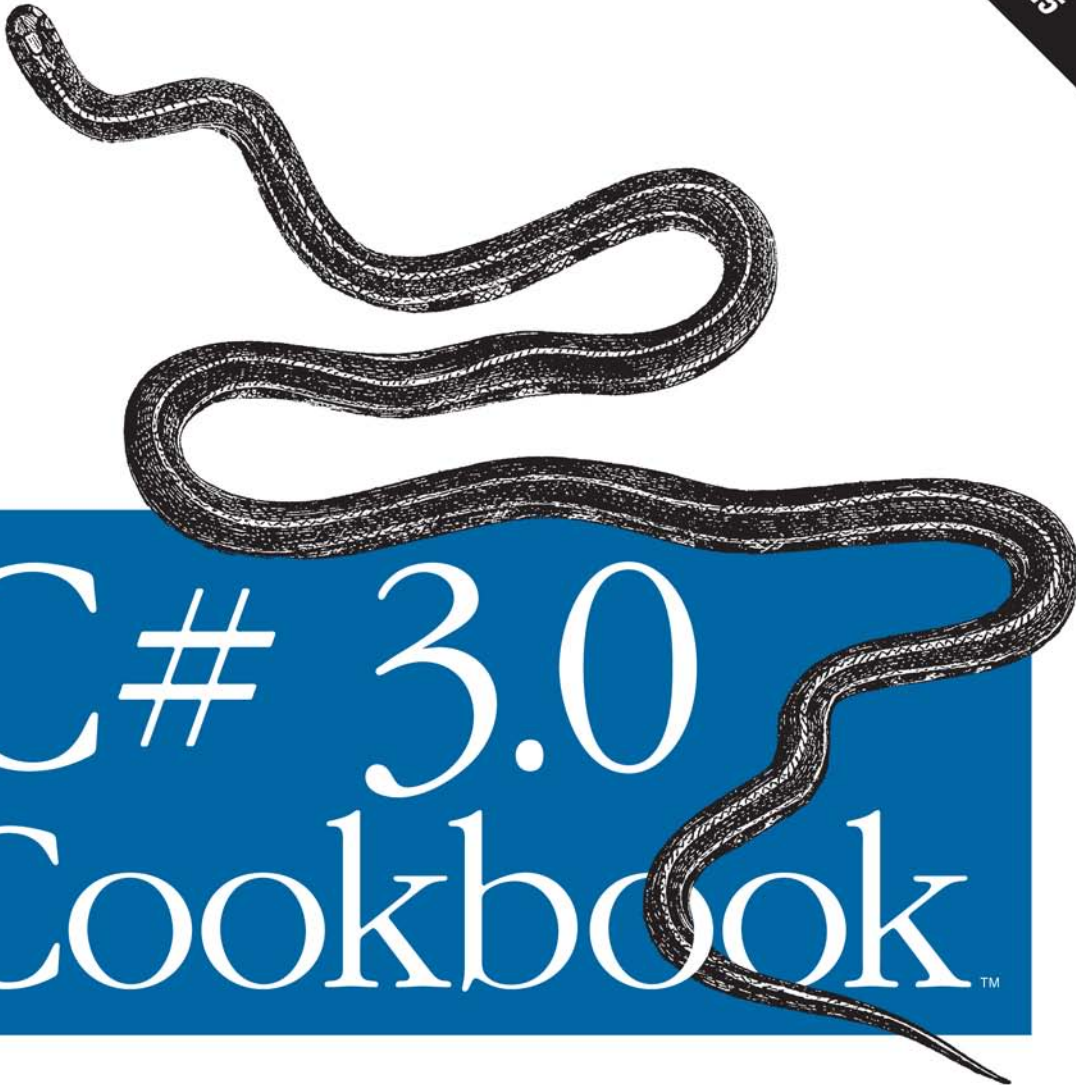


*More Than 250 Solutions for
C# 3.0 Programmers*

3rd Edition
Thoroughly revised
for C# 3.0 and .NET 3.5



C# 3.0 Cookbook™

O'REILLY®

Jay Hilyard & Stephen Teilhet

C# 3.0 Cookbook™

Other Microsoft .NET resources from O'Reilly

Related titles	Building a Web 2.0 Portal with ASP.NET 3.5	Programming ASP.NET
	C# 3.0 Design Patterns	Programming C#
	Learning C#	Visual C# 2005: A Developer's Notebook™

.NET Books Resource Center

dotnet.oreilly.com is a complete catalog of O'Reilly's books on .NET and related technologies, including sample chapters and code examples.



