647
Inspecting an Object's Context 647
Summarizing Processes, AppDomains, and Context 649
Summary 649

Chapter 18: Understanding CIL and the Role of Dynamic Assemblies 651

Reasons for Learning the Grammar of CIL 651

Examining CIL Directives, Attributes, and Opcodes 652

The Role of CIL Directives 653

The Role of CIL Attributes 653

The Role of CIL Opcodes 653

The CIL Opcode/CIL Mnemonic Distinction 653

Pushing and Popping: The Stack-Based Nature of CIL 654

Understanding Round-Trip Engineering 656

The Role of CIL Code Labels 659

Interacting with CIL: Modifying an *.il File 660

Compiling CIL Code Using ilasm.exe 661

The Role of peverify.exe 662

Understanding CIL Directives and Attributes 663

Specifying Externally Referenced Assemblies in CIL 663

Defining the Current Assembly in CIL 663

Defining Namespaces in CIL 664

Defining Class Types in CIL 665

Defining and Implementing Interfaces in CIL 666

Defining Structures in CIL 667

Defining Enums in CIL 667

Defining Generics in CIL 668

Compiling the CILTypes.il file 668

.NET Base Class Library, C#, and CIL Data Type Mappings 669

Defining Type Members in CIL 670

Defining Field Data in CIL 670

Defining Type Constructors in CIL 671

Defining Properties in CIL 671

Defining Member Parameters 672
Examining CIL Opcodes 673
The .maxstack Directive 675
Declaring Local Variables in CIL 676
Mapping Parameters to Local Variables in CIL 677
The Hidden this Reference 677
Representing Iteration Constructs in CIL 678
Building a .NET Assembly with CIL 679
Building CILCars.dll 679
Building CILCarClient.exe 681
Understanding Dynamic Assemblies 683
Exploring the System.Reflection.Emit Namespace 684
The Role of the System.Reflection.Emit.ILGenerator 685
Emitting a Dynamic Assembly 686
Emitting the Assembly and Module Set 688
The Role of the ModuleBuilder Type 689
Emitting the HelloClass Type and the String Member Variable 690
Emitting the Constructors 691
Emitting the SayHello() Method 692
Using the Dynamically Generated Assembly 692
Summary 693
Part VI: Introducing the .NET Base Class Libraries 695
Chapter 19: Multithreaded, Parallel, and Async Programming 697
The Process/AppDomain/Context/Thread Relationship 697
The Problem of Concurrency 698
The Role of Thread Synchronization 699
A Brief Review of the .NET Delegate 699

The Asynchronous Nature of Delegates 701
The BeginInvoke() and EndInvoke() Methods 701
The System.IAsyncResult Interface 702
Invoking a Method Asynchronously 703
Synchronizing the Calling Thread 703
The Role of the AsyncCallback Delegate 705
The Role of the AsyncResult Class 708
Passing and Receiving Custom State Data 708
The System.Threading Namespace 709
The System.Threading.Thread Class 711
Obtaining Statistics About the Current Thread of Execution 712
The Name Property 713
The Priority Property 713
Manually Creating Secondary Threads 714
Working with the ThreadStart Delegate 715
Working with the ParameterizedThreadStart Delegate 717
The AutoResetEvent Class 718
Foreground Threads and Background Threads 719
The Issue of Concurrency 720
Synchronization Using the C# lock Keyword 723
Synchronization Using the System.Threading.Monitor Type 725
Synchronization Using the System.Threading.Interlocked Type 726
Synchronization Using the [Synchronization] Attribute 727
Programming with Timer Callbacks 727
Understanding the CLR ThreadPool 729
Parallel Programming Using the Task Parallel Library 731
The System.Threading.Tasks Namespace 731

The Role of the Parallel Class 732
Data Parallelism with the Parallel Class 733
Accessing UI Elements on Secondary Threads 735
The Task Class 736
Handling Cancellation Request 736
Task Parallelism Using the Parallel Class 738
Parallel LINQ Queries (PLINQ) 741
Opting in to a PLINQ Query 743
Cancelling a PLINQ Query 743
Asynchronous Calls Under .NET 4.5 744
A First Look at the C# async and await Keywords 744
Naming Conventions for Async Methods 746
Async Methods Returning Void 748
Async Methods with Multiple Awaits 748
Retrofitting the AddWithThreads Example Using Asycn/Await 749
Summary 751
Chapter 20: File I/O and Object Serialization 753
Exploring the System.IO Namespace 753
The Directory(Info) and File(Info) Types 754
The Abstract FileSystemInfo Base Class 755
Working with the DirectoryInfo Type 756
Enumerating Files with the DirectoryInfo Type 758
Creating Subdirectories with the DirectoryInfo Type 758
Working with the Directory Type 760
Working with the DriveInfo Class Type 760
Working with the FileInfo Class 762
The FileInfo.Create() Method 763

The FileInfo.Open() Method 764
The FileInfo.OpenRead() and FileInfo.OpenWrite() Methods 765
The FileInfo.OpenText() Method 766
The FileInfo.CreateText() and FileInfo.AppendText() Methods 766
Working with the File Type 766
Additional File-Centric Members 767
The Abstract Stream Class 768
Working with FileStreams 770
Working with StreamWriters and StreamReaders 771
Writing to a Text File 772
Reading from a Text File 773
Directly Creating StreamWriter/StreamReader Types 774
Working with StringWriters and StringReaders 774
Working with BinaryWriters and BinaryReaders 776
Watching Files Programmatically 778
Understanding Object Serialization 780
The Role of Object Graphs 781
Configuring Objects for Serialization 783
Defining Serializable Types 783
Public Fields, Private Fields, and Public Properties 784
Choosing a Serialization Formatter 784
The IFormatter and IRemotingFormatter Interfaces 785
Type Fidelity Among the Formatters 786
Serializing Objects Using the BinaryFormatter 787
Deserializing Objects Using the BinaryFormatter 788
Serializing Objects Using the SoapFormatter 789
Serializing Objects Using the XmlSerializer 790

Controlling the Generated XML Data 791
Serializing Collections of Objects 792
Customizing the Soap/Binary Serialization Process 794
A Deeper Look at Object Serialization 795
Customizing Serialization Using ISerializable 796
Customizing Serialization Using Attributes 799
Summary 800
Chapter 21: ADO.NET Part I: The Connected Layer 801
A High-Level Definition of ADO.NET 801
The Three Faces of ADO.NET 802
Understanding ADO.NET Data Providers 803
The Microsoft-Supplied ADO.NET Data Providers 804
A Word Regarding System.Data.OracleClient.dll 806
Obtaining Third-Party ADO.NET Data Providers 806
Additional ADO.NET Namespaces 806
The Types of the System.Data Namespace 807
The Role of the IDbConnection Interface 809
The Role of the IDbTransaction Interface 809
The Role of the IDbCommand Interface 809
The Role of the IDbDataParameter and IDataParameter Interfaces 810
The Role of the IDbDataAdapter and IDataAdapter Interfaces 811
The Role of the IDataReader and IDataRecord Interfaces 811
Abstracting Data Providers Using Interfaces 812
Increasing Flexibility Using Application Configuration Files 814
Creating the AutoLot Database 815
Creating the Inventory Table 816
Adding Test Records to the Inventory Table 818

