



Professional **C# 5.0** **and .NET 4.5.1**

Christian Nagel, Jay Glynn, Morgan Skinner

PROFESSIONAL **C# 5.0 and .NET 4.5.1**

CURRENT AUTHOR TEAM

Christian Nagel
Jay Glynn
Morgan Skinner

AUTHORS ON PREVIOUS EDITIONS

Bill Evjen
Karli Watson



Professional C# 5.0 and .NET 4.5.1

Published by
John Wiley & Sons, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2014 by John Wiley & Sons, Inc., Indianapolis, Indiana

Published simultaneously in Canada

978-1-118-83303-2
978-1-118-83294-3 (ebk)
978-1-118-83298-1 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

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 Sections 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, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

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 Web site 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 Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites 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 (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley publishes in a variety of print and electronic formats and by print-on-demand. Some material included with standard print versions of this book may not be included in e-books or in print-on-demand. If this book refers to media such as a CD or DVD that is not included in the version you purchased, you may download this material at <http://booksupport.wiley.com>. For more information about Wiley products, visit www.wiley.com.

Library of Congress Control Number: 2013958290

Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

*To my family – Angela, Stephanie, and Matthias –
I love you all!*

—CHRISTIAN NAGEL

*This work is dedicated to my wife and son.
They are my world.*

—JAY GLYNN

*Love is as strong as death;
Many waters cannot quench love,
Neither can the floods drown it.*

—MORGAN SKINNER

CREDITS

ACQUISITIONS EDITOR

Mary James

PROJECT EDITOR

Charlotte Kughen

TECHNICAL EDITORS

Don Reamey

George Evjen

PRODUCTION EDITOR

Christine Mugnolo

EDITORIAL MANAGER

Mary Beth Wakefield

FREELANCER EDITORIAL MANAGER

Rosemarie Graham

ASSOCIATE DIRECTOR OF MARKETING

David Mayhew

MARKETING MANAGER

Ashley Zurcher

BUSINESS MANAGER

Amy Knies

VICE PRESIDENT AND EXECUTIVE GROUP**PUBLISHER**

Richard Swadley

ASSOCIATE PUBLISHER

Jim Minatel

PROJECT COORDINATOR, COVER

Katie Crocker

PROOFREADER

Sarah Kaikini, Word One, New York

INDEXER

Johnna VanHoose Dinse

COVER DESIGNER

Wiley

COVER IMAGE

© Henrik5000/istockphoto.com

ABOUT THE AUTHORS

CHRISTIAN NAGEL is a Microsoft Regional Director and Microsoft MVP, an associate of thinktecture, and founder of CN innovation. A software architect and developer, he offers training and consulting on how to develop solutions using the Microsoft platform. He draws on more than 25 years of software development experience. Christian started his computing career with PDP 11 and VAX/VMS systems, covering a variety of languages and platforms. Since 2000, when .NET was just a technology preview, he has been working with various .NET technologies to build .NET solutions. Currently, he mainly coaches the development of Windows Store apps accessing Windows Azure services. With his profound knowledge of Microsoft technologies, he has written numerous books, and is certified as a Microsoft Certified Trainer (MCT) and Solutions Developer (MCSD). Christian speaks at international conferences such as TechEd, Basta!, and TechDays, and he founded INETA Europe to support .NET user groups. You can contact Christian via his website www.cninnovation.com, read his blog at blogs.thinktecture.com/cnagel, and follow his tweets at @christiannagel.

JAY GLYNN started writing software more than 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, Java, and of course C#. He currently works for VGT as a software engineer writing server-based software.

MORGAN SKINNER began his computing career at a young age on the Sinclair ZX80 at school, where he was underwhelmed by some code a teacher had written and so began programming in assembly language. Since then he has used a wide variety 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's now an independent consultant.

ABOUT THE TECHNICAL EDITORS

DON REAMEY is an architect/principal engineer for TIBCO Software working on TIBCO Spotfire business intelligence analytics software. Prior to TIBCO Don spent 12 years with Microsoft as a software development engineer working on SharePoint, SharePoint Online and InfoPath Forms Service. Don has also spent 10 years writing software in the financial service industry for capital markets.