ONDotnet.com provides independent coverage of fundamental, interoperable, and emerging Microsoft .NET programming and web services technologies.

Conferences

O'Reilly brings diverse innovators together to nurture the ideas that spark revolutionary industries. We specialize in documenting the latest tools and systems, translating the innovator's knowledge into useful skills for those in the trenches. Visit *conferences.oreilly.com* for our upcoming events.



Safari Bookshelf (*safari.oreilly.com*) is the premier online reference library for programmers and IT professionals. Conduct searches across more than 1,000 books. Subscribers can zero in on answers to time-critical questions in a matter of seconds. Read the books on your Bookshelf from cover to cover or simply flip to the page you need. Try it today for free.

THIRD EDITION

C# 3.0 Cookbook™

Jay Hilyard and Stephen Teilhet

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

3.0 Cookbook™, Third Edition

by Jay Hilyard and Stephen Teilhet

Copyright © 2008 Jay Hilyard and Stephen Teilhet. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*safari.oreilly.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor: John Osborn

Production Editor: Adam Witwer

Production Services: nSight, Inc.

Cover Designer: Karen Montgomery

Interior Designer: David Futato

Illustrators: Robert Romano and Jessamyn Read

Printing History:

January 2004: First Edition.

January 2006: Second Edition.

December 2007: Third Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. The *Cookbook* series designations, *C# 3.0 Cookbook*, the image of a garter snake, and related trade dress are trademarks of O'Reilly Media, Inc.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

ISBN-10: 0-596-51610-X

ISBN-13: 978-0-596-51610-9

[M]

To Brooke
My wife, my best friend, and the most supportive
person I know. This one was for you;
you earned it.

—Jay Hilyard

To my loving wife Kandis and my two wonderful
sons, Patrick and Nicholas.

—Stephen Teilhet

Table of Contents

Preface	xvii
1. Language Integrated Query (LINQ)	1
1.1 Query a Message Queue	2
1.2 Using Set Semantics with Data	5
1.3 Reuse Parameterized Queries with LINQ to SQL	8
1.4 Sort Results in a Culture-Sensitive Manner	10
1.5 Adding Functional Extensions for Use with LINQ	12
1.6 Query and Join Across Data Repositories	16
1.7 Querying Configuration Files with LINQ	19
1.8 Creating XML Straight from a Database	22
1.9 Being Selective About Your Query Results	31
1.10 Using LINQ with Collections That Don't Support IEnumerable<T>	33
2. Strings and Characters	36
2.1 Determining the Kind of Character a Char Contains	36
2.2 Controlling Case Sensitivity When Comparing Two Characters	40
2.3 Finding the Location of All Occurrences of a String Within Another String	42
2.4 Controlling Case Sensitivity When Comparing Two Strings	46
2.5 Comparing a String to the Beginning or End of a Second String	47
2.6 Inserting Text into a String	49
2.7 Removing or Replacing Characters Within a String	51
2.8 Encoding Binary Data As Base64	53
2.9 Decoding a Base64-Encoded Binary	54
2.10 Converting a String Returned As a Byte[] Back into a String	56
2.11 Passing a String to a Method That Accepts Only a Byte[]	57

2.12	Converting Strings to Other Types	59
2.13	Creating a Delimited String	62
2.14	Extracting Items from a Delimited String	63
2.15	Iterating over Each Character in a String	64
2.16	Pruning Characters from the Head and/or Tail of a String	65
2.17	Testing a String for Null or Empty	66
2.18	Appending a Line	67
3.	Classes and Structures	68
3.1	Creating Union-Type Structures	70
3.2	Making a Type Sortable	72
3.3	Making a Type Searchable	77
3.4	Indirectly Overloading the +=, -=, /=, and *= Operators	80
3.5	Indirectly Overloading the &&, , and ?: Operators	82
3.6	Making Error-Free Expressions	85
3.7	Reducing Your Boolean Logic	88
3.8	Converting Between Simple Types in a Programming Language-Agnostic Manner	91
3.9	Determining When to Use the cast Operator, the as Operator, or the is Operator	97
3.10	Casting with the as Operator	99
3.11	Determining a Variable's Type with the is Operator	101
3.12	Returning Multiple Items from a Method	104
3.13	Parsing Command-Line Parameters	106
3.14	Initializing a Constant Field at Runtime	117
3.15	Building Cloneable Classes	120
3.16	Assuring an Object's Disposal	124
3.17	Disposing of Unmanaged Resources	126
3.18	Determining Where Boxing and Unboxing Occur	133
4.	Generics	137
4.1	Deciding When and Where to Use Generics	137
4.2	Understanding Generic Types	138
4.3	Replacing the ArrayList with Its Generic Counterpart	146
4.4	Replacing the Stack and Queue with Their Generic Counterparts	150
4.5	Using a Linked List	155
4.6	Creating a Value Type That Can Be Initialized to Null	158
4.7	Reversing the Contents of a Sorted List	160
4.8	Making Read-Only Collections the Generic Way	162

