



Professional

C#

3rd Edition

Simon Robinson, Christian Nagel, Karli Watson, Jay Glynn, Morgan Skinner, Bill Evjen



Updates, source code, and Wrox technical support at www.wrox.com

Professional C#

Third Edition

Simon Robinson

Christian Nagel

Jay Glynn

Morgan Skinner

Karli Watson

Bill Evjen



WILEY

Wiley Publishing, Inc.

Professional C#

Third Edition

Professional C#

Third Edition

Simon Robinson

Christian Nagel

Jay Glynn

Morgan Skinner

Karli Watson

Bill Evjen



WILEY

Wiley Publishing, Inc.

Professional C#, Third Edition

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2004 by Wiley Publishing, Inc., Indianapolis, Indiana. All rights reserved.

Published simultaneously in Canada

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4447, E-mail: permcoordinator@wiley.com.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Trademarks: Wiley, the Wiley Publishing logo, Wrox, the Wrox logo, and Programmer to Programmer are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Library of Congress Control Number: 2004103177

ISBN: 0-7645-5759-9

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

About the Authors



Simon Robinson

Simon Robinson is the editor-in-chief of ASP Today, one of the leading sites related to Web programming on the Windows platform.

Simon's first experience of commercial computer programming was in the early 1980s, when a computer project he was working on at college became the school's student timetabling program, running on the BBC Micro. Later he studied for a Ph.D. in physics and subsequently spent a couple of years working as a university physics researcher. From there he moved on to working as a computer programmer, then writing books about programming, and finally on to his present job at ASP Today.

He has an extremely broad experience of programming on Windows. These days his core specialty is .NET programming. He is comfortable coding in C++, C#, VB, and IL, and has skills ranging from graphics and Windows Forms to ASP.NET to directories and data access to Windows services and the native Windows API.

Simon lives in Lancaster, UK. His outside interests include theater, dance, performing arts, and politics. You can visit Simon's Web site, <http://www.SimonRobinson.com>.



Christian Nagel

Christian Nagel is an independent software architect and developer who offers training and consulting on how to design and develop Microsoft .NET solutions. He looks back to more than 15 years' experience as a developer and software architect. Christian started his computing career with PDP 11 and VAX/VMS platforms, covering a variety of languages and platforms. Since the year 2000—when .NET was just a technology preview—he has been working with various .NET technologies to build distributed solutions. With his profound knowledge of Microsoft technologies, he has also written numerous .NET books; is certified as Microsoft Certified Trainer (MCT), Solution Developer (MCSD), and Systems Engineer (MCSE); and is the

Microsoft Regional Director for Austria. Christian is a speaker at international conferences (TechED, DevDays, VCDC) and is the regional manager of INETA Europe (International .NET User Group Association) supporting .NET user groups. You can contact Christian via his Web site, <http://www.christiannagel.com>.



Jay Glynn

Jay Glynn started writing software nearly 20 years ago, writing applications for the PICK operating system using PICK basic. Since then, he has created software using Paradox PAL and Object PAL, Delphi, VBA, Visual Basic, C, C++, Java, and of course C#. He is currently a Project coordinator and Architect for a large financial services company in Nashville, Tennessee, working on software for the TabletPC platform. He can be contacted at jlsglynn@hotmail.com.



Morgan Skinner

Morgan Skinner began his computing career at a tender age on a Sinclair ZX80 at school, where he was underwhelmed by some code a teacher had written and so began programming in assembly language. After getting hooked on Z80 (which he believes is far better than those paltry 3 registers on the 6502), he graduated through the school's ZX81s to his own ZX Spectrum.

Since then he's used all sorts of languages and platforms, including VAX Macro Assembler, Pascal, Modula2, Smalltalk, X86 assembly language, PowerBuilder, C/C++, VB, and currently C#. He's been programming in .NET since the PDC release in 2000, and liked it so much, he joined Microsoft in 2001. He now works in Premier Support for Developers and spends most of his time assisting customers with C#.

You can reach Morgan at <http://www.morganskinner.com>.

Karli Watson



Karli Watson is a freelance author and the technical director of 3form Ltd (<http://www.3form.net>). Despite starting out by studying nanoscale physics, the lure of cold, hard cash proved too much and dragged Karli into the world of computing. He has since written numerous books on .NET and related technologies, SQL, mobile computing, and a novel that has yet to see the light of day (but that doesn't have any computers in it). Karli is also known for his multicolored clothing, is a snowboarding enthusiast, and still wishes he had a cat.

Bill Evjen



Bill Evjen is an active proponent of the .NET technologies and community-based learning initiatives for .NET. He has been actively involved with .NET since the first bits were released in 2000 and has since become president of the St. Louis .NET User Group (<http://www.stlusergroups.org>). Bill is also the founder and executive director of the International .NET ssoication (<http://www.ineta.org>), which represents more than 125,000 members worldwide. Based in St. Louis, Missouri, USA, Bill is an acclaimed author and speaker on ASP.NET and XML Web services. He has written *XML Web Services for ASP.NET*, *Web Services Enhancements: Understanding the WSE for Enterprise Applications*, *Visual Basic .NET Bible*, and *ASP.NET Professional Secrets* (all published by Wiley). Bill is a Technical Director for Reuters, the

international news and financial services company. He graduated from Western Washington University in Bellingham, Washington, with a Russian language degree. You can reach Bill at evjen@yahoo.com.