Authoring the GetPetName() Stored Procedure 819
Creating the Customers and Orders Tables 820
Visually Creating Table Relationships 823
The ADO.NET Data Provider Factory Model 824
A Complete Data Provider Factory Example 825
A Potential Drawback with the Data Provider Factory Model 828
The <connectionStrings> Element 829
Understanding the Connected Layer of ADO.NET 830
Working with Connection Objects 831
Working with ConnectionStringBuilder Objects 833
Working with Command Objects 834
Working with Data Readers 835
Obtaining Multiple Result Sets Using a Data Reader 837
Building a Reusable Data Access Library 838
Adding the Connection Logic 839
Adding the Insertion Logic 840
Adding the Deletion Logic 841
Adding the Update Logic 841
Adding the Selection Logic 842
Working with Parameterized Command Objects 843
Executing a Stored Procedure 845
Creating a Console UI–Based Front End 847
Implementing the Main() Method 847
Implementing the ShowInstructions() Method 849
Implementing the ListInventory() Method 849
Implementing the DeleteCar() Method 850
Implementing the InsertNewCar() Method 851
Implementing the UpdateCarPetName() Method 851

Implementing LookUpPetName() 852
Understanding Database Transactions 853
Key Members of an ADO.NET Transaction Object 854
Adding a CreditRisks Table to the AutoLot Database 855
Adding a Transaction Method to InventoryDAL 856
Testing Your Database Transaction 857
Summary 858
Chapter 22: ADO.NET Part II: The Disconnected Layer 859
Understanding the Disconnected Layer of ADO.NET 859
Understanding the Role of the DataSet 861
Key Properties of the DataSet 861
Key Methods of the DataSet 862
Building a DataSet 863
Working with DataColumns 864
Building a DataColumn 865
Enabling Autoincrementing Fields 866
Adding DataColumn Objects to a DataTable 866
Working with DataRows 867
Understanding the RowState Property 868
Understanding the DataRowVersion Property 870
Working with DataTables 871
Inserting DataTables into DataSets 872
Obtaining Data in a DataSet 872
Processing DataTable Data Using DataTableReader Objects 873
Serializing DataTable/DataSet Objects As XML 874
Serializing DataTable/DataSet Objects in a Binary Format 876
Binding DataTable Objects to Windows Forms GUIs 877

Hydrating a DataTable from a Generic List<T> 878
Deleting Rows from a DataTable 881
Selecting Rows Based on Filter Criteria 883
Updating Rows Within a DataTable 886
Working with the DataView Type 886
Working with Data Adapters 888
A Simple Data Adapter Example 889
Mapping Database Names to Friendly Names 890
Adding Disconnection Functionality to AutoLotDAL.dll 892
Defining the Initial Class Type 892
Configuring the Data Adapter Using the SqlCommandBuilder 892
Implementing GetAllInventory() 894
Implementing UpdateInventory() 894
Setting Your Version Number 894
Testing the Disconnected Functionality 894
Multitabled DataSet Objects and Data Relationships 896
Prepping the Data Adapters 897
Building the Table Relationships 899
Updating the Database Tables 899
Navigating Between Related Tables 900
The Windows Forms Database Designer Tools 902
Visually Designing the DataGridView 902
The Generated App.config File 907
Examining the Strongly Typed DataSet 908
Examining the Strongly Typed DataTable 909
Examining the Strongly Typed DataRow 910
Examining the Strongly Typed Data Adapter 911
Completing the Windows Forms Application 912

Isolating Strongly Typed Database Code into a Class Library....913
Viewing the Generated Code....915
Selecting Data with the Generated Code....917
Inserting Data with the Generated Code....918
Deleting Data with the Generated Code....919
Invoking a Stored Procedure Using the Generated Code....919
Programming with LINQ to DataSet....920
The Role of the DataSet Extensions Library....922
Obtaining a LINQ-Compatible DataTable....923
The Role of the DataRowExtensions.Field<T>() Extension Method....924
Hydrating New DataTables from LINQ Queries....925
Summary....926
Chapter 23: ADO.NET Part III: The Entity Framework....927
Understanding the Role of Entity Framework....927
The Role of Entities....929
The Building Blocks of the Entity Framework....931
Building and Analyzing Your First EDM....937
Programming Against the Conceptual Model....950
AutoLotDAL Version Four, Now with Entities....956
The Role of Navigation Properties....957
Using Navigation Properties within LINQ to Entity Queries....959
Invoking a Stored Procedure....961
Data Binding Entities to Windows Forms GUIs....961
Adding the Data-Binding Code....964
Going Forward with .NET Data-Access APIs....965
Summary....966

Chapter 24: Introducing LINQ to XML 967

A Tale of Two XML APIs 967

LINQ to XML As a Better DOM 969

VB Literal Syntax As a Better LINQ to XML 969

Members of the System.Xml.Linq Namespace 971

The LINQ to XML Axis Methods 973

The Oddness of XName (and XNamespace) 975

Working with XElement and XDocument 975

Generating Documents from Arrays and Containers 978

Loading and Parsing XML Content 979

Manipulating an In-Memory XML Document 980

Building the UI of the LINQ to XML App 980

Import the Inventory.xml File 981

Defining a LINQ to XML Helper Class 981

Attaching the UI to Your Helper Class 983

Summary 984

Chapter 25: Introducing Windows Communication Foundation 985

A Potpourri of Distributed Computing APIs 985

The Role of DCOM 986

The Role of COM+/Enterprise Services 987

The Role of MSMQ 987

The Role of .NET Remoting 988

The Role of XML Web Services 988

The Role of WCF 990

An Overview of WCF Features 990

An Overview of Service-Oriented Architecture 991

WCF: The Bottom Line 992

Investigating the Core WCF Assemblies 992
The Visual Studio WCF Project Templates 993
The WCF Service Web Site Project Template 995
The Basic Composition of a WCF Application 996
The ABCs of WCF 997
Understanding WCF Contracts 997
Understanding WCF Bindings 999
Understanding WCF Addresses 1001
Building a WCF Service 1002
The [ServiceContract] Attribute 1004
The [OperationContract] Attribute 1005
Service Types As Operational Contracts 1005
Hosting the WCF Service 1006
Establishing the ABCs Within an App.config File 1007
Coding Against the ServiceHost Type 1007
Specifying Base Addresses 1008
Details of the ServiceHost Type 1009
Details of the <system.serviceModel> Element 1011
Enabling Metadata Exchange 1012
Building the WCF Client Application 1015
Generating Proxy Code Using svcutil.exe 1015
Generating Proxy Code Using Visual Studio 1016
Configuring a TCP-Based Binding 1018
Simplifying Configuration Settings 1020
Leveraging Default Endpoints 1020
Exposing a Single WCF Service Using Multiple Bindings 1021
Changing Settings for a WCF Binding 1022
Leveraging the Default MEX Behavior Configuration 1024