4.9	Replacing the Hashtable with Its Generic Counterpart	164
4.10	Using foreach with Generic Dictionary Types	168
4.11	Constraining Type Arguments	169
4.12	Initializing Generic Variables to Their Default Values	173
5.	Collections	175
5.1	Swapping Two Elements in an Array	177
5.2	Reversing an Array Quickly	178
5.3	Writing a More Flexible StackTrace Class	181
5.4	Determining the Number of Times an Item Appears in a List<T>	182
5.5	Retrieving All Instances of a Specific Item in a List<T>	185
5.6	Inserting and Removing Items from an Array	188
5.7	Keeping Your List<T> Sorted	191
5.8	Sorting a Dictionary's Keys and/or Values	193
5.9	Creating a Dictionary with Max and Min Value Boundaries	194
5.10	Storing Snapshots of Lists in an Array	198
5.11	Persisting a Collection Between Application Sessions	199
5.12	Testing Every Element in an Array or List<T>	201
5.13	Performing an Action on Each Element in an Array or List<T>	202
5.14	Creating a Read-Only Array or List<T>	204
6.	Iterators, Partial Types, and Partial Methods	206
6.1	Creating an Iterator on a Generic Type	207
6.2	Creating an Iterator on a Nongeneric Type	209
6.3	Creating Custom Enumerators	211
6.4	Implementing Iterator Logic	215
6.5	Forcing an Iterator to Stop Iterating	218
6.6	Dealing with Finally Blocks and Iterators	220
6.7	Implementing Nested foreach Functionality in a Class	224
6.8	Organizing Your Interface Implementations	229
6.9	Generating Code That Is No Longer in Your Main Code Paths	234
6.10	Adding Hooks to Generated Entities	237
7.	Exception Handling	240
7.1	Knowing When to Catch and Rethrow Exceptions	247
7.2	Assuring Exceptions Are Not Lost When Using Finally Blocks	248
7.3	Handling Exceptions Thrown from Methods Invoked via Reflection	251
7.4	Preventing Unhandled Exceptions	254
7.5	Getting Exception Information	256

7.6	Getting to the Root of a Problem Quickly	260
7.7	Creating a New Exception Type	261
7.8	Obtaining a Stack Trace	269
7.9	Breaking on a First-Chance Exception	272
7.10	Handling Exceptions Thrown from an Asynchronous Delegate	275
7.11	Giving Exceptions the Extra Info They Need with Exception.Data	277
7.12	Dealing with Unhandled Exceptions in WinForms Applications	279
7.13	Dealing with Unhandled Exceptions in Windows Presentation Foundation (WPF) Applications	281
7.14	Analyzing Exceptions for Common Errors	283
8.	Diagnostics	286
8.1	Providing Fine-Grained Control over Debugging/Tracing Output	287
8.2	Determining Whether a Process Has Stopped Responding	290
8.3	Using Event Logs in Your Application	292
8.4	Searching Event Log Entries	299
8.5	Watching the Event Log for a Specific Entry	302
8.6	Implementing a Simple Performance Counter	304
8.7	Enabling and Disabling Complex Tracing Code	307
8.8	Capturing Standard Output for a Process	311
8.9	Creating Custom Debugging Displays for Your Classes	313
9.	Delegates, Events, and Lambda Expressions	316
9.1	Controlling When and If a Delegate Fires Within a Multicast Delegate	318
9.2	Obtaining Return Values from Each Delegate in a Multicast Delegate	322
9.3	Handling Exceptions Individually for Each Delegate in a Multicast Delegate	324
9.4	Converting Delegate Invocation from Synchronous to Asynchronous	327
9.5	An Advanced Interface Search Mechanism	330
9.6	Observing Additions and Modifications to Dictionaries	332
9.7	Using Lambda Expressions	344
9.8	Set Up Event Handlers Without the Mess	348
9.9	Using Different Parameter Modifiers in Lambda Expressions	352
9.10	Using Closures in C#	356
9.11	Performing Multiple Operations on a List Using Functors	361