GEORGE EVJEN is the director of development for ArchitectNow, a St. Louis-based consulting company specializing in custom client application architecture, design, and development, with clients ranging from small technology start-ups to global enterprises. Prior to his involvement in the software industry, George spent more than a dozen years coaching men's basketball at all levels of the collegiate ranks. As a motivational leader with an infectious positive outlook in nearly all situations, he is the ideal person to take the lead directly for many of ArchitectNow's largest projects and clients. Not only does he work as a lead developer, but he also manages most of the coordination between ArchitectNow and the company's external contractors and resources.

George has extensive experience and expertise in all of Microsoft's web-based and XAML-based technologies, as well as the newest web frameworks available. His specialties include enterprise-level WPF, Silverlight, and Windows 8 projects, as well as ASP.NET MVC business application development. He speaks to groups and at conferences around the region on topics of motivational leadership, project management, and organization. You can find additional information on George and ArchitectNow's capabilities at <http://www.architectnow.net>.

ACKNOWLEDGMENTS

I WOULD LIKE TO THANK Charlotte Kughen for making this text more readable; Mary James; and Jim Minatel; and everyone else at Wiley who helped to get another edition of this great book published. I would also like to thank my wife and children for supporting my writing. You're my inspiration.

—CHRISTIAN NAGEL

I WANT TO THANK my wife and son for putting up with the time and frustrations of working on a project like this. I also want to thank all the dedicated people at Wiley for getting this book out the door.

—JAY GLYNN

CONTENTS

INTRODUCTION

xxiii

PART I: THE C# LANGUAGE

CHAPTER 1: .NET ARCHITECTURE	3
The Relationship of C# to .NET	3
The Common Language Runtime	4
A Closer Look at Intermediate Language	7
Assemblies	14
.NET Framework Classes	16
Namespaces	17
Creating .NET Applications Using C#	18
The Role of C# in the .NET Enterprise Architecture	21
Summary	22
CHAPTER 2: CORE C#	23
Fundamental C#	24
Your First C# Program	24
Variables	27
Predefined Data Types	31
Flow Control	37
Enumerations	43
Namespaces	45
The Main() Method	47
More on Compiling C# Files	49
Console I/O	50
Using Comments	52
The C# Preprocessor Directives	54
C# Programming Guidelines	57
Summary	63
CHAPTER 3: OBJECTS AND TYPES	65
Creating and Using Classes	65
Classes and Structs	66
Classes	66
Anonymous Types	79

Structs	80
Weak References	82
Partial Classes	83
Static Classes	85
The Object Class	85
Extension Methods	87
Summary	88
CHAPTER 4: INHERITANCE	89
Inheritance	89
Types of Inheritance	89
Implementation Inheritance	90
Modifiers	99
Interfaces	100
Summary	105
CHAPTER 5: GENERICS	107
Generics Overview	107
Creating Generic Classes	110
Generics Features	114
Generic Interfaces	118
Generic Structs	122
Generic Methods	124
Summary	128
CHAPTER 6: ARRAYS AND TUPLES	129
Multiple Objects of the Same and Different Types	129
Simple Arrays	130
Multidimensional Arrays	132
Jagged Arrays	133
Array Class	134
Arrays as Parameters	139
Enumerations	140
Tuples	146
Structural Comparison	147
Summary	149
CHAPTER 7: OPERATORS AND CASTS	151
Operators and Casts	151
Operators	151

Type Safety	157
Comparing Objects for Equality	162
Operator Overloading	163
User-Defined Casts	172
Summary	181
CHAPTER 8: DELEGATES, LAMBDA, AND EVENTS	183
Referencing Methods	183
Delegates	184
Lambda Expressions	198
Events	201
Summary	208
CHAPTER 9: STRINGS AND REGULAR EXPRESSIONS	209
Examining System.String	210
Regular Expressions	221
Summary	228
CHAPTER 10: COLLECTIONS	229
Overview	230
Collection Interfaces and Types	230
Lists	231
Queues	241
Stacks	245
Linked Lists	246
Sorted List	251
Dictionaries	252
Sets	259
Observable Collections	260
Bit Arrays	262
Immutable Collections	266
Concurrent Collections	268
Performance	275
Summary	277
CHAPTER 11: LANGUAGE INTEGRATED QUERY	279
LINQ Overview	279
Standard Query Operators	287
Parallel LINQ	305
Expression Trees	307