Refreshing the Client Proxy and Selecting the Binding 1025
Using the WCF Service Library Project Template 1027
Building a Simple Math Service 1027
Testing the WCF Service with WcfTestClient.exe 1027
Altering Configuration Files Using SvcConfigEditor.exe 1028
Hosting the WCF Service Within a Windows Service 1030
Specifying the ABCs in Code 1031
Enabling MEX 1033
Creating a Windows Service Installer 1033
Installing the Windows Service 1035
Invoking a Service Asynchronously from the Client 1036
Designing WCF Data Contracts 1039
Using the Web-Centric WCF Service Project Template 1040
Implementing the Service Contract 1042
The Role of the *.svc File 1043
Examining the Web.config File 1043
Testing the Service 1044
Summary 1044
Chapter 26: Introducing Windows Workflow Foundation 1047
Defining a Business Process 1047
The Role of WF 1048
Building a Simple Workflow 1048
The Workflow Runtime 1051
Hosting a Workflow Using WorkflowInvoker 1052
Hosting a Workflow Using WorkflowApplication 1055
Recap of Your First Workflow 1056
Examining the Workflow Activities 1057

Control Flow Activities 1057
Flowchart Activities 1058
Messaging Activities 1058
The State Machine Activities 1059
The Runtime and Primitives Activities 1059
The Transaction Activities 1060
The Collection and Error Handling Activities 1060
Building a Flowchart Workflow 1061
Connecting Activities in a Flowchart 1062
Working with the InvokeMethod Activity 1063
Defining Workflow-Wide Variables 1064
Working with the FlowDecision Activity 1065
Working with the TerminateWorkflow Activity 1066
Building the “True” Condition 1067
Working with the ForEach<T> Activity 1067
Completing the Application 1070
Reflecting on What We Have Done 1071
Building a Sequence Workflow (in a Dedicated DLL) 1073
Defining the Initial Project 1073
Importing Assemblies and Namespaces 1074
Defining the Workflow Arguments 1075
Defining Workflow Variables 1076
Working with the Assign Activity 1077
Working with the If and Switch Activities 1078
Building a Custom Code Activity 1081
Consuming the Workflow Library 1084
Retrieving the Workflow Output Argument 1085
Summary 1086

Part VII: Windows Presentation Foundation 1089

Chapter 27: Introducing Windows Presentation Foundation and XAML 1091

The Motivation Behind WPF 1091

Unifying Diverse APIs 1092

Providing a Separation of Concerns via XAML 1093

Providing an Optimized Rendering Model 1093

Simplifying Complex UI Programming 1094

The Various Flavors of WPF 1094

Traditional Desktop Applications 1094

Navigation-Based WPF Applications 1096

XBAP Applications 1097

The WPF/Silverlight Relationship 1099

Investigating the WPF Assemblies 1099

The Role of the Application Class 1101

The Role of the Window Class 1103

Building a WPF Application Without XAML 1107

Creating a Strongly Typed Window 1109

Creating a Simple User Interface 1110

Interacting with Application-Level Data 1112

Handling the Closing of a Window Object 1113

Intercepting Mouse Events 1114

Intercepting Keyboard Events 1115

Building a WPF Application Using Only XAML 1117

Defining a Window Object in XAML 1118

Defining the Application Object in XAML 1119

Processing the XAML Files Using msbuild.exe 1120

Transforming Markup into a .NET Assembly 1122

Mapping the Window XAML Markup to C# Code 1122

The Role of BAML 1124
Mapping the Application XAML Markup to C# Code 1125
XAML-to-Assembly Process Summary 1126
Understanding the Syntax of WPF XAML 1127
Introducing Kaxaml 1127
XAML XML Namespaces and XAML "Keywords" 1128
Controlling Class and Member Variable Visibility 1131
XAML Elements, XAML Attributes, and Type Converters 1131
Understanding XAML Property-Element Syntax 1133
Understanding XAML Attached Properties 1133
Understanding XAML Markup Extensions 1134
Building a WPF Application Using Code-Behind Files 1136
Adding a Code File for the MainWindow Class 1136
Adding a Code File for the MyApp Class 1137
Processing the Code Files with msbuild.exe 1138
Building WPF Applications Using Visual Studio 1139
The WPF Project Templates 1139
The Toolbox and XAML Designer/Editor 1141
Setting Properties Using the Properties Window 1143
Handling Events Using the Properties Window 1145
Handling Events in the XAML Editor 1146
The Document Outline Window 1146
Viewing the AutoGenerated Code Files 1147
Building a Custom XAML Editor with Visual Studio 1148
Designing the GUI of Our Window 1148
Implementing the Loaded Event 1150
Implementing the Button's Click Event 1151
Implementing the Closed Event 1153

Testing Your Application 1153
Exploring the WPF Documentation 1154
Summary 1155
Chapter 28: Programming with WPF Controls 1157
A Survey of the Core WPF Controls 1157
The WPF Ink Controls 1158
The WPF Document Controls 1158
WPF Common Dialog Boxes 1159
The Details Are in the Documentation 1159
A Brief Review of the Visual Studio WPF Designer 1160
Working with WPF Controls Using Visual Studio 1161
Working with the Document Outline Editor 1162
Controlling Content Layout Using Panels 1163
Positioning Content Within Canvas Panels 1166
Positioning Content Within WrapPanel Panels 1167
Positioning Content Within StackPanel Panels 1169
Positioning Content Within Grid Panels 1170
Positioning Content Within DockPanel Panels 1173
Enabling Scrolling for Panel Types 1174
Configuring Panels Using the Visual Studio Designers 1175
Building a Window's Frame Using Nested Panels 1178
Building the Menu System 1179
Building the ToolBar 1183
Building the StatusBar 1183
Finalizing the UI Design 1183
Implementing the MouseEnter/MouseLeave Event Handlers 1184
Implementing the Spell Checking Logic 1185
Understanding WPF Commands 1186

The Intrinsic Command Objects ... 1186
Connecting Commands to the Command Property ... 1187
Connecting Commands to Arbitrary Actions ... 1188
Working with the Open and Save Commands ... 1190
A Deeper Look at WPF APIs and Controls ... 1192
Working with the TabControl ... 1192
Building the Ink API Tab ... 1195
Designing the ToolBar ... 1196
The RadioButton Control ... 1199
Handling Events for the Ink API Tab ... 1201
The InkCanvas Control ... 1202
The ComboBox Control ... 1205
Saving, Loading, and Clearing InkCanvas Data ... 1206
Introducing the Documents API ... 1207
Block Elements and Inline Elements ... 1208
Document Layout Managers ... 1208
Building the Documents Tab ... 1209
Populating a FlowDocument Using Code ... 1210
Enabling Annotations and Sticky Notes ... 1211
Saving and Loading a Flow Document ... 1213
Introducing the WPF Data-Binding Model ... 1214
Building the Data Binding Tab ... 1214
Establishing Data Bindings Using Visual Studio ... 1215
The DataContext Property ... 1217
Data Conversion Using IValueConverter ... 1218
Establishing Data Bindings in Code ... 1219
Building the DataGrid Tab ... 1219
Summary ... 1221