10. Regular Expressions	366
10.1 Enumerating Matches	367
10.2 Extracting Groups from a MatchCollection	370
10.3 Verifying the Syntax of a Regular Expression	373
10.4 Quickly Finding Only the Last Match in a String	375
10.5 Augmenting the Basic String Replacement Function	376
10.6 Implementing a Better Tokenizer	379
10.7 Counting Lines of Text	380
10.8 Returning the Entire Line in Which a Match Is Found	383
10.9 Finding a Particular Occurrence of a Match	387
10.10 Using Common Patterns	389
11. Data Structures and Algorithms	394
11.1 Creating a Hash Code for a Data Type	394
11.2 Creating a Priority Queue	402
11.3 Creating a One-to-Many Map (MultiMap)	410
11.4 Creating a Binary Search Tree	418
11.5 Creating an n-ary Tree	432
11.6 Using a HashSet Object	444
12. Filesystem I/O	449
12.1 Manipulating File Attributes	450
12.2 Renaming a File	452
12.3 Outputting a Platform-Independent EOL Character	453
12.4 Manipulating Directory Attributes	455
12.5 Renaming a Directory	457
12.6 Searching for Directories or Files Using Wildcards	459
12.7 Obtaining the Directory Tree	464
12.8 Parsing a Path	466
12.9 Parsing Paths in Environment Variables	468
12.10 Launching and Interacting with Console Utilities	469
12.11 Locking Subsections of a File	471
12.12 Waiting for an Action to Occur in the Filesystem	474
12.13 Comparing Version Information of Two Executable Modules	477
12.14 Querying Information for All Drives on a System	479
12.15 Compressing and Decompressing Your Files	482

13. Reflection	489
13.1 Listing Referenced Assemblies	490
13.2 Listing Exported Types	492
13.3 Finding Overridden Methods	493
13.4 Finding Members in an Assembly	499
13.5 Determining and Obtaining Nested Types Within an Assembly	500
13.6 Displaying the Inheritance Hierarchy for a Type	501
13.7 Finding the Subclasses of a Type	504
13.8 Finding All Serializable Types Within an Assembly	505
13.9 Dynamically Invoking Members	507
13.10 Determining If a Type or Method Is Generic	511
13.11 Accessing Local Variable Information	512
13.12 Creating a Generic Type	514
14. Web	516
14.1 Converting an IP Address to a Hostname	516
14.2 Converting a Hostname to an IP Address	517
14.3 Parsing a URI	518
14.4 Handling Web Server Errors	522
14.5 Communicating with a Web Server	524
14.6 Going Through a Proxy	525
14.7 Obtaining the HTML from a URL	527
14.8 Using the Web Browser Control	528
14.9 Tying Database Tables to the Cache	530
14.10 Prebuilding an ASP.NET Web Site Programmatically	532
14.11 Escaping and Unescaping Data for the Web	535
14.12 Using the UriBuilder Class	537
14.13 Inspect and Change Your Web Application Configuration	539
14.14 Using Cached Results When Working with HTTP for Faster Performance	541
14.15 Checking Out a Web Server's Custom Error Pages	543
15. XML	548
15.1 Reading and Accessing XML Data in Document Order	548
15.2 Reading XML on the Web	552
15.3 Querying the Contents of an XML Document	554
15.4 Validating XML	558
15.5 Creating an XML Document Programmatically	564
15.6 Detecting Changes to an XML Document	566