Contributor

Allen Jones

Allen Jones has a career spanning 15 years that covers a broad range of IT disciplines, including enterprise management, solution and enterprise architecture, and project management. But software development has always been Allen's passion. Allen has architected and developed Microsoft Windows-based solutions since 1990, including a variety of e-commerce, trading, and security systems.

Allen has co-authored four popular .NET books including the *C# Programmer's Cookbook* (Microsoft Press) and *Programming .NET Security* (O'Reilly), and he is actively involved in the development of courseware for Microsoft Learning covering emerging .NET technologies.

Credits

Vice President and Executive Group Publisher

Richard Swadley

Editorial Manager

Kathryn A. Malm

Vice President and Executive Publisher

Bob Ipsen

Development Editor

Sharon Nash

Vice President and Publisher

Joseph B. Wikert

Production Editor

Eric Newman

Executive Editorial Director

Mary Bednarek

Text Design & Composition

Wiley Indianapolis Composition Services

Acquisitions Editors

Sharon Cox

Katie Mohr

Contents

Introduction	xxvii
---------------------	--------------

Part I: The C# Language	1
--------------------------------	----------

Chapter 1: .NET Architecture	3
-------------------------------------	----------

The Relationship of C# to .NET	4
The Common Language Runtime	4
Advantages of Managed Code	4
A Closer Look at Intermediate Language	7
Support for Object Orientation and Interfaces	8
Distinct Value and Reference Types	9
Strong Data Typing	9
Error Handling with Exceptions	16
Use of Attributes	17
Assemblies	17
Private Assemblies	18
Shared Assemblies	19
Reflection	19
.NET Framework Classes	19
Namespaces	21
Creating .NET Applications Using C#	21
Creating ASP.NET Applications	21
Creating Windows Forms	24
Windows Services	24
The Role of C# in the .NET Enterprise Architecture	24
Summary	26

Chapter 2: C# Basics	29
-----------------------------	-----------

Before We Start	30
Our First C# Program	30
The Code	30
Compiling and Running the Program	31

Contents

A Closer Look	31
Variables	34
Initialization of Variables	34
Variable Scope	35
Constants	38
Predefined Data Types	39
Value Types and Reference Types	39
CTS Types	40
Predefined Value Types	41
Predefined Reference Types	44
Flow Control	47
Conditional Statements	47
Loops	51
Jump Statements	54
Enumerations	55
Arrays	57
Namespaces	58
The using Statement	59
Namespace Aliases	60
The Main() Method	61
Multiple Main() Methods	61
Passing Arguments to Main()	62
More on Compiling C# Files	63
Console I/O	65
Using Comments	67
Internal Comments Within the Source Files	67
XML Documentation	68
The C# Preprocessor Directives	70
#define and #undef	70
#if, #elif, #else, and #endif	71
#warning and #error	72
#region and #endregion	72
#line	72
C# Programming Guidelines	73
Rules for Identifiers	73
Usage Conventions	74
Summary	81
Chapter 3: Objects and Types	83
Classes and Structs	84
Class Members	85
Data Members	85
Function Members	85

readonly Fields	99
Structs	101
Structs Are Value Types	102
Structs and Inheritance	103
Constructors for Structs	103
The Object Class	104
System.Object Methods	104
The ToString() Method	105
Summary	107
Chapter 4: Inheritance	109
<hr/>	
Types of Inheritance	109
Implementation Versus Interface Inheritance	109
Multiple Inheritance	110
Structs and Classes	110
Implementation Inheritance	111
Virtual Methods	112
Hiding Methods	113
Calling Base Versions of Functions	114
Abstract Classes and Functions	115
Sealed Classes and Methods	115
Constructors of Derived Classes	116
Modifiers	122
Visibility Modifiers	122
Other Modifiers	123
Interfaces	123
Defining and Implementing Interfaces	125
Derived Interfaces	128
Summary	130
Chapter 5: Operators and Casts	131
<hr/>	
Operators	131
Operator Shortcuts	133
The Ternary Operator	134
The checked and unchecked Operators	134
The is Operator	135
The as Operator	136
The sizeof Operator	136
The typeof Operator	136

Contents

Operator Precedence	137
Type Safety	137
Type Conversions	138
Boxing and Unboxing	141
Comparing Objects for Equality	142
Comparing Reference Types for Equality	142
The ReferenceEquals() Method	142
The virtual Equals() Method	143
The static Equals() Method	143
Comparison Operator (==)	143
Comparing Value Types for Equality	143
Operator Overloading	144
How Operators Work	145
Operator Overloading Example: The Vector Struct	146
Which Operators Can You Overload?	153
User-Defined Casts	154
Implementing User-Defined Casts	155
Multiple Casting	161
Summary	165
Chapter 6: Delegates and Events	167
Delegates	167
Using Delegates in C#	169
SimpleDelegate Example	172
BubbleSorter Example	174
Multicast Delegates	177
Events	179
The Receiver's View of Events	180
Generating Events	182
Summary	186
Chapter 7: Memory Management and Pointers	187
Memory Management under the Hood	187
Value Data Types	188
Reference Data Types	190
Garbage Collection	192
Freeing Unmanaged Resources	193
Destructors	193
The IDisposable Interface	195