Chapter 29: WPF Graphics Rendering Services 1223

Understanding WPF's Graphical Rendering Services 1223

WPF Graphical Rendering Options 1224

Rendering Graphical Data Using Shapes 1225

Adding Rectangles, Ellipses, and Lines to a Canvas 1227

Removing Rectangles, Ellipses, and Lines from a Canvas 1230

Working with Polylines and Polygons 1231

Working with Paths 1232

WPF Brushes and Pens 1236

Configuring Brushes Using Visual Studio 1236

Configuring Brushes in Code 1239

Configuring Pens 1240

Applying Graphical Transformations 1241

A First Look at Transformations 1242

Transforming Our Canvas Data 1243

Working with the Visual Studio Transform Editor 1245

Building the Initial Layout 1245

Applying Transformations at Design Time 1247

Transforming the Canvas in Code 1249

Rendering Graphical Data Using Drawings and Geometries 1250

Building a DrawingBrush Using Geometries 1251

Painting with the DrawingBrush 1252

Containing Drawing Types in a DrawingImage 1253

The Role of Expression Design 1254

Exporting a Sample Design File As XAML 1254

Importing the Graphical Data into a WPF Project 1256

Interacting with the Bear 1258

Rendering Graphical Data Using the Visual Layer 1258
Summary 1265
Chapter 30: WPF Resources, Animations, and Styles 1267
Understanding the WPF Resource System 1267
Working with Binary Resources 1267
Working with Object (Logical) Resources 1273
The Role of the Resources Property 1274
Defining Window-Wide Resources 1274
The {StaticResource} Markup Extension 1277
The {DynamicResource} Markup Extension 1277
Application-Level Resources 1278
Defining Merged Resource Dictionaries 1280
Defining a Resource-Only Assembly 1281
Understanding WPF's Animation Services 1284
The Role of the Animation Class Types 1284
The To, From, and By Properties 1285
The Role of the Timeline Base Class 1285
Authoring an Animation in C# Code 1286
Controlling the Pace of an Animation 1288
Reversing and Looping an Animation 1288
Authoring Animations in XAML 1289
The Role of Storyboards 1290
The Role of Event Triggers 1290
Animation Using Discrete Key Frames 1291
Understanding the Role of WPF Styles 1292
Defining and Applying a Style 1293
Overriding Style Settings 1294
Automatically Applying a Style with TargetType 1294

Subclassing Existing Styles 1295
The Role of Unnamed Styles 1295
Defining Styles with Triggers 1296
Defining Styles with Multiple Triggers 1297
Animated Styles 1297
Assigning Styles Programmatically 1298
Summary 1300
Chapter 31: Dependency Properties, Routed Events, and Templates 1301
Understanding the Role of Dependency Properties 1301
Examining an Existing Dependency Property 1303
Important Notes Regarding CLR Property Wrappers 1305
Building a Custom Dependency Property 1306
Adding a Data Validation Routine 1311
Responding to the Property Change 1312
Understanding Routed Events 1313
The Role of Routed Bubbling Events 1314
Continuing or Halting Bubbling 1315
The Role of Routed Tunneling Events 1315
Logical Trees, Visual Trees, and Default Templates 1317
Programmatically Inspecting a Logical Tree 1318
Programmatically Inspecting a Visual Tree 1319
Programmatically Inspecting a Control's Default Template 1321
Building a Control Template with the Trigger Framework 1325
Templates As Resources 1326
Incorporating Visual Cues Using Triggers 1329
The Role of the {TemplateBinding} Markup Extension 1329
The Role of ContentPresenter 1331
Incorporating Templates into Styles 1332

Summary 1333

Part VIII: ASP.NET Web Forms 1335

Chapter 32: Introducing ASP.NET Web Forms 1337

The Role of HTTP 1337

The HTTP Request/Response Cycle 1337

HTTP Is a Stateless Protocol 1338

Understanding Web Applications and Web Servers 1338

The Role of IIS Virtual Directories 1339

The ASP.NET Development Web Server 1339

The Role of HTML 1340

HTML Document Structure 1340

The Role of an HTML Form 1341

The Visual Studio HTML Designer Tools 1342

Building an HTML Form 1344

The Role of Client-Side Scripting 1346

A Client-Side Scripting Example 1347

Posting Back to the Web Server 1348

Postbacks Under ASP.NET 1349

An Overview of the ASP.NET API 1349

Major Features of ASP.NET 2.0 and Higher 1350

Major Features of ASP.NET 3.5 (and .NET 3.5 SP1) and Higher 1351

Major Features of ASP.NET 4.0 and 4.5 1352

Building a Single-File ASP.NET Web Page 1352

Referencing AutoLotDAL.dll 1354

Designing the UI 1354

Adding the Data Access Logic 1355

The Role of ASP.NET Directives 1358

Analyzing the "Script" Block 1359
Analyzing the ASP.NET Control Declarations 1360
Building an ASP.NET Web Page Using Code Files 1361
Referencing the AutoLotDAL.dll Assembly 1364
Updating the Code File 1365
Debugging and Tracing ASP.NET Pages 1365
ASP.NET Web Sites vs. ASP.NET Web Applications 1367
The ASP.NET Web Site Directory Structure 1369
Referencing Assemblies 1369
The Role of the App_Code Folder 1370
The Inheritance Chain of the Page Type 1371
Interacting with the Incoming HTTP Request 1372
Obtaining Browser Statistics 1374
Access to Incoming Form Data 1374
The IsPostBack Property 1375
Interacting with the Outgoing HTTP Response 1375
Emitting HTML Content 1376
Redirecting Users 1377
The Life Cycle of an ASP.NET Web Page 1378
The Role of the AutoEventWireup Attribute 1379
The Error Event 1380
The Role of the web.config File 1381
The ASP.NET Web Site Administration Utility 1382
Summary 1382
Chapter 33: ASP.NET Web Controls, Master Pages, and Themes 1383
Understanding the Nature of Web Controls 1383
Understanding Server-Side Event Handling 1384

The AutoPostBack Property 1384
The Control and WebControl Base Classes 1385
Enumerating Contained Controls 1386
Dynamically Adding and Removing Controls 1389
Interacting with Dynamically Created Controls 1390
Functionality of the WebControl Base Class 1391
Major Categories of ASP.NET Web Controls 1392
A Brief Word Regarding System.Web.UI.HtmlControls 1395
Web Control Documentation 1396
Building the ASP.NET Cars Web Site 1396
Working with ASP.NET Master Pages 1397
Defining the Default Content Page 1404
Designing the Inventory Content Page 1407
Designing the Build-a-Car Content Page 1412
The Role of the Validation Controls 1415
Enabling Client-Side JavaScript Validation Support 1417
The RequiredFieldValidator 1417
The RegularExpressionValidator 1418
The RangeValidator 1418
The CompareValidator 1418
Creating Validation Summaries 1419
Defining Validation Groups 1421
Working with Themes 1422
Understanding *.skin Files 1423
Applying Site-Wide Themes 1426
Applying Themes at the Page Level 1426
The SkinID Property 1426
Assigning Themes Programmatically 1427

