

Joe Mayo



# C# 3.0

**UNLEASHED**

With the .NET  
Framework 3.5

**SAMS**

## **C# 3.0 Unleashed**

### **With the .NET Framework 3.5**

Copyright © 2008 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-32981-4

ISBN-10: 0-672-32981-6

*Library of Congress Cataloging-in-Publication Data*

Mayo, Joseph.

C# 3.0 unleashed : with the .NET Framework 3.5 / Joe Mayo. — 1st ed.  
p. cm.

ISBN 978-0-672-32981-4

1. C# (Computer program language) 2. Microsoft .NET Framework. I.  
Title.

QA76.73.C154M38 2008

006.7'882—dc22

2008026117

Printed in the United States of America

First Printing June 2008

### **Trademarks**

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

### **Warning and Disclaimer**

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

### **Bulk Sales**

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

#### **U.S. Corporate and Government Sales**

**1-800-382-3419**

**corpsales@pearsontechgroup.com**

For sales outside of the U.S., please contact

#### **International Sales**

**international@pearson.com**

#### **Editor-in-Chief**

Karen Gettman

#### **Executive Editor**

Neil Rowe

#### **Acquisitions Editor**

Brook Farling

#### **Development Editor**

Mark Renfrow

#### **Managing Editor**

Kristy Hart

#### **Project Editor**

Andrew Beaster

#### **Copy Editor**

Keith Cline

#### **Indexer**

Brad Herriman

#### **Proofreader**

San Dee Phillips

#### **Technical Editors**

Tony Gravagno

Todd Meister

J. Boyd Nolan

#### **Publishing**

##### **Coordinator**

Cindy Teeters

#### **Cover Designer**

Gary Adair

#### **Composition**

Jake McFarland



The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days. Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book:

- ▶ Go to <http://www.sampublishing.com/safarienabled>
- ▶ Complete the brief registration form
- ▶ Enter the coupon code **VDD7-6QGD-AVQZ-VFEX-CZ76**

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please e-mail [customer-service@safaribooksonline.com](mailto:customer-service@safaribooksonline.com).

# Introduction

Welcome to *C# 3.0 Unleashed*, a programmer's guide and reference to the C# (pronounced "C sharp") programming language. C# is primarily an object-oriented programming language, created at Microsoft, which emphasizes a component-based approach to software development. In its third version, C# is still evolving, and this book guides you on a journey of learning how that evolution helps you accomplish more in your software engineering endeavors.

C# is one of several languages of the .NET (pronounced "dot net") platform, which includes a runtime engine called the Common Language Runtime (CLR) and a huge class library. The runtime is a virtual machine that manages code and provides several other services. The class library includes literally thousands of reusable objects and supports several user interface technologies for both desktop and Web Application development.

C# is evolving as a programming language. It began life as an object-oriented, component-based language but now is growing into areas that were once considered the domain of functional programming languages. Throughout this book, you'll see examples of objects and components being used as building blocks for applications. You'll also see many examples that include Language Integrated Query (LINQ), which is a declarative way to query data sources, whether the data source is in the form of objects, relational, XML, or any other format.

Just as C# (and the .NET platform) has evolved, so has this book. *C# Unleashed* began as a language-centric learning guide and reference for applying the C# programming language. The audience was varied because C# was new and developers from all types of backgrounds were

programming with it. All the applications compiled on the command line, and all you needed was the .NET Framework SDK and an editor to do everything.

At its essence, the same concepts driving the first version of this book made it into this version. For example, you don't need to already know .NET before getting started. If you've programmed with any programming language, *C# 3.0 Unleashed* should be an easy on-ramp for you. This book contains a few command-line examples, especially in the beginning, because I believe that using the command line is a skill that is still necessary and useful. However, I quickly move to the Visual Studio 2008 (VS2008) Integrated Development Environment (IDE) for the largest share of the rest of the book. You aren't required to use VS2008, however; I show you right away how to build your applications without it, and Appendix A, "Compiling Programs," is a guide to command-line options with examples (just like the first version of *C# Unleashed*). However, VS2008 is an incredible tool for increasing productivity, and I provide tips throughout this book for cranking out algorithms with code-focused RAD.

In addition to coverage of VS2008, I've included several new chapters for the newest technologies, such as Windows Presentation Foundation (WPF), Windows Communication Foundation (WCF), and AJAX. If you like the cutting edge, there are chapters on the ADO.NET Entity Framework and ADO.NET Data Services. Speaking of data, I've added an entire part of this book with multiple chapters on working with data.