Implementing IDisposable and a Destructor	196
Unsafe Code	197
Pointers	198
Pointer Example: PointerPlayaround	207
Using Pointers to Optimize Performance	212
Summary	216
Chapter 8: Strings and Regular Expressions	217
<hr/>	
System.String	218
Building Strings	219
Format Strings	223
Regular Expressions	229
Introduction to Regular Expressions	229
The RegularExpressionsPlayaround Example	230
Displaying Results	233
Matches, Groups, and Captures	234
Summary	237
Chapter 9: Collections	239
<hr/>	
Examining Groups of Objects	239
Array Lists	240
Collections	241
Dictionaries	245
Summary	256
Chapter 10: Reflection	257
<hr/>	
Custom Attributes	258
Writing Custom Attributes	258
Custom Attribute Example: WhatsNewAttributes	262
Reflection	265
The System.Type Class	266
The TypeView Example	268
The Assembly Class	271
Completing the WhatsNewAttributes Sample	272
Summary	276

Contents

Chapter 11: Errors and Exceptions **277**

Looking into Errors and Exception Handling	277
Exception Classes	278
Catching Exceptions	280
User-Defined Exception Classes	290
Summary	297

Part II: The .NET Environment **299**

Chapter 12: Visual Studio .NET **301**

Working with Visual Studio .NET 2003	301
Creating a Project	304
Solutions and Projects	311
Windows Application Code	314
Reading in Visual Studio 6 Projects	314
Exploring and Coding a Project	315
Building a Project	326
Debugging	331
Other .NET Tools	334
The ASP.NET Web Matrix Project	335
WinCV	335
Summary	337

Chapter 13: Assemblies **339**

What Are Assemblies?	339
The Answer to DLL Hell	340
Features of Assemblies	341
Application Domains and Assemblies	341
Assembly Structure	344
Assembly Manifests	346
Namespaces, Assemblies, and Components	346
Private and Shared Assemblies	347
Viewing Assemblies	347
Building Assemblies	348
Cross-Language Support	353
The CTS and the CLS	353
Language Independence in Action	354
CLS Requirements	364

Global Assembly Cache	366
Native Image Generator	366
Global Assembly Cache Viewer	367
Global Assembly Cache Utility (gacutil.exe)	368
Creating Shared Assemblies	369
Shared Assembly Names	369
Creating a Shared Assembly	371
Configuration	376
Configuration Categories	376
Versioning	377
Configuring Directories	387
Summary	390
Chapter 14: .NET Security	391
Code Access Security	392
Code Groups	393
Code Access Permissions and Permissions Sets	399
Policy Levels: Machine, User, and Enterprise	403
Support for Security in the Framework	405
Demanding Permissions	406
Requesting Permissions	407
Implicit Permission	410
Denying Permissions	411
Asserting Permissions	412
Creating Code Access Permissions	414
Declarative Security	414
Role-Based Security	415
The Principal	415
Windows Principal	416
Roles	417
Declarative Role-Based Security	418
Managing Security Policy	419
The Security Configuration File	419
Managing Code Groups and Permissions	423
Turning Security On and Off	423
Resetting Security Policy	423
Creating a Code Group	423
Deleting a Code Group	424
Changing a Code Group's Permissions	424
Creating and Applying Permissions Sets	425
Distributing Code Using a Strong Name	427

Contents

Distributing Code Using Certificates	429
Managing Zones	435
Summary	437
Chapter 15: Threading	439
Threading	439
Applications with Multiple Threads	441
Manipulating Threads	441
The ThreadPlayaround Sample	444
Thread Priorities	448
Synchronization	449
Summary	453
Chapter 16: Distributed Applications with .NET Remoting	455
What Is .NET Remoting?	456
Application Types and Protocols	456
CLR Object Remoting	457
.NET Remoting Overview	457
Contexts	460
Activation	461
Attributes and Properties	461
Communication between Contexts	462
Remote Objects, Clients, and Servers	462
Remote Objects	462
A Simple Server	464
A Simple Client	465
.NET Remoting Architecture	466
Channels	466
Formatters	470
ChannelServices and RemotingConfiguration	471
Object Activation	472
Message Sinks	476
Passing Objects in Remote Methods	476
Lifetime Management	481
Miscellaneous .NET Remoting Features	484
Configuration Files	484
Hosting Applications	494
Classes, Interfaces, and SoapSuds	495
Asynchronous Remoting	498
Remoting and Events	499
Call Contexts	505
Summary	507