15.7	Handling Invalid Characters in an XML String	569
15.8	Transforming XML	572
15.9	Tearing Apart an XML Document	579
15.10	Putting Together an XML Document	585
15.11	Validating Modified XML Documents Without Reloading	591
15.12	Extending Transformations	595
15.13	Getting Your Schemas in Bulk from Existing XML Files	599
15.14	Passing Parameters to Transformations	601
16.	Networking	606
16.1	Writing a TCP Server	606
16.2	Writing a TCP Client	612
16.3	Simulating Form Execution	615
16.4	Transferring Data via HTTP	619
16.5	Using Named Pipes to Communicate	621
16.6	Pinging Programmatically	629
16.7	Send SMTP Mail Using the SMTP Service	631
16.8	Use Sockets to Scan the Ports on a Machine	636
16.9	Use the Current Internet Connection Settings	641
16.10	Transferring Files Using FTP	648
17.	Security	651
17.1	Controlling Access to Types in a Local Assembly	651
17.2	Encrypting/Decrypting a String	661
17.3	Encrypting and Decrypting a File	665
17.4	Cleaning Up Cryptography Information	670
17.5	Verifying That a String Remains Uncorrupted Following Transmission	672
17.6	Storing Data Securely	676
17.7	Making a Security Assert Safe	683
17.8	Verifying That an Assembly Has Been Granted Specific Permissions	685
17.9	Minimizing the Attack Surface of an Assembly	687
17.10	Obtaining Security/Audit Information	688
17.11	Granting/Revoking Access to a File or Registry Key	693
17.12	Protecting String Data with Secure Strings	696
17.13	Securing Stream Data	699
17.14	Encrypting web.config Information	708
17.15	Obtaining the Full Reason a SecurityException Was Thrown	710
17.16	Achieving Secure Unicode Encoding	712
17.17	Obtaining a Safer File Handle	713

18. Threading and Synchronization	716
18.1 Creating Per-Thread Static Fields	716
18.2 Providing Thread-Safe Access to Class Members	719
18.3 Preventing Silent Thread Termination	725
18.4 Being Notified of the Completion of an Asynchronous Delegate	727
18.5 Storing Thread-Specific Data Privately	730
18.6 Granting Multiple Access to Resources with a Semaphore	734
18.7 Synchronizing Multiple Processes with the Mutex	738
18.8 Using Events to Make Threads Cooperate	750
18.9 Get the Naming Rights for Your Events	752
18.10 Performing Atomic Operations Among Threads	755
18.11 Optimizing Read-Mostly Access	757
19. Toolbox	770
19.1 Dealing with Operating System Shutdown, Power Management, or User Session Changes	770
19.2 Controlling a Service	775
19.3 List What Processes an Assembly Is Loaded In	778
19.4 Using Message Queues on a Local Workstation	780
19.5 Finding the Path to the Current Framework Version	783
19.6 Determining the Versions of an Assembly That Are Registered in the Global Assembly Cache (GAC)	784
19.7 Capturing Output from the Standard Output Stream	787
19.8 Running Code in Its Own AppDomain	789
19.9 Determining the Operating System and Service Pack Version of the Current Operating System	791
20. Numbers and Enumerations	793
20.1 Converting Between Degrees and Radians	795
20.2 Using the Bitwise Complement Operator with Various Data Types	796
20.3 Converting a Number in Another Base to Base10	797
20.4 Determining Whether a String Is a Valid Number	798
20.5 Rounding a Floating-Point Value	799
20.6 Choosing a Rounding Algorithm	800
20.7 Converting Between Temperature Scales	801
20.8 Safely Performing a Narrowing Numeric Cast	802
20.9 Displaying an Enumeration Value As a String	804

20.10	Converting Plain Text to an Equivalent Enumeration Value	807
20.11	Testing for a Valid Enumeration Value	808
20.12	Testing for a Valid Enumeration of Flags	810
20.13	Using Enumerated Members in a Bit Mask	812
20.14	Determining Whether One or More Enumeration Flags Are Set	815
20.15	Determining the Integral Part of a Decimal or Double	819
Index	821

Preface

C# is a language targeted at developers for the Microsoft .NET platform who have already worked with a C-like language such as C, C++, or Java. Unlike previous versions of C or C++ for the Microsoft Windows platform, C# code runs under a *managed execution environment*. Microsoft portrays C# as a modern and innovative language for .NET development and continues to deliver on that with new features such as Language Integrated Query (LINQ). The new features in C# 3.0 allow for more of a declarative and functional style of programming, when that is appropriate, while it still has great object-oriented features as well. The main idea is to use the style of programming that fits your problem, and C# will support your endeavor.