Summary 1428

Chapter 34: ASP.NET State Management Techniques 1429

The Issue of State 1429

ASP.NET State Management Techniques 1431

Understanding the Role of ASP.NET View State 1432

Demonstrating View State 1432

Adding Custom View State Data 1434

The Role of the Global.asax File 1434

The Global Last-Chance Exception Event Handler 1436

The HttpApplication Base Class 1437

Understanding the Application/Session Distinction 1438

Maintaining Application-Level State Data 1438

Modifying Application Data 1440

Handling Web Application Shutdown 1442

Working with the Application Cache 1442

Fun with Data Caching 1443

Modifying the *.aspx File 1445

Maintaining Session Data 1448

Additional Members of HttpSessionState 1450

Understanding Cookies 1451

Creating Cookies 1452

Reading Incoming Cookie Data 1453

The Role of the <sessionState> Element 1454

Storing Session Data in the ASP.NET Session State Server 1454

Storing Session Data in a Dedicated Database 1455

Introducing the ASP.NET Profile API 1456

The ASPNETDB.mdf Database 1456

Defining a User Profile Within web.config 1457
Accessing Profile Data Programmatically 1458
Grouping Profile Data and Persisting Custom Objects 1460
Summary 1462
Index 1463

About the Author

Andrew Troelsen fondly recalls his very first computer, an Atari 400 complete with a tape deck storage device and a black-and-white TV serving as a monitor (which his parents permitted him to have in his bedroom—thanks guys!).

Andrew is employed with Intertech (`www.intertech.com`), a .NET/Java training and consulting center located in Minneapolis, Minnesota.

He has authored a number of books, including *Developer's Workshop to COM and ATL 3.0* (Wordware Publishing, 2000), *COM and .NET Interoperability* (Apress, 2002), and *Visual Basic 2008 and the .NET 3.5 Platform: An Advanced Guide* (Apress, 2008).

About the Technical Reviewer

Andy Olsen runs a software training company based in the UK, delivering training in .NET, Java, web, and mobile technologies in Europe, the U.S., and Asia. Andy has been working with .NET since the days of the first beta and is actively involved with the latest features in the .NET 4.5 platform. Andy lives by the sea in Swansea with his wife Jayne and their children, Emily and Tom. Andy enjoys running along the sea front (with copious coffee stops along the way), skiing, and following the Swans! Andy can be reached at `andyo@olsensoft.com`

Acknowledgments

One might think that it would be easier to update a book, rather than create a new book from the ground up. In my experience, I feel it is the exact opposite. While I am the individual responsible for the overall content, this book would never have been published if a number of people had not worked tirelessly beside me.

Huge thanks are in order to my technical editor, Andy Olsen. As always, Andy made many excellent suggestions (I only wish I had the time to incorporate everything—maybe in the next edition!).

To the folks at Apress, you have continued to show me why I am grateful to be working with you. Thanks to all for taking my raw manuscript and transforming it into a professional, high-quality, published text.

Introduction

Many moons ago (circa 2001), I was given the opportunity to write a book on a forthcoming Microsoft technology that was, at the time, dubbed NGWS (Next Generation Windows Software). As I began to examine the source code provided by Microsoft, I noticed numerous code comments referring to the "COOL" (Common Object Oriented Language) programming language.

While I worked on my first initial manuscript of *C# and the .NET Platform* using a pre-alpha build (and no documentation to speak of), NGWS was eventually rebranded as the Microsoft .NET platform. COOL, as you might guess, is what we now know today as C#.

The first edition of this book was released in step with .NET 1.0, beta 2. Since then, I have updated the text to account for the numerous updates to the C# programming language, as well as the explosion of new APIs introduced with each new release of the .NET platform.

Over the years, this book has (thankfully and gratefully) been very well received, by the press (a JOLT award finalist and ReferenceWare programming book of the year), readers, and various university programs in computer science and software engineering.

It has been just wonderful to communicate with readers and educators around the globe. Thank you for all of your suggestions, comments, and (yes) criticism. I might not be able to respond to every e-mail, but everything is taken under consideration, to be sure.

We're a Team, You and I

Technology authors write for a demanding group of people (I should know—I'm one of them). You know that building software solutions using any platform or language is extremely complicated and is very specific to your department, company, client base, and subject matter. Perhaps you work in the electronic publishing industry, develop systems for the state or local government, or work at NASA or a branch of the military. Speaking for myself, I have developed children's educational software (Oregon Trail/Amazon Trail), various n-tier systems, and projects within the medical and financial industries. The chances are almost 100 percent that the code you write at your place of employment has little to do with the code I write at mine (unless we happened to work together previously!).

Therefore, in this book, I have deliberately chosen to avoid creating demonstrations that tie the example code to a specific industry or vein of programming. Given this, I explain C#, OOP, the CLR, and the .NET base class libraries using industry-agnostic examples. Rather than having every blessed example fill a grid with data, calculate payroll, or whatnot, I'll stick to subject matter we can all relate to: automobiles (with some geometric structures and employee payroll systems thrown in for good measure). And that's where you come in.

My job is to explain the C# programming language and the core aspects of the .NET platform the best I possibly can. As well, I will do everything I can to equip you with the tools and strategies you need to continue your studies at this book's conclusion.

Your job is to take this information and apply it to your specific programming assignments. I obviously understand that your projects most likely don't revolve around automobiles with friendly pet names (Zippy the BMW, or a Yugo named Clunker, among others), but that's what applied knowledge is all about!

Rest assured, once you understand the topics and concepts presented within this text, you will be in a perfect position to build .NET solutions that map to your own unique programming environment.

An Overview of This Book

Pro C# 5.0 and the .NET 4.5 Framework, Sixth Edition, is logically divided into eight distinct parts, each of which contains a number of related chapters. Here is a part-by-part and chapter-by-chapter breakdown of the text.

Part I: Introducing C# and the .NET Platform

The purpose of Part 1 is to acclimate you to the nature of the .NET platform and various development tools (many of which are open source) used during the construction of .NET applications.

Chapter 1: The Philosophy of .NET

This first chapter functions as the backbone for the remainder of the text. The primary goal of this chapter is to acquaint you with a number of .NET-centric building blocks, such as the Common Language Runtime (CLR), Common Type System (CTS), Common Language Specification (CLS), and base class libraries. Here, you will take an initial look at the C# programming language and the .NET assembly format. As well, you will learn the role of the .NET platform within the Windows 8 operating system, and understand the distinction between a Windows 8 app and a .NET application.

Chapter 2: Building C# Applications

The goal of this chapter is to introduce you to the process of compiling C# source code files using various tools and techniques. You will begin by learning how to use the command-line compiler (`csc.exe`) and C# response files. Over the remainder of the chapter, you will examine numerous code editors and integrated development environments (IDEs), including Notepad++, SharpDevelop, Visual C# Express, and Visual Studio. You will also learn how to configure your development machine with a local installation of the all-important .NET Framework 4.5 SDK documentation.