Since July 2000, when I cracked open the first public pre-beta release of .NET, I've been hooked, with C# as my language of choice. I've made a good living and found my C# skills in demand, even in a difficult economy. Most of all, I've gained an enormous amount of experience in both teaching, as a formal course instructor, and as a developer, delivering value to customers with an awesome toolset. I hope that all the gotchas, tips, and doses of reality that I've encountered and shared in this book will help you learn and thrive as I have.

## Why This Book Is for You

If you've developed software in any other computer programming language, you will be able to understand the contents of this book with no trouble. You already know how to make logical decisions and construct iterative code. You also understand variables and basic number systems such as hexadecimal. Novices may want to start with something at the introductory level, such as *Sams Teach Yourself C# in 21 Days*. Honestly, ambitious beginners could do well with this book if they're motivated.

This is a book written for any programmer who wants to learn C# and .NET. It's basic enough for you to see every aspect of C# that's possible, yet it's sufficiently advanced to provide insight into the modern enterprise-level tasks you deal with every day.

## Organization and Goals

*C# 3.0 Unleashed* is divided into eight parts. To promote learning from the beginning, it starts with the simpler material and those items strictly related to the C# language itself. Later, the book moves into other C#-related areas, showing how to use data, user interface technologies, web services, and other useful .NET technologies.

Part 1 is the beginning, covering basic C# language syntax and other essentials. Chapter 1 starts you off by discussing the .NET platform. This is an important chapter because you need to know the environment that you are building applications for. It permeates everything else you do as a C# developer and should be a place you return to on occasion to remind yourself of the essential ingredients of being a successful C# developer. In Chapter 2, you learn how to build a simple C# application using both the command line and VS2008. It is just the beginning of much VS2008 coverage to come. Chapter 3 is another essential milestone for success in developing .NET applications with C#, learning the type system. Chapters 4 and 5 show you how to work with strings and arrays, respectively. By the time you reach Chapter 7, you'll have enough skills necessary to write a simple application and encounter bugs. So, I hope you find my tips on using the VS2008 debugger helpful before moving on to more complexity with object-oriented programming in Part 2.

Part 2 covers object and component programming in C#. In the first version of *C# Unleashed*, I dedicated an entire chapter to basic object-oriented programming concepts. What changed in *C# 3.0 Unleashed* is that I weaved some of those concepts into other chapters. This way, developers who already know object-oriented programming don't have to skip over an entire chapter, but those who don't aren't completely left out. Mostly, I concentrate on how C# implements object-oriented programming, explaining those nuances that are of interest to existing object-oriented programmers and necessary for any C# developer.

Part 3 teaches you some of the more advanced features of C#. With an understanding of objects from Part 2, you learn about object lifetime—when objects are first instantiated and when they are cleaned up from memory. An entire body of knowledge builds upon earlier chapters, leading to where you need to be to understand .NET memory management, the Garbage Collector, what it means for you as a C# developer, and mostly, what you can do to ensure that your objects and the resources they work with are properly managed.

Part 4 gives you five chapters of data. Feedback from the first version of this book indicated that you wanted more. So, now you can learn about LINQ to Objects, LINQ to SQL, ADO.NET, LINQ to DataSet, XML, LINQ to XML, ADO.NET Entity Framework, LINQ to Entities, ADO.NET Data Services, and LINQ to Data Services. Really, five chapters aren't the end of the story, and there is good reason why I moved data earlier in the book: I use LINQ throughout the rest of the book. In addition to learning how to use all of these data access technologies, you'll see many examples in the whole book.

Part 5 demonstrates how to use various desktop user interface technologies. You have choices, console applications, which were beefed up in .NET 2.0, Windows Forms, and WPF. By the way, if you are interested in Silverlight, you'll want to read the WPF chapter

first because both technologies use XAML, the same layout, and the same control set. Not only does it help me bring more information to you on these new technologies, but it also should be comforting that what you learn with one technology is useful with another, expanding your skill set as a .NET developer.

Part 6 teaches you how to build web user interfaces. ASP.NET is the primary web UI technology for .NET today, and I provide a fair amount of coverage to help you get up-to-speed with it. You'll want to pay attention to the discussion of the difference between desktop and web applications because it affects how you develop ASP.NET applications. In recent years, Asynchronous JavaScript and XML (AJAX) has become a hot topic. I show you how to use ASP.NET AJAX, which ships with VS2008, to make your ASP.NET pages more responsive to the user. The newest web UI technology is Silverlight, which enables you to build interactive websites that were once only possible with desktop UI technologies. A couple of the new capabilities of Silverlight are easier ways to play audio and video on the web and animation; these new capabilities allow you to build web experiences similar to Adobe Flash.