C# allows you to perform many C/C++-like functions, such as direct memory access via pointers and operator overloading, that are not supported in Visual Basic .NET. C# is the system-level programming language for .NET. You can still do great application-level work in C#, but it really shines when you need to build code a little closer to the Framework.

If you have seen C#, you may have noticed that it looks a lot like Java; Java programmers will feel very much at home in C# once they learn the Framework SDK. C# can also be a great language for Visual Basic .NET programmers when they need a little more control over what the code is doing and don't want to have to write C++ to gain an advantage. On the Web, you'll find a large community of people doing really neat things with C# and tons of sample code on sites such as <http://www.codeplex.com> and <http://www.codeproject.com>.

We started writing this book together based on programming problems we ran into when we were first learning C# and have continued to expand it based on new challenges and capabilities in the language. In this edition, we have reworked the approach of many solutions to take advantage of LINQ and have also created entirely new solutions based on LINQ and the other new features in C# 3.0. We hope that it will help you get past some of the common (and not-so-common) pitfalls and initial questions everyone has when learning a new language as well as the slightly off-the-beaten-path items that come up during a development cycle. There

are recipes addressing things we found missing from the .NET Framework Class Library (FCL), even though Microsoft has provided tons of functionality to keep folks from reinventing the wheel. Some of these solutions you might immediately use, and some may never darken your door, but we hope this book helps you get the most out of C# and the .NET Framework.

The book is laid out with respect to the types of problems you will solve as you progress through your life as a C# programmer. These solutions are called *recipes*; each recipe consists of a single problem, its solution, a discussion of the solution and other relevant related information, and finally, where you can look for more information about the classes used from the FCL, other books addressing this topic, related articles, and other recipes. The question-answer format provides complete solutions to problems, making the book easy to read and use. Nearly every recipe contains a complete, documented code sample, showing you how to solve the specific problem, as well as a discussion of how the underlying technology works and a list of alternatives, limitations, and other considerations when appropriate.

Who This Book Is For

You don't have to be an experienced C# or .NET developer to use this book—it is designed for users of all levels. This book provides solutions to problems that developers face every day as well as some that may come along less frequently. The recipes are targeted at the real-world developer who needs to solve problems now, not learn lots of theory before being able to solve the problem. While reference or tutorial books can teach general concepts, they do not generally provide the help you need in solving real-world problems. We choose to teach by example, the natural way for most people to learn.

The majority of the problems addressed in this book are frequently faced by C# developers, but some of the more advanced problems call for more intricate solutions that combine many techniques. Each recipe is designed to help you quickly understand the problem, learn how to solve it, and find out any potential trade-offs or ramifications to help you solve your problems quickly, efficiently, and with minimal effort.

To save you even the effort of typing in the solution, we provide the sample code for the book on the O'Reilly web site to facilitate the “editor inheritance” mode of development (copy and paste) as well as to help less-experienced developers see good programming practice in action. The sample code provides a running test harness that exercises each of the solutions, but enough of the code is provided in each solution in the book to allow you to implement the solution without the sample code. The sample code is available from the book's catalog page: <http://www.oreilly.com/catalog/9780596516109>.

What You Need to Use This Book

To run the samples in this book, you need a computer running Windows XP or later. A few of the networking and XML solutions require Microsoft Internet Information Server (IIS) Version 5.1 or later, and the FTP recipes in the Networking chapter require a locally configured FTP server.

To open and compile the samples in this book, you need Visual Studio .NET 2008. If you are proficient with the downloadable Framework SDK and its command-line compilers, you should not have any trouble following the text of this book and the code samples.

Platform Notes

The solutions in this book were developed using Visual Studio .NET 2008. The differences between C# 3.0 and C# 2.0 are significant, and the sample code has changed from the second edition to reflect that.

It is worth mentioning that although C# is now at version 3.0, the .NET Framework has progressed to version 3.5. .NET 3.0 introduced Windows Communication Foundation, Windows Presentation Foundation, and Windows Workflow Foundation as additional functionality to the 2.0 framework base, but C# was not changed. Now in C# 3.0, there is a bunch of new functionality, mostly due to LINQ and the ability to do more functional programming.