Part II: Core C# Programming

The topics presented in this part of the book are quite important because you will use them regardless of which type of .NET software you intend to develop (e.g., web applications, desktop GUI applications, code libraries, or Windows services). Here, you will learn about the fundamental data types of .NET, work with text manipulation, and learn the role of various C# parameter modifiers (including optional and named arguments).

Chapter 3: Core C# Programming Constructs, Part I

This chapter begins your formal investigation of the C# programming language. Here, you will learn about the role of the `Main()` method and numerous details regarding the intrinsic data types of the .NET platform, including the manipulation of textual data using `System.String` and `System.Text.StringBuilder`. You will also examine iteration and decision constructs, narrowing and widening operations, and the `unchecked` keyword.

Chapter 4: Core C# Programming Constructs, Part II

This chapter completes your examination of the core aspects of C#, beginning with the construction of overloaded type methods and defining parameters using the `out`, `ref`, and `params` keywords. This chapter will examine two C# features called *arguments* and *optional parameters*. You will also learn how to create and manipulate arrays of data, define nullable data types (with the `?` and `??` operators), and understand the distinction between value types (including enumerations and custom structures) and reference types.

Part III: Object-Oriented Programming with C#

In this section you will come to understand the core constructs of the C# language, including the details of *object-oriented programming* (OOP). This part will also examine how to process runtime exceptions, and dive into the details of working with strongly typed interfaces.

Chapter 5: Understanding Encapsulation

This chapter begins your examination of object-oriented programming (OOP) using the C# programming language. After you are introduced to the pillars of OOP (encapsulation, inheritance, and polymorphism), the remainder of this chapter will show you how to build robust class types using constructors, properties, static members, constants, and read-only fields. You will wrap up with an examination of partial type definitions, object initialization syntax, and automatic properties.

Chapter 6: Understanding Inheritance and Polymorphism

Here, you will examine the remaining pillars of OOP (inheritance and polymorphism), which allow you to build families of related class types. As you do this, you will examine the role of virtual methods, abstract methods (and abstract base classes), and the nature of the polymorphic interface. Last but not least, this chapter will explain the role of the supreme base class of the .NET platform, `System.Object`.

Chapter 7: Understanding Structured Exception Handling

The point of this chapter is to discuss how to handle runtime anomalies in your code base through the use of *structured exception handling*. Not only will you learn about the C# keywords that allow you to handle such problems (`try`, `catch`, `throw`, and `finally`), but you will also come to understand the

distinction between application-level and system-level exceptions. In addition, this chapter will examine various tools within Visual Studio that allow you to debug the exceptions that escape your notice.

Chapter 8: Working with Interfaces

The material in this chapter builds upon your understanding of object-based development by covering the topic of *interface-based programming*. Here, you will learn how to define classes and structures that support multiple behaviors, how to discover these behaviors at runtime, and how to selectively hide particular behaviors using explicit interface implementation. In addition to creating a number of custom interfaces, you will also learn how to implement standard interfaces found within the .NET platform. You will use these to build objects that can be sorted, copied, enumerated, and compared.

Part IV: Advanced C# Programming

This section of the book will deepen your understanding of the C# language by walking you through a number of more advanced (but very important) concepts. Here, you will complete your examination of the .NET type system by examining interfaces and delegates. You will also learn about the role of generics, take a first look at Language Integrated Query (LINQ), and examine a number of more advanced features of C# (e.g., extension methods, partial methods, and pointer manipulation).

Chapter 9: Collections and Generics

This chapter explores the topic of *generics*. As you will see, generic programming gives you a way to create types and type members, which contain various *placeholders* that can be specified by the caller. In a nutshell, generics greatly enhance application performance and type safety. Not only will you explore various generic types within the `System.Collections.Generic` namespace, but you will also learn how to build your own generic methods and types (with and without constraints).

Chapter 10: Delegates, Events, and Lambda Expressions

The purpose of Chapter 10 is to demystify the *delegate* type. Simply put, a .NET delegate is an object that *points* to other methods in your application. Using this type, you can build systems that allow multiple objects to engage in a two-way conversation. After you have examined the use of .NET delegates, you will then be introduced to the C# `event` keyword, which you can use to simplify the manipulation of raw delegate programming. You will wrap up this chapter by investigating the role of the C# lambda operator (`=>`) and exploring the connection between delegates, anonymous methods, and lambda expressions.

Chapter 11: Advanced C# Language Features

This chapter deepens your understanding of the C# programming language by introducing you to a number of advanced programming techniques. Here, you will learn how to overload operators and create custom conversion routines (both implicit and explicit) for your types. You will also learn how to build and interact with *type indexers*, as well as work with *extension methods*, *anonymous types*, *partial methods*, and C# pointers using an *unsafe* code context.

Chapter 12: LINQ to Objects

This chapter will begin your examination of Language Integrated Query (LINQ). LINQ allows you to build strongly typed *query expressions* that can be applied to a number of LINQ targets to manipulate *data* in the broadest sense of the word. Here, you will learn about LINQ to Objects, which allows you to apply LINQ expressions to containers of data (e.g., arrays, collections, and custom types). This information will serve you well as you encounter a number of additional LINQ APIs throughout the remainder of this book (e.g., LINQ to XML, LINQ to DataSet, PLINQ, and LINQ to Entities).

Chapter 13: Understanding Object Lifetime

The final chapter of this section examines how the CLR manages memory using the .NET garbage collector. Here you will come to understand the role of application roots, object generations, and the `System.GC` type. Once you understand the basics, you will examine the topics of *disposable objects* (using the `IDisposable` interface) and the finalization process (using the `System.Object.Finalize()` method). This chapter will also investigate the `Lazy<T>` class, which allows you to define data that will not be allocated until requested by a caller. As you will see, this feature can be very helpful when you want to ensure you do not clutter the heap with objects that are not actually required by your programs.

Part V: Programming with .NET Assemblies

Part 5 dives into the details of the .NET assembly format. Not only will you learn how to deploy and configure .NET code libraries, but you will also come to understand the internal composition of a .NET binary image. This part also explains the role of .NET attributes and the role of resolving type information at runtime. This section will also explain the role of the Dynamic Language Runtime (DLR) and the C# `dynamic` keyword. Later chapters will examine some fairly advanced topics regarding assemblies, such as application domains, the syntax of CIL, and the construction of in-memory assemblies.

Chapter 14: Building and Configuring Class Libraries

At a very high level, *assembly* is the term used to describe a `*.dll` or `*.exe` binary file created with a .NET compiler. However, the true story of .NET assemblies is far richer than that. Here, you will learn the distinction between single-file and multifile assemblies, as well as how to build and deploy each entity. You'll also examine how you can configure private and shared assemblies using XML-based `*.config` files and publisher policy assemblies. Along the way, you will investigate the internal structure of the global assembly cache (GAC).

Chapter 15: Type Reflection, Late Binding, and Attribute-Based Programming

Chapter 15 continues your examination of .NET assemblies by checking out the process of runtime type discovery using the `System.Reflection` namespace. Using the types of this namespace, you can build applications that can read an assembly's metadata on the fly. You will also learn how to load and create types at runtime dynamically using *late binding*. The final topic of this chapter will explore the role of

.NET attributes (both standard and custom). To illustrate the usefulness of each of these topics, the chapter shows you how to construct an extendable Windows Forms application.