Chapter 17: Localization	509
Namespace System.Globalization	510
Unicode Issues	510
Cultures and Regions	511
Cultures in Action	516
Sorting	520
Resources	522
Creating Resource Files	522
ResGen	523
ResourceWriter	523
Using Resource Files	524
The System.Resources Namespace	527
Localization Example Using Visual Studio .NET	527
Outsourcing Translations	533
Changing the Culture Programmatically	534
Using Binary Resource Files	536
Using XML Resource Files	537
Automatic Fallback for Resources	539
Globalization and Localization with ASP.NET	539
A Custom Resource Reader	540
Creating a DatabaseResourceReader	541
Creating a DatabaseResourceSet	542
Creating a DatabaseResourceManager	543
Client Application for DatabaseResourceReader	544
Summary	544
Chapter 18: Deployment	545
Designing for Deployment	545
Deployment Options	546
Xcopy	546
Copy Project	546
Deployment Projects	546
Deployment Requirements	546
Simple Deployment	547
Xcopy	548
Xcopy and Web Applications	548
Copy Project	550
Installer Projects	551
What Is Windows Installer?	551
Creating Installers	552
Advanced Options	562
Summary	569

Part III: Windows Forms **571**

Chapter 19: Windows Forms **573**

Creating a Windows Form Application	574
Control Class	579
Size and Location	580
Appearance	580
User Interaction	580
Windows Functionality	582
Miscellaneous Functionality	582
Class Hierarchy	582
Standard Controls and Components	584
Forms	598
Form Class	599
Multiple Document Interface (MDI)	607
Custom Controls	610
Summary	622

Chapter 20: Graphics with GDI+ **623**

Understanding Drawing Principles	624
GDI and GDI+	624
Drawing Shapes	626
Painting Shapes Using OnPaint()	629
Using the Clipping Region	630
Measuring Coordinates and Areas	632
Point and PointF	632
Size and SizeF	634
Rectangle and RectangleF	635
Region	636
A Note about Debugging	637
Drawing Scrollable Windows	638
World, Page, and Device Coordinates	644
Colors	645
Red-Green-Blue (RGB) Values	645
The Named Colors	646
Graphics Display Modes and the Safety Palette	646
The Safety Palette	647
Pens and Brushes	648
Brushes	648
Pens	649

Drawing Shapes and Lines	650
Displaying Images	652
Issues When Manipulating Images	655
Drawing Text	655
Simple Text Example	656
Fonts and Font Families	657
Example: Enumerating Font Families	659
Editing a Text Document: The CapsEditor Sample	661
The Invalidate() Method	666
Calculating Item Sizes and Document Size	667
OnPaint()	668
Coordinate Transforms	670
Responding to User Input	671
Printing	675
Implementing Print and Print Preview	676
Summary	680
Part IV: Data	683
Chapter 21: Data Access with .NET	685
ADO.NET Overview	685
Namespaces	686
Shared Classes	686
Database-Specific Classes	687
Using Database Connections	688
Using Connections Efficiently	689
Transactions	692
Commands	693
Executing Commands	694
Calling Stored Procedures	698
Fast Data Access: The Data Reader	701
Managing Data and Relationships: The DataSet Class	704
Data Tables	704
Data Columns	705
Data Relationships	711
Data Constraints	713
XML Schemas	715
Generating Code with XSD	716
Populating a DataSet	721
Populating a DataSet Class with a Data Adapter	722
Populating a DataSet from XML	723

Contents

Persisting DataSet Changes	723
Updating with Data Adapters	724
Writing XML Output	726
Working with ADO.NET	728
Tiered Development	728
Key Generation with SQL Server	730
Naming Conventions	732
Summary	734
Chapter 22: Viewing .NET Data	735
The DataGrid Control	735
Displaying Tabular Data	735
Data Sources	738
DataGrid Class Hierarchy	746
Data Binding	750
Simple Binding	750
Data-Binding Objects	751
Visual Studio.NET and Data Access	757
Creating a Connection	758
Selecting Data	759
Generating a DataSet	762
Updating the Data Source	763
Building a Schema	764
Other Common Requirements	770
Summary	778
Chapter 23: Manipulating XML	781
XML Standards Support in .NET	782
Introducing the System.Xml Namespace	782
Using MSXML in .NET	783
Using System.Xml Classes	786
Reading and Writing Streamed XML	786
Using the XmlTextReader Class	787
Using the XmlValidatingReader Class	791
Using the XmlTextWriter Class	794
Using the DOM in .NET	795
Using the XmlDocument Class	797
Using XPath and XSLT in .NET	802
The System.Xml.XPath Namespace	803
The System.Xml.Xsl Namespace	807

XML and ADO.NET	812
Converting ADO.NET Data to XML	812
Converting XML to ADO.NET Data	820
Reading and Writing a DiffGram	822
Serializing Objects in XML	825
Serialization without Source Code Access	833
Summary	836
Chapter 24: Working with Active Directory	837
The Architecture of Active Directory	838
Features	838
Active Directory Concepts	839
Characteristics of Active Directory Data	843
Schema	843
Administration Tools for Active Directory	845
Active Directory Users and Computers	845
ADSI Edit	846
Active Directory Service Interfaces (ADSI)	847
Programming Active Directory	848
Classes in System.DirectoryServices	849
Binding	849
Getting Directory Entries	854
Object Collections	855
Cache	857
Creating New Objects	857
Updating Directory Entries	858
Accessing Native ADSI Objects	859
Searching in Active Directory	860
Searching for User Objects	864
User Interface	864
Get the Schema Naming Context	864
Get the Property Names of the User Class	866
Search for User Objects	867
Summary	869
Part V: Web Programming	871
Chapter 25: ASP.NET Pages	873
ASP.NET Introduction	874
State Management in ASP.NET	875