Part 7 brings you in touch with various communications technologies. In a connected world, these chapters teach you how to use essential tools. You learn how to use TCP/IP, HTTP, and FTP, and send email using .NET Framework libraries. The remoting chapter is still there, as is the web services chapter. However, an additional chapter covers the new WCF web services.

Part 8 covers topics in architecture and design. Many programmers learn C# and all the topics discussed previously and then find their own way to build applications with what they've learned. If they find an effective way to build applications, then that is positive. However, it's common for people to want to know what the best way is for putting together all of these objects, components, and services to build a usable application. I don't have all the answers because architecture and design is a big topic, and there are as many opinions about it as there are questions. However, I've taken a quick foray into the subject, showing you some of the techniques that have worked for me. You learn how C# and .NET support common design patterns and make it easy for you to use these patterns. I show you how to build an n-layered application and describe a couple more ways that you can take what I've presented and use it in your own way. I also show you how to use a couple .NET tools, including the Class Designer, and introduce you to Windows Workflow (WF), which has a graphical design surface for building applications graphically.

Part 9 is a grab bag of technologies that could be important to your development, depending on what you want to do. For example, multithreading is something that most programmers will do on occasion. However, multithreading is a skill that most programmers will need as multiprocessing and multicore CPUs become more common, meaning that I added more multiprocessing/multithreaded information in this version of the book. Depending on where you are in the world, localization and globalization could be very important, so I explain the essentials of resources and satellite assemblies for localization

purposes. There is still a lot of legacy code that people need to communicate with, depending on the needs of the project you are working on. To help out, the chapter on Interop covers P/Invoke for interoperating with Win32 DLLs and COM Interop for working with COM. There's also some information on working with COM+. For those of you who like a solution out of the box, I explain how to use the .NET trace facilities for instrumenting and logging. There's also a section on how to use existing performance counters and how to instrument your own code with a custom performance counter for diagnostics through the Windows Performance Monitor.

Part 10 helps you with your ultimate goal: deploying code. This is a series of quick chapters to help you build setup programs and deploy desktop or web applications. Before that, I give you some more information about assemblies and what they are made of. The Security chapter will help you learn how the .NET Code Access Security (CAS) system works. Along the way, I throw in several tips to ensure that your deployment endeavors go more smoothly than if you would have had to do it alone.

That's what this book is all about. I wish you luck in learning C# and hope that you find *C# 3.0 Unleashed* a helpful learning tool and useful reference.

# CHAPTER 1

## Introducing the .NET Platform

As a C# developer, it's important to understand the environment you are building applications on: Microsoft .NET (pronounced "Dot Net"). After all, your design and development decisions will often be influenced by code-compilation practicalities, the results of compilation, and the behavior of applications in the runtime environment. The foundation of all .NET development begins here, and throughout this book I occasionally refer back to this chapter when explaining concepts that affect the practical implementation of C#.

By learning about the .NET environment, you can gain an understanding of what .NET is and what it means to you. You learn about the parts of .NET, including the Common Language Runtime (CLR), the .NET Framework Class Library, and how .NET supports multiple languages. Along the way, you see how the parts of .NET tie together, their relationships, and what they do for you. First, however, you need to know what .NET is, which is explained in the next section.

### What Is .NET?

Microsoft .NET, which I refer to as just .NET, is a platform for developing "managed" software. The word *managed* is key here—a concept setting the .NET platform apart from many other development environments. I'll explain what the word *managed* means and why it is an integral capability of the .NET platform.

When referring to other development environments, as in the preceding paragraph, I'm focusing on the traditional

### IN THIS CHAPTER

- ▶ What Is .NET?
- ▶ The Common Language Runtime (CLR)
- ▶ The .NET Framework Class Library (FCL)
- ▶ C# and Other .NET Languages
- ▶ The Common Type System (CTS)
- ▶ The Common Language Specification (CLS)



practice of compiling to an executable file that contains machine code and how that file is loaded and executed by the operating system. Figure 1.1 shows what I mean about the traditional compilation-to-execution process.



FIGURE 1.1 Traditional compilation.

In the traditional compilation process, the executable file is binary and can be executed by the operating system immediately. However, in the managed environment of .NET, the file produced by the compiler (the C# compiler in our case) is not an executable binary. Instead, it is an assembly, shown in Figure 1.2, which contains metadata and intermediate language code.



FIGURE 1.2 Managed compilation.

As mentioned in the preceding paragraph, an assembly contains intermediate language and metadata rather than binary code. This intermediate language is called Microsoft Intermediate Language (MSIL), which is commonly referred to as IL. IL is a high-level, component-based assembly language. In later sections of this chapter, you learn how IL supports a common type system and multiple languages in the same platform.