How This Book Is Organized

This book is organized into 20 chapters, each of which focuses on a particular topic in creating C# solutions. The following paragraphs summarize each chapter to give you an overview of this book's contents:

Chapter 1, *Language Integrated Query (LINQ)*

This chapter covers Language Integrated Query (LINQ) and its usage with objects, ADO.NET, and XML. There are recipes using many of the Standard Query Operators and showing how to use some of the query operators that are not keywords in the language, but are still quite powerful.

Chapter 2, *Strings and Characters*

This chapter covers both the String and Char data types. Recipes show such things as how to compare strings in various ways, encode/decode strings, break strings apart, and put them back together again.

Chapter 3, *Classes and Structures*

This large chapter contains recipes dealing with both class and structure data types. This chapter covers a wide range of recipes, from design patterns to converting a class to a full-blown command-line argument-processing system.

Chapter 4, *Generics*

This chapter focuses on the generics capacity in C#, which allows you to have code operate uniformly on values of different types. There are recipes to help your general understanding of generics as well as when they are appropriate to use, what support is provided in the Framework for them, and how to create custom implementations of collections using generics.

Chapter 5, *Collections*

This chapter examines recipes that make use of collections. The collection recipes make use of—as well as extend the functionality of—the array (single, multi, and jagged), the `List<T>`, and the `Hashtable`. The generic-based collections are explored, and the various ways to create your own strongly typed collection are also discussed.

Chapter 6, *Iterators, Partial Types, and Partial Methods*

In this chapter, two of the features of C# are used to solve very different programming problems. We show how you can implement iterators for generic and nongeneric types and implement `foreach` functionality using iterators, as well as custom iterator implementations. The other feature of C# in this chapter is partial types and methods. We show how you can use partial types and methods to do such things as better segmenting your code and how to generate code that is more easily extensible.

Chapter 7, *Exception Handling*

The recipes in this chapter focus on the best ways to implement exception handling in your application. Preventing unhandled exceptions, reading and displaying stack traces, and throwing/rethrowing exceptions are included recipes. In addition, specific recipes show how to overcome some tricky situations, such as exceptions from late-bound called methods.

Chapter 8, *Diagnostics*

This chapter presents recipes that use data types that fall under the `System.Diagnostics` namespace. Recipes deal with the `Trace/Debug` classes, event logs, processes, performance counters, and custom debugger displays for your types.

Chapter 9, *Delegates, Events, and Lambda Expressions*

This chapter's recipes show how delegates, events, and lambda expressions can be used in your applications. Recipes allow manipulation of delegates that call more than one method, synchronous delegates, and asynchronous delegates. Lambda expressions are explored, and recipes show their usage in place of old-style delegates as well as their use in implementing closures and functors.

Chapter 10, *Regular Expressions*

This chapter covers a useful set of classes that are employed to run regular expressions against strings. Recipes enumerate regular expression matches, break up strings into tokens, find/replace characters, and verify the syntax of a regular expression. We also include a recipe that contains many common regular expression patterns.

Chapter 11, *Data Structures and Algorithms*

This chapter ventures a bit outside of what is provided for you in the .NET Framework Class Library and implements certain data structures and algorithms that are not in the FCL, or possibly are not in existence exactly the way you would like to use them, but are ones that you have used to solve problems before. Items such as queues, maps, trees, and hashes are examined.

Chapter 12, *Filesystem I/O*

This chapter deals with file system interactions in four distinct ways. The first way is to look at typical file interactions; the second way looks at directory- or folder-based interactions; the third way deals with paths and temporary files; and the fourth way deals with advanced file system I/O topics.

Chapter 13, *Reflection*

This chapter shows ways to use the built-in assembly inspection system provided by the .NET Framework to determine what types, interfaces, and methods are implemented within an assembly and how to access them in a late-bound fashion.

Chapter 14, *Web*

This chapter covers accessing a web site and its content as well as programmatically determining web site configuration. Among the recipes in this chapter are using the web browser control and setting up caching triggers to refresh cached data when a database table changes.

Chapter 15, *XML*

If you use .NET, it is likely that you will be dealing with XML to one degree or another; in this chapter, we explore some of the uses for XML and how to program against it using LINQ to XML, the XmlReader/XmlWriter, and XmlDocument. There are examples using both XPath and XSLT, and topics such as the validation of XML and transformation of XML to HTML are shown.