Contents

ASP.NET Web Forms	875
ASP.NET Server Controls	880
ADO.NET and Data Binding	892
Updating the Event-Booking Application	893
More on Data Binding	901
Application Configuration	906
Summary	907
<hr/> Chapter 26: Web Services	<hr/> 909
SOAP	910
WSDL	911
Web Services	913
Exposing Web Services	913
Consuming Web Services	916
Extending the Event-Booking Example	918
The Event-Booking Web Service	919
The Event-Booking Client	922
Exchanging Data Using SOAP Headers	924
Summary	929
<hr/> Chapter 27: User Controls and Custom Controls	<hr/> 931
User Controls	932
A Simple User Control	932
Custom Controls	939
Custom Control Project Configuration	940
Basic Custom Controls	944
Creating a Composite Custom Control	949
A Straw Poll Control	951
The Candidate Controls	953
The StrawPoll Control Builder	954
Straw Poll Style	955
The Straw Poll Control	956
Summary	962
Part VI: Interop	963
<hr/> Chapter 28: COM Interoperability	<hr/> 965
.NET and COM	966
Metadata	966
Freeing Memory	966

Interfaces	967
Method Binding	969
Data Types	969
Registration	969
Threading	969
Error Handling	971
Event Handling	972
Marshaling	972
Using a COM Component from a .NET Client	973
Creating a COM Component	973
Creating a Runtime Callable Wrapper	977
Threading Issues	980
Adding Connection Points	980
Using ActiveX Controls in Windows Forms	982
Using COM Objects from within ASP.NET	985
Using a .NET Component from a COM Client	985
COM Callable Wrapper	986
Creating a .NET Component	986
Creating a Type Library	987
COM Interop Attributes	989
COM Registration	992
Creating a COM Client	993
Adding Connection Points	995
Creating a Client with a Sink Object	996
Running Windows Forms Controls in Internet Explorer	997
Summary	998
Chapter 29: Enterprise Services	999
<hr/>	
Overview	999
History	999
Where to Use Enterprise Services?	1000
Contexts	1001
Automatic Transactions	1001
Distributed Transactions	1001
Object Pooling	1002
Role-based Security	1002
Queued Components	1002
Loosely Coupled Events	1002
Creating a Simple COM+ Application	1003
Class ServicedComponent	1003
Application Attributes	1003
Creating the Component	1004

Contents

Deployment	1005
Automatic Deployment	1005
Manual Deployment	1005
Component Services Admin Tool	1006
Client Application	1008
Transactions	1009
ACID Properties	1009
Transaction Attributes	1009
Transaction Results	1010
Sample Application	1011
Summary	1021
Part VII: Windows Base Services	1023
Chapter 30: File and Registry Operations	1025
<hr/>	
Managing the File System	1026
.NET Classes That Represent Files and Folders	1027
The Path Class	1029
Example: A File Browser	1030
Moving, Copying, and Deleting Files	1035
Example: FilePropertiesAndMovement	1035
Reading and Writing to Files	1039
Streams	1040
Reading and Writing to Binary Files	1042
Reading and Writing to Text Files	1047
Reading and Writing to the Registry	1054
The Registry	1055
The .NET Registry Classes	1057
Example: SelfPlacingWindow	1059
Summary	1066
Chapter 31: Accessing the Internet	1067
<hr/>	
The WebClient Class	1068
Downloading Files	1068
Basic Web Client Example	1068
Uploading Files	1070
WebRequest andWebResponse Classes	1070
Other WebRequest andWebResponse Features	1071
Displaying Output as an HTML Page	1074
The Web Request and Web Response Hierarchy	1075

Utility Classes	1077
URIs	1077
IP Addresses and DNS Names	1079
Lower-Level Protocols	1082
Lower-Level Classes	1083
Summary	1088
Chapter 32: Windows Services	1091
What Is a Windows Service?	1091
Windows Services Architecture	1093
Service Program	1093
Service Control Program	1095
Service Configuration Program	1095
System.ServiceProcess Namespace	1095
Creating a Windows Service	1096
A Class Library Using Sockets	1096
TcpClient Example	1100
Windows Service Project	1102
Threading and Services	1107
Service Installation	1107
Installation Program	1108
Monitoring and Controlling the Service	1113
MMC Computer Management	1114
net.exe	1114
sc.exe	1115
Visual Studio .NET Server Explorer	1116
ServiceController Class	1116
Troubleshooting	1122
Interactive Services	1123
Event Logging	1123
Performance Monitoring	1130
Power Events	1135
Summary	1135
At www.wrox.com	
Appendix A: Principles of Object-Oriented Programming	1137
Appendix B: C# for Visual Basic 6 Developers	1177
Appendix C: C# for Java Developers	1225
Appendix D: C# for C++ Developers	1253
Index	1307

Introduction

If we were to describe the C# language and its associated environment, the .NET Framework, as the most important new technology for developers for many years, we would not be exaggerating. .NET is designed to provide a new environment within which you can develop almost any application to run on Windows, while C# is a new programming language that has been designed specifically to work with .NET. Using C# you can, for example, write a dynamic Web page, an XML Web service, a component of a distributed application, a database access component, or a classic Windows desktop application. This book covers the .NET Framework 1.1, the second release of the framework, though most of this book also applies to .NET Framework 1.0. If you are coding using version 1.0, then you might have to make some changes, which we try to note throughout the book.