LINQ Providers	310
Summary	310
CHAPTER 12: DYNAMIC LANGUAGE EXTENSIONS	313
Dynamic Language Runtime	313
The Dynamic Type	314
Hosting the DLR ScriptRuntime	318
DynamicObject and ExpandoObject	321
Summary	324
CHAPTER 13: ASYNCHRONOUS PROGRAMMING	325
Why Asynchronous Programming Is Important	325
Asynchronous Patterns	326
Foundation of Asynchronous Programming	338
Error Handling	341
Cancellation	344
Summary	346
CHAPTER 14: MEMORY MANAGEMENT AND POINTERS	347
Memory Management	347
Memory Management Under the Hood	348
Freeing Unmanaged Resources	353
Unsafe Code	358
Summary	372
CHAPTER 15: REFLECTION	373
Manipulating and Inspecting Code at Runtime	373
Custom Attributes	374
Using Reflection	380
Summary	389
CHAPTER 16: ERRORS AND EXCEPTIONS	391
Introduction	391
Exception Classes	392
Catching Exceptions	393
User-Defined Exception Classes	402
Caller Information	409
Summary	411

PART II: VISUAL STUDIO**CHAPTER 17: VISUAL STUDIO 2013 415**

Working with Visual Studio 2013	415
Creating a Project	420
Exploring and Coding a Project	425
Building a Project	437
Debugging Your Code	441
Refactoring Tools	447
Architecture Tools	448
Analyzing Applications	451
Unit Tests	457
Windows Store Apps, WCF, WF, and More	463
Summary	467

CHAPTER 18: DEPLOYMENT 469

Deployment as Part of the Application Life Cycle	469
Planning for Deployment	470
Traditional Deployment	471
ClickOnce	473
Web Deployment	479
Windows Store Apps	481
Summary	486

PART III: FOUNDATION**CHAPTER 19: ASSEMBLIES 489**

What are Assemblies?	489
Application Domains	499
Shared Assemblies	503
Configuring .NET Applications	510
Versioning	513
Sharing Assemblies Between Different Technologies	517
Summary	520

CHAPTER 20: DIAGNOSTICS 521

Diagnostics Overview	521
Code Contracts	522
Tracing	528

Event Logging	540
Performance Monitoring	548
Summary	554
CHAPTER 21: TASKS, THREADS, AND SYNCHRONIZATION	555
Overview	556
Parallel Class	557
Tasks	561
Cancellation Framework	566
Thread Pools	569
The Thread Class	570
Threading Issues	574
Synchronization	579
Timers	597
Data Flow	598
Summary	602
CHAPTER 22: SECURITY	605
Introduction	605
Authentication and Authorization	606
Encryption	614
Access Control to Resources	621
Code Access Security	623
Distributing Code Using Certificates	629
Summary	630
CHAPTER 23: INTEROP	631
.NET and COM	631
Using a COM Component from a .NET Client	638
Using a .NET Component from a COM Client	649
Platform Invoke	659
Summary	663
CHAPTER 24: MANIPULATING FILES AND THE REGISTRY	665
File and the Registry	665
Managing the File System	666
Moving, Copying, and Deleting Files	674
Reading and Writing to Files	677
Mapped Memory Files	692
Reading Drive Information	693