Chapter 16, *Networking*

This chapter explores the connectivity options provided by the .NET Framework and how to programmatically access network resources. Recipes for using TCP/IP directly, named pipes for communication, building your own port scanner, and more are covered here.

Chapter 17, *Security*

There are many ways to write secure code and protect data using the .NET Framework, and in this chapter, we explore areas such as controlling access to types, encryption and decryption, securely storing data, and using programmatic and declarative security.

Chapter 18, *Threading and Synchronization*

This chapter addresses the subject of using multiple threads of execution in a .NET program and issues such as how to implement threading in your application, protecting resources from and allowing safe concurrent access, storing per-thread data, and how to use the synchronization primitives in .NET to write thread-safe code.

Chapter 19, *Toolbox*

This chapter has recipes for those random sorts of operations that developers run into over and over again, such as determining locations of system resources, sending email, and working with services. It also covers some less frequently accessed but helpful application pieces such as message queuing, running code in a separate AppDomain, and finding the versions of assemblies in the GAC.

Chapter 20, *Numbers and Enumerations*

This chapter focuses on the numeric and enumeration data types used in C# code. Recipes cover such things as numeric conversions, using bitwise operators on numbers, and testing strings to determine whether they contain a numeric value. The display, conversion, and testing of enumeration types and recipes on using enumerations that consist of bit flags are also shown.

In some cases, certain recipes are related. In these cases, the See Also section of the recipe as well as some text in the Discussion will note the relationships.

What Was Left Out

This book is not a reference or a primer about C#. Some good primers and reference books are *C# in a Nutshell*, *C# Language Pocket Reference*, and *Learning C#*, all titles available from O'Reilly. The MSDN Library is also invaluable. It is included with Visual Studio .NET 2008 and available online at <http://msdn.microsoft.com/library/default.asp>.

This book is not about how to use Visual Studio .NET 2008 to build, compile, and deploy applications. See *Mastering Visual Studio .NET* (O'Reilly) for excellent coverage of these topics.

Conventions Used in This Book

This book uses the following typographic conventions:

Italic

Used for URLs, names of directories and files, options, and occasionally for emphasis.

Constant width

Used for program listings and for code items such as commands, options, switches, variables, attributes, keys, functions, types, classes, namespaces, methods, modules, properties, parameters, values, objects, events, event handlers, XML tags, HTML tags, macros, the contents of files, and the output from commands.

Constant width bold

Used in program listings to highlight an important part of the code.

Constant width italic

Used to indicate replaceable parts of code.

//...

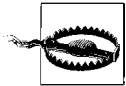
Ellipses in C# code indicate text that has been omitted for clarity.

<!--...-->

Ellipses in XML Schemas and documents' code indicate text that has been omitted for clarity.



This icon indicates a tip, suggestion, or general note.



This icon indicates a warning or caution.

About the Code

Nearly every recipe in this book contains one or more code samples. These samples are included in a single solution and are pieces of code and whole projects that are immediately usable in your application. Most of the code samples are written within a class or structure, making it easier to use within your applications. In addition to this, any using directives are included for each recipe so that you will not have to search for which ones to include in your code.

Complete error handling is included only in critical areas, such as input parameters. This allows you to easily see what is correct input and what is not. Many recipes omit error handling. This makes the solution easier to understand by focusing on the key concepts.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

- [click **Jonathan Edwards: Writings from the Great Awakening \(Library of America, Volume 24\)** online](#)
- [download online The Dog Killer of Utica \(Eliot Conte, Book 2\) pdf, azw \(kindle\)](#)
- [download *The Green Suit* pdf](#)
- [download **Habitat for Humanity: How to Build a House \(Revised & Updated\)**](#)
- [read Learning Cypher online](#)

- <http://weddingcellist.com/lib/Jonathan-Edwards--Writings-from-the-Great-Awakening--Library-of-America--Volume-245-.pdf>
- <http://junkrobots.com/ebooks/Reinvention--Sewing-with-Rescued-Materials.pdf>
- <http://chelseaprintandpublishing.com/?freebooks/The-Green-Suit.pdf>
- <http://www.experienceolvera.co.uk/library/Habitat-for-Humanity--How-to-Build-a-House--Revised---Updated-.pdf>
- <http://korplast.gr/lib/Getting-It-Wrong--Ten-of-the-Greatest-Misreported-Stories-in-American-Journalism.pdf>