Don't be fooled by the .NET label. The NET bit in the name is there to emphasize Microsoft's belief that distributed applications, in which the processing is distributed between client and server, are the way forward, but C# is not just a language for writing Internet or network-aware applications. It provides a means for you to code up almost any type of software or component that you might need to write for the Windows platform. Between them, C# and .NET are set both to revolutionize the way that you write programs, and to make programming on Windows much easier than it has ever been.

That's quite a substantial claim, and it needs to be justified. After all, we all know how quickly computer technology changes. Every year Microsoft brings out new software, programming tools, or versions of Windows, with the claim that these will be hugely beneficial to developers. So what's different about .NET and C#?

The Significance of .NET and C#

In order to understand the significance of .NET, it is useful to remind ourselves of the nature of many of the Windows technologies that have appeared in the past ten years or so. Although they may look quite different on the surface, all of the Windows operating systems from Windows 3.1 (introduced in 1992) through Windows Server 2003 have the same familiar Windows API at their core. As we've progressed through new versions of Windows, huge numbers of new functions have been added to the API, but this has been a process of evolving and extending the API rather than replacing it.

The same can be said for many of the technologies and frameworks that we've used to develop software for Windows. For example, **COM (Component Object Model)** originated as **OLE (Object Linking and Embedding)**. At the time, it was, to a large extent, simply a means by which different types of Office documents could be linked, so that for example you could place a small Excel spreadsheet in your Word document. From that it evolved into COM, **DCOM (Distributed COM)**, and eventually COM+—a sophisticated technology that formed the basis of the way almost all components communicated, as well as implementing transactions, messaging services, and object pooling.

Microsoft chose this evolutionary approach to software for the obvious reason that it is concerned about backward compatibility. Over the years a huge base of third-party software has been written for Windows, and Windows wouldn't have enjoyed the success it has had if every time Microsoft introduced a new technology it broke the existing code base!

Introduction

While backward compatibility has been a crucial feature of Windows technologies and one of the strengths of the Windows platform, it does have a big disadvantage. Every time some technology evolves and adds new features, it ends up a bit more complicated than it was before.

It was clear that something had to change. Microsoft couldn't go on forever extending the same development tools and languages, always making them more and more complex in order to satisfy the conflicting demands of keeping up with the newest hardware and maintaining backward compatibility with what was around when Windows first became popular in the early 1990s. There comes a point where you have to start with a clean slate if you want a simple yet sophisticated set of languages, environments, and developer tools, which make it easy for developers to write state-of-the-art software.

This fresh start is what C# and .NET are all about. Roughly speaking, .NET is a new framework—a new API—for programming on the Windows platform. Along with the .NET Framework, C# is a new language that has been designed from scratch to work with .NET, as well as to take advantage of all the progress in developer environments and in our understanding of object-oriented programming principles that have taken place over the past 20 years.

Before we continue, we should make it clear that backward compatibility has not been lost in the process. Existing programs will continue to work, and .NET was designed with the ability to work with existing software. Communication between software components on Windows presently almost entirely takes place using COM. Taking account of this, .NET does have the ability to provide wrappers around existing COM components so that .NET components can talk to them.

It is true that you don't need to learn C# in order to write code for .NET. Microsoft has extended C++, provided another new language called J#, and made substantial changes to Visual Basic to turn it into the more powerful language Visual Basic .NET, in order to allow code written in either of these languages to target the .NET environment. These other languages, however, are hampered by the legacy of having evolved over the years rather than having been written from the start with today's technology in mind.

This book will equip you to program in C#, while at the same time provide the necessary background in how the .NET architecture works. We will not only cover the fundamentals of the C# language but also go on to give examples of applications that use a variety of related technologies, including database access, dynamic Web pages, advanced graphics, and directory access. The only requirement is that you be familiar with at least one other high-level language used on Windows—either C++, Visual Basic, or J++.

Advantages of .NET

We've talked in general terms about how great .NET is, but we haven't said much about how it helps to make your life as a developer easier. In this section, we'll discuss some of the improved features of .NET in brief.

- ❑ **Object-Oriented Programming**—both the .NET Framework and C# are entirely based on object-oriented principles right from the start.
- ❑ **Good Design**—a base class library, which is designed from the ground up in a highly intuitive way.