Chapter 16: Dynamic Types and the Dynamic Language Runtime

.NET 4.0 introduces a new aspect of the .NET runtime environment called the *dynamic language runtime*. Using the DLR and the C# 2010 `dynamic` keyword, you can define data that is not truly resolved until runtime. Using these features simplifies some very complex .NET programming tasks dramatically. In this chapter, you will learn some practical uses of dynamic data, including how to leverage the .NET reflection APIs in a streamlined manner, as well as how to communicate with legacy COM libraries with a minimum of fuss and bother.

Chapter 17: Processes, AppDomains, and Object Contexts

Now that you have a solid understanding of assemblies, this chapter dives deeper into the composition of a loaded .NET executable. The goal of this chapter is to illustrate the relationship between processes, application domains, and contextual boundaries. These topics provide the proper foundation for Chapter 19, where you will examine the construction of multithreaded applications.

Chapter 18: Understanding CIL and the Role of Dynamic Assemblies

The goal of the final chapter in this section is twofold. In the first half (more or less), you will examine the syntax and semantics of CIL in much greater detail than in previous chapters. The remainder of this chapter will cover the role of the `System.Reflection.Emit` namespace. You can use these types to build software that can generate .NET assemblies in memory at runtime. Formally speaking, assemblies defined and executed in memory are termed *dynamic assemblies*.

Part VI: Introducing the .NET Base Class Libraries

By this point in the text, you have a solid handle on the C# language and the details of the .NET assembly format. Part 6 leverages your newfound knowledge by exploring a number of commonly used services found within the base class libraries, including the creation of multithreaded applications, file I/O, and database access using ADO.NET. This part also covers the construction of distributed applications using Windows Communication Foundation (WCF), workflow-enabled applications that use the Windows Workflow Foundation (WF) API, and the LINQ to XML API.

Chapter 19: Multithreaded, Parallel, and Async Programming

This chapter examines how to build multithreaded applications and illustrates a number of techniques you can use to author thread-safe code. The chapter opens by revisiting the .NET delegate type to ensure, explaining a delegate's intrinsic support for asynchronous method invocations. Next, you will investigate the types within the `System.Threading` namespace. The remainder of this chapter covers the *Task Parallel Library* (TPL). Using the TPL, .NET developers can build applications that distribute their workload across all available CPUs in a wickedly simple manner. At this point, you will also learn about

the role of *Parallel LINQ* (PINQ), which provides a way to create LINQ queries that scale across multiple machine cores. We wrap up by examining some new C# keywords introduced in .NET 4.5, which integrate asynchronous method calls directly into the language.

Chapter 20: File I/O and Object Serialization

The `System.IO` namespace allows you to interact with a machine's file and directory structure. Over the course of this chapter, you will learn how to create (and destroy) a directory system programmatically. You will also learn how to move data into and out of various streams (e.g., file based, string based, and memory based). The latter part of this chapter will examine the object serialization services of the .NET platform. Simply put, *serialization* allows you to persist the state of an object (or a set of related objects) into a stream for later use. *Deserialization* (as you might expect) is the process of plucking an object from the stream into memory for consumption by your application. After you understand the basics, you will learn how to customize the serialization process using the `ISerializable` interface and a set of .NET attributes.

Chapter 21: ADO.NET Part I: The Connected Layer

In this first of three database-centric chapters, you will take your first look at the database access API of the .NET platform, ADO.NET. Specifically, this chapter will introduce you to the role of .NET data providers and how to communicate with a relational database using the *connected layer* of ADO.NET, which is represented by connection objects, command objects, transaction objects, and data reader objects. Be aware that this chapter will also walk you through the creation of a custom database and the first iteration of a custom data access library (`AutoLotDAL.dll`); you will use this library throughout the remainder of this book.

Chapter 22: ADO.NET Part II: The Disconnected Layer

This chapter continues your study of database manipulation by examining the *disconnected layer* of ADO.NET. Here, you will learn the role of the `DataSet` type and data adapter objects. You will also learn about the many tools of Visual Studio 2010 that can greatly simplify the creation of data-driven applications. Along the way, you will learn how to bind `DataTable` objects to user interface elements, as well as how to apply LINQ queries to in-memory `DataSet` objects using LINQ to DataSet.

Chapter 23: ADO.NET Part III: The Entity Framework

This chapter wraps up your investigation of ADO.NET by examining the role of the *Entity Framework* (EF). Essentially, EF is a way for you to author data-access code using strongly typed classes that directly map to your business model. Here, you will come to understand the role of EF Object Services, the Entity Client and Object Context, and the composition of an `*.edmx` file. While doing so, you will learn to interact with relational databases using LINQ to Entities. You will also build the final version of your custom data-access library (`AutoLotDAL.dll`), which you will use in several of the remaining chapters of the book.

Chapter 24: Introducing LINQ to XML

Chapter 14 introduced you to the core LINQ programming model—specifically LINQ to Objects. Here, you will deepen your understanding of Language Integrated Query by examining how to apply LINQ queries to XML documents. You will begin by learning about the "warts" that were present in .NET's initial foray into XML manipulation as you use the types of the `System.Xml.dll` assembly. With this brief history lesson behind you, you will explore how to create XML documents in memory, how to persist them to the hard drive, and how to navigate their contents using the LINQ programming model (LINQ to XML).

Chapter 25: Introducing Windows Communication Foundation

Until this point in the book, all of the sample applications have executed on a single computer. In this chapter, you will learn about the *Windows Communication Foundation* (WCF) API that allows you to build distributed applications in a symmetrical manner, regardless of their underlying plumbing. This chapter will expose you to the construction of WCF services, hosts, and clients. As you will see, WCF services are extremely flexible because they allow clients and hosts to leverage XML-based configuration files to specify addresses, bindings, and contracts declaratively.

Chapter 26: Introducing Windows Workflow Foundation

In this chapter, you will begin by learning about the role of a workflow-enabled application, and you will come to understand the various ways to model business processes using the .NET 4.0 WF API. Next, you will examine the scope of the WF activity library, as well as learn how to build custom activities that will use the custom database access library you created earlier in the book.

Part VII: Windows Presentation Foundation

.NET 3.0 introduced programmers to an amazing API called *Windows Presentation Foundation* (WFP). This API has quickly become the heir apparent to the Windows Forms desktop programming model. In essence, WPF allows you to build desktop applications that incorporate vector graphics, interactive animations, and data-binding operations using a declarative markup grammar called *XAML*. Furthermore, the WPF control architecture provides a trivial way to restyle the look-and-feel of a typical control radically using little more than some well-formed XAML.