File Security	695
Reading and Writing to the Registry	699
Reading and Writing to Isolated Storage	704
Summary	707
CHAPTER 25: TRANSACTIONS	709
Introduction	709
Overview	710
Database and Entity Classes	712
Traditional Transactions	713
System.Transactions	716
Dependent Transactions	721
Isolation Level	729
Custom Resource Managers	731
File System Transactions	737
Summary	740
CHAPTER 26: NETWORKING	741
Networking	741
The HttpClient Class	742
Displaying Output as an HTML Page	746
Utility Classes	756
Lower-Level Protocols	759
Summary	771
CHAPTER 27: WINDOWS SERVICES	773
What Is a Windows Service?	773
Windows Services Architecture	775
Creating a Windows Service Program	777
Monitoring and Controlling Windows Services	793
Troubleshooting and Event Logging	802
Summary	803
CHAPTER 28: LOCALIZATION	805
Global Markets	805
Namespace System.Globalization	806
Resources	817
Windows Forms Localization Using Visual Studio	823
Localization with ASP.NET Web Forms	830
Localization with WPF	832

A Custom Resource Reader	837
Creating Custom Cultures	840
Localization with Windows Store Apps	842
Summary	845
CHAPTER 29: CORE XAML	847
Uses of XAML	847
XAML Foundation	848
Dependency Properties	853
Bubbling and Tunneling Events	856
Attached Properties	859
Markup Extensions	861
Reading and Writing XAML	863
Summary	864
CHAPTER 30: MANAGED EXTENSIBILITY FRAMEWORK	865
Introduction	865
MEF Architecture	866
Defining Contracts	873
Exporting Parts	875
Importing Parts	884
Containers and Export Providers	889
Catalogs	892
Summary	893
CHAPTER 31: WINDOWS RUNTIME	895
Overview	895
Windows Runtime Components	902
Windows Store Apps	905
The Life Cycle of Applications	907
Application Settings	913
Summary	916
PART IV: DATA	
CHAPTER 32: CORE ADO.NET	919
ADO.NET Overview	919
Using Database Connections	922
Commands	927

Fast Data Access: The Data Reader	934
Asynchronous Data Access: Using Task and Await	936
Managing Data and Relationships: The DataSet Class	938
XML Schemas: Generating Code with XSD	948
Populating a DataSet	953
Persisting DataSet Changes	955
Working with ADO.NET	958
Summary	963
CHAPTER 33: ADO.NET ENTITY FRAMEWORK	965
Programming with the Entity Framework	965
Entity Framework Mapping	967
Entities	972
Data Context	973
Relationships	975
Querying Data	980
Writing Data to the Database	982
Using the Code First Programming Model	987
Summary	995
CHAPTER 34: MANIPULATING XML	997
XML	997
XML Standards Support in .NET	998
Introducing the System.Xml Namespace	998
Using System.Xml Classes	999
Reading and Writing Streamed XML	1000
Using the DOM in .NET	1007
Using XPathNavigators	1011
XML and ADO.NET	1020
Serializing Objects in XML	1027
LINQ to XML and .NET	1036
Working with Different XML Objects	1036
Using LINQ to Query XML Documents	1042
More Query Techniques for XML Documents	1045
Summary	1048
PART V: PRESENTATION	
CHAPTER 35: CORE WPF	1051
Understanding WPF	1052

Shapes	1055
Geometry	1056
Transformation	1058
Brushes	1060
Controls	1063
Layout	1068
Styles and Resources	1071
Triggers	1077
Templates	1080
Animations	1089
Visual State Manager	1095
3-D	1098
Summary	1102
CHAPTER 36: BUSINESS APPLICATIONS WITH WPF	1103
Introduction	1103
Menu and Ribbon Controls	1104
Commanding	1107
Data Binding	1109
TreeView	1139
DataGrid	1143
Summary	1154
CHAPTER 37: CREATING DOCUMENTS WITH WPF	1155
Introduction	1155
Text Elements	1156
Flow Documents	1164
Fixed Documents	1168
XPS Documents	1171
Printing	1173
Summary	1175
CHAPTER 38: WINDOWS STORE APPS: USER INTERFACE	1177
Overview	1177
Microsoft Modern Design	1178
Sample Application Core Functionality	1180
App Bars	1187
Launching and Navigation	1188
Layout Changes	1190
Storage	1195