- ❑ **Language Independence**—with .NET, all of the languages Visual Basic .NET, C#, J#, and managed C++ compile to a common **Intermediate Language**. This means that languages are interoperable in a way that has not been seen before.
- ❑ **Better Support for Dynamic Web Pages**—while ASP offered a lot of flexibility, it was also inefficient because of its use of interpreted scripting languages, and the lack of object-oriented design often resulted in messy ASP code. .NET offers an integrated support for Web pages, using a new technology—ASP.NET. With ASP.NET, code in your pages is compiled, and may be written in a .NET-aware high-level language such as C#, J#, or Visual Basic .NET.
- ❑ **Efficient Data Access**—a set of .NET components, collectively known as ADO.NET, provides efficient access to relational databases and a variety of data sources. Components are also available to allow access to the file system, and to directories. In particular, XML support is built into .NET, allowing you to manipulate data, which may be imported from or exported to non-Windows platforms.
- ❑ **Code Sharing**—.NET has completely revamped the way that code is shared between applications, introducing the concept of the **assembly**, which replaces the traditional DLL. Assemblies have formal facilities for versioning, and different versions of assemblies can exist side by side.
- ❑ **Improved Security**—each assembly can also contain built-in security information that can indicate precisely who or what category of user or process is allowed to call which methods on which classes. This gives you a very fine degree of control over how the assemblies that you deploy can be used.
- ❑ **Zero Impact Installation**—there are two types of assembly: shared and private. Shared assemblies are common libraries available to all software, while private assemblies are intended only for use with particular software. A private assembly is entirely self-contained, so the process of installing it is simple. There are no registry entries; the appropriate files are simply placed in the appropriate folder in the file system.
- ❑ **Support for Web Services**—.NET has fully integrated support for developing Web services as easily as you'd develop any other type of application.
- ❑ **Visual Studio .NET 2003**—.NET comes with a developer environment, Visual Studio .NET, which can cope equally well with C++, C#, J#, and Visual Basic .NET, as well as with ASP.NET code. Visual Studio .NET integrates all the best features of the respective language-specific environments of Visual Studio 6.
- ❑ **C#**—C# is a new object-oriented language intended for use with .NET.

We will be looking more closely at the benefits of the .NET architecture in Chapter 1.

What's New in the .NET Framework 1.1

The first version of the .NET Framework (1.0) was released in 2002 to much enthusiasm. The latest version, the .NET Framework 1.1, was introduced in 2003 and is considered a minor release of the framework. Even though this is considered a minor release of the framework, there are some pretty outstanding new changes and additions to this new version and it definitely deserves some attention.

With all the changes made to version 1.1 of the framework, Microsoft tried to ensure that there were minimal breaking changes to code developed in using version 1.0. Even though the effort was there,

Introduction

there are some breaking changes between the versions. A lot of these breaking changes were made in order to improve security. You will find a comprehensive list of breaking changes on Microsoft's GotDotNet Web site at <http://www.gotdotnet.com>.

Make sure that you create a staging server to completely test the upgrade of your applications to the .NET Framework 1.1 as opposed to just upgrading a live application.

The following details some of the changes that are new to the .NET Framework 1.1 as well as new additions to Visual Studio .NET 2003—the development environment for the .NET Framework 1.1.

Mobility

When using the .NET Framework 1.0 and Visual Studio .NET 2002, to be able to build mobile applications you had to go out and download the Microsoft Mobile Internet Toolkit (MMIT). Now, with the .NET Framework 1.1 and Visual Studio .NET 2003, this is built right in and therefore no separate download is required.

This is all quite evident when you create a new project using Visual Studio .NET 2003. For instance, when you look at the list of available C# project types you can create, you will find ASP.NET Mobile Web Application and Smart Device Application. You would use the ASP.NET Mobile Web Application project type to build Web-based mobile applications (as the name describes). Building a Smart Device Application allows you to create applications for the Pocket PC or any other Windows CE device. The thick-client applications built for a Windows CE device utilize the Compact Framework, a trimmed-down version of the .NET Framework.

Opening one of these mobile project types, you will then be presented with a list of available mobile server controls in the Visual Studio .NET Toolbox that you can then use to build your applications.

New Data Providers

Another big area of change in the framework is to ADO.NET. ADO.NET, the .NET way of accessing and working with data, now has two new data providers—one for ODBC and another for Oracle.

An ODBC data provider was available when working with the .NET Framework 1.0, but this required a separate download. Also, once downloaded, the namespace for this data provider was `Microsoft.Data.Odbc`.

With the .NET Framework 1.1, the ODBC data provider is built right in, and no separate download is required. You will now be able to work with ODBC data sources through the `System.Data.Odbc` namespace. This also gives you access to ODBC data connection, data adapter, and data reader objects.

The other new data provider is for working with Oracle databases. This database is quite popular in the enterprise space, and the lack of an Oracle data provider often times was a big barrier for .NET to enter this space. To work with this new data provider, you will need to make a reference to the `System.Data.OracleClient` namespace in your project.

A New Language: Visual J#

When you install Visual Studio .NET 2003, you will notice that a new language is available to you for building .NET applications—J#. Prior to this, when using Visual Studio .NET 2002, you were forced to install the language as a separate download.

Visual J#, or simply J# (pronounced *J-sharp*), is the next version of the Visual J++ language. You will find that it is very similar to the Java language. The hope with this language is that Java developers will find it an easy transition to .NET. A J# developer will use the .NET class libraries in place of the Java runtime libraries.