Chapter 27: Introducing Windows Presentation Foundation and XAML

Essentially, WPF allows you to build extremely interactive and media-rich front ends for desktop applications (and indirectly, web applications). Unlike Windows Forms, this supercharged UI framework integrates a number of key services (e.g., 2D and 3D graphics, animations, and rich documents) into a single, unified object model. In this chapter, you will begin your examination of WPF and the Extendable Application Markup Language (XAML). Here, you will learn how to build WPF programs without XAML, as well as using nothing but XAML, and by using a combination of both approaches. You will wrap up the chapter by building a custom XAML editor that you will use for the remainder of the WPF-centric chapters.

Chapter 28: Programming with WPF Controls

This chapter will expose you to the process of using intrinsic WPF controls and layout managers. For example, you will learn to build menu systems, splitter windows, toolbars, and status bars. This chapter will also introduce you to a number of WPF APIs (and their related controls), including the WPF Documents API, the WPF Ink API, and the data-binding model. Just as importantly, this chapter will begin your investigation of Expression Blend IDE, which simplifies the task of creating rich UIs for a WPF application.

Chapter 29: WPF Graphics Rendering Services

WPF is a graphically intensive API; given this fact, WPF provides three ways to render graphics: *shapes*, *drawings and geometrics*, and *visuals*. In this chapter, you will evaluate each option and learn about a number of important graphics primitives (e.g., brushes, pens, and transformations) along the way. This chapter will also examine a number of ways in which Expression Blend can help you simplify the process of creating WPF graphics, as well as how to perform hit-testing operations against graphical data.

Chapter 30: WPF Resources, Animations, and Styles

This chapter will introduce you to three important (and interrelated) topics that will deepen your understanding of the Windows Presentation Foundation API. The first order of business is to learn the role of *logical resources*. As you will see, the logical resource (also termed an *object resource*) system provides a way for you to name and refer to commonly used objects within a WPF application. Next, you will learn how to define, execute, and control an *animation* sequence. Despite what you might be thinking, however, WPF animations are not limited to the confines of video game or multimedia applications. You will wrap up the chapter by learning about the role of WPF *styles*. Similar to a web page that uses CSS or the ASP.NET theme engine, a WPF application can define a common look-and-feel for a set of controls.

Chapter 31: Dependency Properties, Routed Events, and Templates

This chapter begins by examining two topics that are important when creating a custom control: *dependency properties* and *routed events*. After you understand these topics, you will learn about the role of a *default template*, as well as how to view them programmatically at runtime. After this foundation has been laid, the remainder of this chapter will examine how to build custom templates.

Part VIII: ASP.NET Web Forms

Part 8 is devoted to an examination of constructing web applications using the ASP.NET programming API. Microsoft designed ASP.NET to model the creation of desktop user interfaces by layering an event-driven, object-oriented framework on top of a standard HTTP request/response.

Chapter 32: Introducing ASP.NET Web Forms

This chapter begins your study of web application development using ASP.NET. As you will see, server-side scripting code has now been replaced with real object-oriented languages (e.g., C# and VB .NET). This chapter will examine the construction of an ASP.NET web page, the underlying programming model, and other key aspects of ASP.NET, such as your choice of web server and the use of `web.config` files.

Chapter 33: ASP.NET Web Controls, Master Pages, and Themes

Whereas the previous chapter showed you how to construct ASP.NET `Page` objects, this chapter will examine the controls that populate the internal control tree. Here, you will examine the core ASP.NET *web controls*, including validation controls, the intrinsic site navigation controls, and various data-binding operations. This chapter will also illustrate the role of *master pages* and the ASP.*NET theme engine*, which is a server-side alternative to traditional style sheets.

Chapter 34: ASP.NET State Management Techniques

This chapter extends your understanding of ASP.NET by examining various ways to handle state management under .NET. Like classic ASP, ASP.NET allows you to create cookies and application-level and session-level variables quite easily. However, ASP.NET also introduces a new state management technique: the *application cache*. After you look at the numerous ways to handle state with ASP.NET, you will examine the role of the `HttpApplication` base class and learn how to alter the runtime behavior of your web application dynamically using the `web.config` file.

Downloadable Appendixes

As if 34 chapters were not enough, I have made two additional chapters available for download from the home page of this book at the Apress web site (`www.apress.com`). The first appendix covers the basics of the Windows Forms API, which is used for a few of the UI examples in this text. The second appendix examines the platform-independent nature of .NET via the Mono platform.

Downloadable Appendix A: Programming with Windows Forms

The original desktop GUI toolkit that shipped with the .NET platform is called *Windows Forms*. This appendix will walk you through the role of this UI framework and illustrate how to build main windows, dialog boxes, and menu systems. You will also learn about the role of form inheritance and see how to render 2D graphical data using the `System.Drawing` namespace. You will wrap things up by building a (semicapable) painting application that illustrates the various topics discussed throughout this appendix.

Downloadable Appendix B: Platform-Independent .NET Development with Mono

Last but not least, Appendix B covers how to use an open source implementation of the .NET platform named *Mono*. You can use Mono to build feature-rich .NET applications that can be created, deployed, and executed upon a variety of operating systems, including Mac OS X, Solaris, and numerous Linux distributions. Given that Mono is largely comparable with Microsoft's .NET platform, you already know most of what Mono has to offer. Therefore, this appendix will focus on the Mono installation process, Mono development tools, and Mono runtime engine.

Obtaining This Book's Source Code

You can find all of the code examples contained in this book available as a free download from the Source Code/Download area of the Apress website. Simply navigate to `www.apress.com`, select the Source Code/Download link, and look up this title by name. Once you are on the home page for Pro C# 5

With this book, you can download a self-extracting `*.zip` file. After you unzip the contents, you will find that the code has been partitioned on a chapter-by-chapter basis.

On a related note, be aware that you will find "Source Code" notes, such as the following, in many of the book's chapters. These notes serve as your visual cue that you can load the example under discussion into Visual Studio for further examination and modification.

■ **Source Code** This is a source code note that refers you to a specific directory in the ZIP archive.

To open a solution into Visual Studio, use the File ➤ Open ➤ Project/Solution menu option, and then navigate to the correct `*.sln` file within the correct subdirectory of the unzipped archive.

Obtaining Updates for This Book

As you read through this text, you might find an occasional grammatical or code error (although I sure hope not). If this is the case, please accept my apologies. Being human, I am sure that a glitch or two might be present, despite my best efforts. If this is the case, you can obtain the current errata list from the Apress web site at `www.apress.com` (again, this is located on the home page for this book), as well as information on how to notify me of any errors you might find.

Contacting Me

If you have any questions regarding this book's source code, are in need of clarification for a given example, or simply would like to offer your thoughts regarding the .NET platform, feel free to drop me a line at the following e-mail address (to ensure your messages don't end up in my junk mail folder, please include "C# SixthEd" in the Subject line somewhere):

atroelsen@intertech.com

Please understand that I will do my best to get back to you in a timely fashion; however, like yourself, I get busy from time to time. If I don't respond within a week or two, please be aware that I am not trying to be a jerk, nor am I trying to avoid talking to you. I'm just busy (or, if I'm lucky, on vacation somewhere). So, then! Thanks for buying this text (or at least looking at it in the bookstore while you try to decide whether you will buy it). I hope you enjoy reading this book and putting your newfound knowledge to good use.

— Andrew Troelsen