Pickers	1201
Live Tiles	1202
Summary	1204
CHAPTER 39: WINDOWS STORE APPS: CONTRACTS AND DEVICES	1205
Overview	1205
Searching	1206
Sharing Contract	1208
Camera	1212
Geolocation	1213
Sensors	1216
Summary	1221
CHAPTER 40: CORE ASP.NET	1223
.NET Frameworks for Web Applications	1223
Web Technologies	1225
Hosting and Configuration	1226
Handlers and Modules	1229
Global Application Class	1233
Request and Response	1234
State Management	1236
ASP.NET Identity System	1247
Summary	1251
CHAPTER 41: ASP.NET WEB FORMS	1253
Overview	1253
ASPX Page Model	1254
Master Pages	1263
Navigation	1267
Validating User Input	1268
Accessing Data	1271
Security	1280
Ajax	1283
Summary	1296
CHAPTER 42: ASP.NET MVC	1297
ASP.NET MVC Overview	1297
Defining Routes	1299

Creating Controllers	1300
Creating Views	1304
Submitting Data from the Client	1314
HTML Helpers	1318
Creating a Data-Driven Application	1323
Action Filters	1331
Authentication and Authorization	1332
Summary	1336

PART VI: COMMUNICATION

CHAPTER 43: WINDOWS COMMUNICATION FOUNDATION 1339

WCF Overview	1339
Creating a Simple Service and Client	1342
Contracts	1354
Service Behaviors	1358
Binding	1362
Hosting	1368
Clients	1370
Duplex Communication	1372
Routing	1374
Summary	1379

CHAPTER 44: ASP.NET WEB API 1381

Overview	1381
Creating Services	1382
Creating a .NET Client	1385
Web API Routing and Actions	1388
Using OData	1391
Security with the Web API	1400
Self-Hosting	1405
Summary	1406

CHAPTER 45: WINDOWS WORKFLOW FOUNDATION 1407

A Workflow Overview	1407
Hello World	1408
Activities	1409
Custom Activities	1413
Workflows	1419
Summary	1432

CHAPTER 46: PEER-TO-PEER NETWORKING	1433
Peer-to-Peer Networking Overview	1433
Peer Name Resolution Protocol (PNRP)	1437
Building P2P Applications	1439
Summary	1445
CHAPTER 47: MESSAGE QUEUING	1447
Overview	1448
Message Queuing Products	1450
Message Queuing Architecture	1451
Message Queuing Administrative Tools	1452
Programming Message Queuing	1453
Course Order Application	1460
Receiving Results	1470
Transactional Queues	1471
Message Queuing with WCF	1472
Message Queue Installation	1478
Summary	1478
INDEX	1479

INTRODUCTION

IF YOU WERE TO DESCRIBE THE C# LANGUAGE and its associated environment, the .NET Framework, as the most significant technology for developers available, you would not be exaggerating. .NET is designed to provide an environment within which you can develop almost any application to run on Windows, whereas C# is a programming language designed specifically to work with the .NET Framework. By using C#, you can, for example, write a dynamic web page, a Windows Presentation Foundation application, an XML web service, a component of a distributed application, a database access component, a classic Windows desktop application, or even a new smart client application that enables online and offline capabilities. This book covers the .NET Framework 4.5.1. If you code using any of the prior versions, there may be sections of the book that will not work for you. This book notifies you of items that are new and specific to the .NET Framework 4.5 and 4.5.1.

Don't be fooled by the .NET label in the Framework's name and think that this is a purely Internet-focused framework. 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. You must also understand that C# is not just a language for writing Internet or network-aware applications. It provides a means for you to code almost any type of software or component that you need to write for the Windows platform. Between them, C# and .NET have revolutionized the way that developers write their programs and have made programming on Windows much easier than it has ever been before.

So what's the big deal about .NET and C#?