J# developers will have access to much of the same capabilities as a C# developer on the .NET platform. Using J#, it is just as possible to build .NET classes, Windows Forms applications, ASP.NET Web applications, and XML Web services. In addition, you can use J# in the same cross-language ways that you can use other .NET-compliant languages. For instance, you can build a J# class and use this class in your C# application or vice versa.

Also like the other languages, there is a built-in compiler for J# now in the .NET Framework. To find any of the compilers, you will see them at `C:\Windows\Microsoft.NET\Framework\v1.1.xxxx`. The C# compiler is `csc.exe`, the Visual Basic .NET compiler is `vbc.exe`, and the J# compiler is `vjc.exe`.

Side-by-Side Execution

Side-by-side execution is the ability to run multiple versions of an application on the same server where different application versions target different runtime versions. This was always promised to us as developers, but it was always hard to visualize as only one version of the framework was available. With the release of a second version of the framework (.NET Framework 1.1), we can actually see that it is possible to have this capability. Therefore, you can build new versions of your .NET applications that target this latest .NET Framework version release, but at the same time you can allow the older versions of your application that target the .NET Framework 1.0 to continue to work just as they always have.

Support for Internet Protocol Version 6 (IPv6)

Presently, much of the Internet runs using IP version 4, also referred to as IPv4. IPv4 gives us IP addresses such as 255.255.255.255. The .NET Framework 1.1 now supports IPv6, which was created in 1995 to address many of the problems that the world was facing with IPv4. Most of the problems deal with the fact that by the world's continual use of IPv4, we are rapidly running out of available IP addresses.

IPv6 is supported in the .NET Framework 1.1 through the `System.Net` namespace as well as in ASP.NET and XML Web services.

Visual Studio .NET 2003 Enhancements

Along with the upgrade to the .NET Framework, Visual Studio .NET itself has also undergone an upgrade. You will notice that there are some new graphics on the Start Page available and that things on this page are organized a little differently. Besides that, the biggest thing to notice with this new IDE is that once installed, it does not simply upgrade Visual Studio .NET 2002 to Visual Studio .NET 2003.

Introduction

Instead, it installs a completely new version of the IDE, and if you already have VS.NET 2002 on your machine, then you will have two complete VS.NET IDEs on your box. The reason for this is so that if you want to build and work with applications that target the .NET Framework version 1.0, then you will use VS.NET 2002, and if you want to build and work with applications that target the .NET Framework version 1.1 then you will use VS.NET 2003.

You should also be aware that when you open a project that was built using VS.NET 2002, you will be asked if you want to upgrade the project to be a VS.NET 2003 project. Doing this will then cause the project to be re-targeted at the .NET Framework 1.1. Be careful about doing this as it is an irreversible process.

Besides these big changes, you will find that VS.NET 2003 is a better IDE with smarter Intellisense and code completion. This version of the IDE is the IDE that is used throughout the examples of this book.

Where C# Fits In

In one sense, C# can be seen as being the same thing to programming languages as .NET is to the Windows environment. Just as Microsoft has been adding more and more features to Windows and the Windows API over the past decade, Visual Basic and C++ have undergone expansion. Although Visual Basic and C++ have ended up as hugely powerful languages as a result of this, both languages also suffer from problems due to the legacies of how they have evolved.

In the case of Visual Basic 6 and earlier, the main strength of the language was the fact that it was simple to understand and didn't make many programming tasks easy, largely hiding the details of the Windows API and the COM component infrastructure from the developer. The downside to this was that Visual Basic was never truly object-oriented, so that large applications quickly become disorganized and hard to maintain. As well as this, because Visual Basic's syntax was inherited from early versions of BASIC (which, in turn, was designed to be intuitively simple for beginning programmers to understand, rather than to write large commercial applications), it didn't really lend itself to well-structured or object-oriented programs.

C++, on the other hand, has its roots in the ANSI C++ language definition. It isn't completely ANSI-compliant for the simple reason that Microsoft first wrote its C++ compiler before the ANSI definition had become official, but it comes close. Unfortunately, this has led to two problems. First, ANSI C++ has its roots in a decade-old state of technology, and this shows up in a lack of support for modern concepts (such as Unicode strings and generating XML documentation), and in some archaic syntax structures designed for the compilers of yesteryear (such as the separation of declaration from definition of member functions). Second, Microsoft has been simultaneously trying to evolve C++ into a language that is designed for high-performance tasks on Windows, and in order to achieve that they've been forced to add a huge number of Microsoft-specific keywords as well as various libraries to the language. The result is that on Windows, the language has become a complete mess. Just ask C++ developers how many definitions for a string they can think of: `char*`, `LPTSTR`, `string`, `CString` (MFC version), `CString` (WTL version), `wchar_t*`, `OLECHAR*`, and so on.

Now enter .NET—a completely new environment that is going to involve new extensions to both languages. Microsoft has gotten around this by adding yet more Microsoft-specific keywords to C++, and by completely revamping Visual Basic into Visual Basic .NET, a language that retains some of the basic VB syntax but that is so different in design that we can consider it to be, for all practical purposes, a new language.