THE SIGNIFICANCE OF .NET AND C#

To understand the significance of .NET, you must consider the nature of many of the Windows technologies that have appeared in the past 20 years. Although they may look quite different on the surface, all the Windows operating systems from Windows NT 3.1 (introduced in 1993) through Windows 8.1 and Windows Server 2012 R2 have the same familiar Windows API for Windows desktop and server applications at their core. Progressing through new versions of Windows, huge numbers of new functions have been added to the API, but this has been a process to evolve and extend the API rather than replace it.

With Windows 8, the main API of the operating system gets a replacement with Windows Runtime. However, this runtime is still partly based on the familiar Windows API.

The same can be said for many of the technologies and frameworks used to develop software for Windows. For example, *Component Object Model* (COM) originated as *Object Linking and Embedding* (OLE). Originally, it was largely a means by which different types of Office documents could be linked so that you could place a small Excel spreadsheet in your Word document, for example. From that it evolved into COM, *Distributed COM* (DCOM), 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 with backward compatibility. Over the years, a huge base of third-party software has been written for Windows, and Windows would not have enjoyed the success it has had if every time Microsoft introduced a new technology it broke the existing code base!

Although 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 could not go on forever extending the same development tools and languages, always making them more and more complex 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 in which you must start with a clean slate if you want a simple yet sophisticated set of languages, environments, and developer tools, which makes it easy for developers to write state-of-the-art software.

This fresh start is what C# and .NET were all about in the first incarnation. Roughly speaking, .NET is a framework — an API — for programming on the Windows platform. Along with the .NET Framework, C# is a 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 your understanding of object-oriented programming principles that have taken place over the past 25 years.

Before continuing, you must understand that backward compatibility has not been lost in the process. Existing programs continue to work, and .NET was designed with the capability to work with existing software. Presently, communication between software components on Windows takes place almost entirely using COM. Taking this into account, the .NET Framework does have the capability 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# to write code for .NET. Microsoft has extended C++ and made substantial changes to Visual Basic to turn it into a more powerful language to enable 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 can equip you to program in C#, while at the same time provides the necessary background in how the .NET architecture works. You not only cover the fundamentals of the C# language, but also see examples of applications that use a variety of related technologies, including database access, dynamic web pages, advanced graphics, and directory access.

While the Windows API evolved and was extended since the early days of Windows NT in 1993, the .NET Framework offered a major change on how programs are written since the year 2002; now, starting with the year 2012, we have the days of the next big change. Do such changes happen every 10 years? Windows 8 offers a new API: the Windows Runtime (WinRT) for Windows Store apps. This runtime is a native API (like the Windows API) that is not build with the .NET runtime as its core, but offers great new features that are based on ideas of .NET. Windows 8 includes the first release of this API available for modern-style apps. Although this is not based on .NET, you still can use a subset of .NET with Windows Store apps, and write the apps with C#. This new runtime evolves, and with Windows 8.1 version 2 is included. This book will give you a start in writing Windows Store apps with C# and WinRT.

ADVANTAGES OF .NET

So far, you've read in general terms about how great .NET is, but it can help to make your life as a developer easier. This section briefly identifies some of the features of .NET:

- **Object-oriented programming** — Both the .NET Framework and C# are entirely based on object-oriented principles 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 the languages — Visual Basic, C#, 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** — Though Classic 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 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# or Visual Basic 2013. .NET now takes it even further with outstanding support for the latest web technologies such as Ajax and jQuery.
- **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 enable access to the file system and to directories. In particular, XML support is built into .NET, enabling 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 fine degree of control over how the assemblies that you deploy can be used.
- **Zero-impact installation** — There are two types of assemblies: shared and private. Shared assemblies are common libraries available to all software, whereas private assemblies are intended only for use with particular software. A private assembly is entirely self-contained, so the process to install 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 would develop any other type of application.
- **Visual Studio 2013** — .NET comes with a developer environment, Visual Studio 2013, which can cope equally well with C++, C#, and Visual Basic 2013, as well as with ASP.NET or XML code. Visual Studio 2013 integrates all the best features of the respective language-specific environments of all the previous versions of this amazing IDE.
- **C#** — C# is a powerful and popular object-oriented language intended for use with .NET.

You look more closely at the benefits of the .NET architecture in Chapter 1, “.NET Architecture.”

WHAT'S NEW IN THE .NET FRAMEWORK 4.5 AND .NET 4.5.1

The first version of the .NET Framework (1.0) was released in 2002 to much enthusiasm. The .NET Framework 2.0 was introduced in 2005 and was considered a major release of the Framework. The major new feature of 2.0 was generics support in C# and the runtime (IL code changed for generics), and new classes and interfaces. .NET 3.0 was based on the 2.0 runtime and introduced a new way to create UIs (WPF with XAML and vector-based graphics instead of pixel-based), and a new communication technology (WCF). .NET 3.5 together with C# 3.0 introduced LINQ, one query syntax that can be used for all data sources. .NET 4.0 was another major release of the product that also brought a new version of the runtime (4.0) and a new version of C# (4.0) to offer dynamic language integration and a huge new library for parallel programming. The .NET Framework 4.5 is based on an updated version of the 4.0 runtime with many outstanding new features. The .NET Framework 4.5.1 gives some small increments. However, with more

and more libraries that are part of the .NET Framework being distributed as NuGet packages, more and more features are delivered out of band to the .NET Framework. For example, the Entity Framework, ASP .NET Web API, and other .NET libraries got huge improvements.

With each release of the Framework, Microsoft has always tried to ensure that there were minimal breaking changes to code developed. Thus far, Microsoft has been successful at this goal.

The following section details some of the changes that are new to C# 5.0 and the .NET Framework 4.5.1.

Asynchronous Programming

Blocking the UI is unfriendly to the user; the user becomes impatient if the UI does not react. Maybe you've had this experience with Visual Studio as well. Good news: Visual Studio has become a lot better in reacting faster in many scenarios.

The .NET Framework always offered calling methods asynchronously. However, using synchronous methods was a lot easier than calling their asynchronous variant. This changed with C# 5.0. Programming asynchronously has become as easy as writing synchronous programs. New C# keywords are based on the .NET Parallel Library that is available since .NET 4.0. Now the language offers productivity features.

Windows Store Apps and the Windows Runtime

Windows Store apps can be programmed with C# using the Windows Runtime and a subset of the .NET Framework. The Windows Runtime is a new native API that offers classes, methods, properties, and events that look like .NET; although it is native. For using language projection features, the .NET runtime has been enhanced. With .NET 4.5, the .NET 4.0 runtime gets an in-place update.

Enhancements with Data Access

The ADO.NET Entity Framework offered important new features. Its version changed from 4.0 with .NET 4.0 to 5.0 with .NET 4.5, and to 6.0 with .NET 4.5.1. After the release of .NET 4.0, the Entity Framework already received updates with versions 4.1, 4.2, and 4.3. New features such as Code First, spatial types, using enums, and table-valued functions are now available.

Enhancements with WPF

For programming Windows desktop applications, WPF has been enhanced. Now you can fill collections from a non-UI thread; the ribbon control is now part of the framework; weak references with events have been made easier; validation can be done asynchronously with the `INotifyDataErrorInfo` interface; and live shaping allows easy dynamic sorting and grouping with data that changes.

ASP.NET MVC

Visual Studio 2010 included ASP.NET MVC 2.0. With the release of Visual Studio 2013, ASP.NET MVC 5.0 is available. ASP.NET MVC supplies you with the means to create ASP.NET using the model-view-controller model that many developers expect. ASP.NET MVC provides developers with testability, flexibility, and maintainability in the applications they build. ASP.NET MVC is not meant to be a replacement for ASP.NET Web Forms but is simply a different way to construct your applications.

WHERE C# FITS IN

In one sense, C# is the same thing to programming languages that .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 15 years, Visual Basic 2013 and C++ have undergone expansion. Although Visual Basic and C++ have resulted in hugely powerful languages, both languages also suffer from problems because of the legacies left over from the way they evolved.

For Visual Basic 6 and earlier versions, the main strength of the language was that it was simple to understand and made 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 large applications quickly became disorganized and hard to maintain. Also, 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 lend itself to well-structured or object-oriented programs.

C++, on the other hand, has its roots in the ANSI C++ language definition. It is not 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 for 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 designed for high-performance tasks on Windows, and to achieve that, it has 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 enters .NET — a completely revolutionary environment that has brought forth 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 to the current Visual Basic 2013, a language that retains some of the basic VB syntax but that is so different in design from the original VB that it can be considered, for all practical purposes, a new language.

It is in this context that Microsoft has provided developers an alternative — a language designed specifically for .NET and designed with a clean slate. C# is the result. Officially, Microsoft describes C# as a “simple, modern, object-oriented, and type-safe programming language derived from C and C++.” Most independent observers would probably change that to “derived from C, C++, and Java.” Such descriptions are technically accurate but do little to convey the beauty or elegance of the language. Syntactically, C# is similar to both C++ and Java, to such an extent that many keywords are the same, and C# also shares the same block structure with braces (`{ }`) to mark blocks of code and semicolons to separate statements. The first impression of a piece of C# code is that it looks quite like C++ or Java code. Beyond that initial similarity, however, C# is a lot easier to learn than C++ and of comparable difficulty to Java. Its design is more in tune with modern developer tools than both of those other languages, and it has been designed to provide, simultaneously, the ease of use of Visual Basic and the high-performance, low-level memory access of C++, if required. Some of the features of C# follow:

- Full support for classes and object-oriented programming, including interface and implementation inheritance, virtual functions, and operator overloading.
- A consistent and well-defined set of basic types.
- Built-in support for an automatic generation of XML documentation.
- Automatic cleanup of dynamically allocated memory.

- The facility to mark classes or methods with user-defined attributes. This can be useful for documentation and can have some effects on compilation (for example, marking methods to be compiled only in debug builds).
- Full access to the .NET base class library and easy access to the Windows API (if you need it, which will not be often).
- Pointers and direct memory access are available if required, but the language has been designed in such a way that you can work without them in almost all cases.
- Support for properties and events in the style of Visual Basic.
- Just by changing the compiler options, you can compile either to an executable or to a library of .NET components that can be called up by other code in the same way as ActiveX controls (COM components).
- C# can be used to write ASP.NET dynamic web pages and XML web services.

Most of these statements, it should be pointed out, also apply to Visual Basic 2013 and Managed C++. Because C# is designed from the start to work with .NET, however, means that its support for the features of .NET is both more complete and offered within the context of a more suitable syntax than those of other languages. Although the C# language is similar to Java, there are some improvements; in particular, Java is not designed to work with the .NET environment.

Before leaving the subject, you must understand a couple of limitations of C#. The one area the language is not designed for is time-critical or extremely high-performance code — the kind where you are worried about whether a loop takes 1,000 or 1,050 machine cycles to run through, and you need to clean up your resources the millisecond they are no longer needed. C++ is likely to continue to reign supreme among low-level languages in this area. C# lacks certain key facilities needed for extremely high-performance apps, including the capability to specify inline functions and destructors guaranteed to run at particular points in the code. However, the proportions of applications that fall into this category are low.

WHAT YOU NEED TO WRITE AND RUN C# CODE

The .NET Framework 4.5.1 can run on the client operating systems Windows Vista, 7, 8, 8.1, and the server operating systems Windows Server 2008, 2008 R2, 2012, and 2012 R2. To write code using .NET, you need to install the .NET 4.5.1 SDK.

In addition, unless you intend to write your C# code using a text editor or some other third-party developer environment, you almost certainly also want Visual Studio 2013. The full SDK is not needed to run managed code, but the .NET runtime is needed. You may find you need to distribute the .NET runtime with your code for the benefit of those clients who do not have it already installed.

WHAT THIS BOOK COVERS

This book starts by reviewing the overall architecture of .NET in Chapter 1 to give you the background you need to write managed code. After that, the book is divided into a number of sections that cover both the C# language and its application in a variety of areas.

Part I: The C# Language

This section gives a good grounding in the C# language. This section doesn't presume knowledge of any particular language; although, it does assume you are an experienced programmer. You start by looking at