## .NET STANDARDIZATION

.NET has been standardized by both the European Computer Manufacturers Association (ECMA) and the Open Standards Institute (OSI). The standard is referred to as the Common Language Infrastructure (CLI). Similarly, the standardized term for IL is Common Intermediate Language (CIL).

In addition to .NET, there are other implementations of CIL—the two most well known by Microsoft and Novell. Microsoft's implementation is an open source offering for the purposes of research and education called the Shared Source Common Language Infrastructure (SSCLI). The Novell offering is called Mono, which is also open source.

Beyond occasional mention, this book focuses mainly on the Microsoft .NET implementation of the CLI standard.

The other part of an assembly is metadata, which is extra information about the code being used in the assembly. Figure 1.3 shows the contents of an assembly.

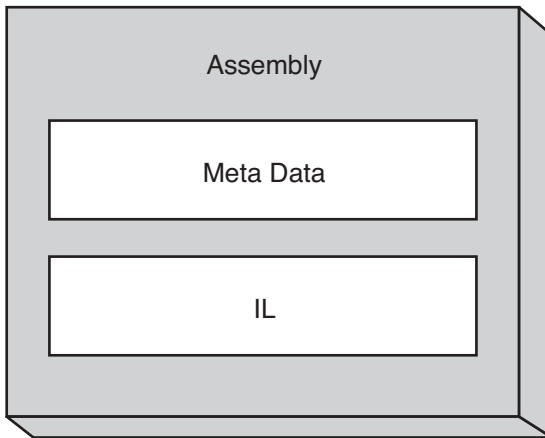


FIGURE 1.3 Assembly contents.

Figure 1.3 is a simplified version of an assembly, showing only those parts pertaining to the current discussion. Assemblies have other features that illustrate the difference between an assembly and an executable file. Specifically, the role of an assembly is to be a unit of deployment, execution, identity, and security in the managed environment. In Part X, Chapters 43 and 44 explain more about the role of the assembly in deployment, identity, and security. The fact that an assembly contains metadata and IL, instead of only binary code, has a significant advantage, allowing execution in a managed environment. The next section explains how the CLR uses the features of an assembly to manage code during execution.

## The Common Language Runtime (CLR)

As introduced in the preceding section, C# applications are compiled to IL, which is executed by the CLR. This section highlights several features of the CLR. You'll also see how the CLR manages your application during execution.

### Why Is the CLR Important?

In many traditional execution environments of the past, programmers needed to perform a lot of the low-level work (plumbing) that applications needed to support. For example, you had to build custom security systems, implement error handling, and manage memory.

The degree to which these services were supported on different language platforms varied considerably. Visual Basic (VB) programmers had built-in memory management and an error-handling system, but they didn't always have easy access to all the features of COM+, which opened up more sophisticated security and transaction processing. C++ programmers have full access to COM+ and exception handling, but memory management is a totally manual process. In a later section, you learn about how .NET supports multiple

languages, but knowing just a little about a couple of popular languages and a couple of the many challenges they must overcome can help you to understand why the CLR is such a benefit for a C# developer.

The CLR solves many problems of the past by offering a feature-rich set of plumbing services that all languages can use. The features described in the next section further highlight the value of the CLR.

**CLR Features**

This section describes, more specifically, what the CLR does for you. Table 1.1 summarizes CLR features with descriptions and chapter references (if applicable) in this book where you can find more detailed information.

TABLE 1.1    CLR Features

Feature	Description
.NET Framework Class Library support	Contains built-in types and libraries to manage assemblies, memory, security, threading, and other runtime system support
Debugging	Facilities for making it easier to debug code. (Chapter 7)
Exception management	Allows you to write code to create and handle exceptions. (Chapter 11)
Execution management	Manages the execution of code
Garbage collection	Automatic memory management and garbage collection (Chapter 15)
Interop	Backward-compatibility with COM and Win32 code. (Chapter 41)
Just-In-Time (JIT) compilation	An efficiency feature for ensuring that the CLR only compiles code just before it executes
Security	Traditional role-based security support, in addition to Code Access Security (CAS) (Chapter 44)
Thread management	Allows you to run multiple threads of execution (Chapter 39)
Type loading	Finds and loads assemblies and types
Type safety	Ensures references match compatible types, which is very useful for reliable and secure code (Chapter 4)

In addition to the descriptions provided in Table 1.1, the following sections expand upon a few of the CLR features. These features are included in the CLR execution process.

**The CLR Execution Process**

Beyond just executing code, parts of the execution process directly affect your application design and how a program behaves at runtime. Many of these subjects are handled throughout this book, but this section highlights specific additional items you should know about.

From the time you or another process selects a .NET application for execution, the CLR executes a special process to run your application, shown in Figure 1.4.

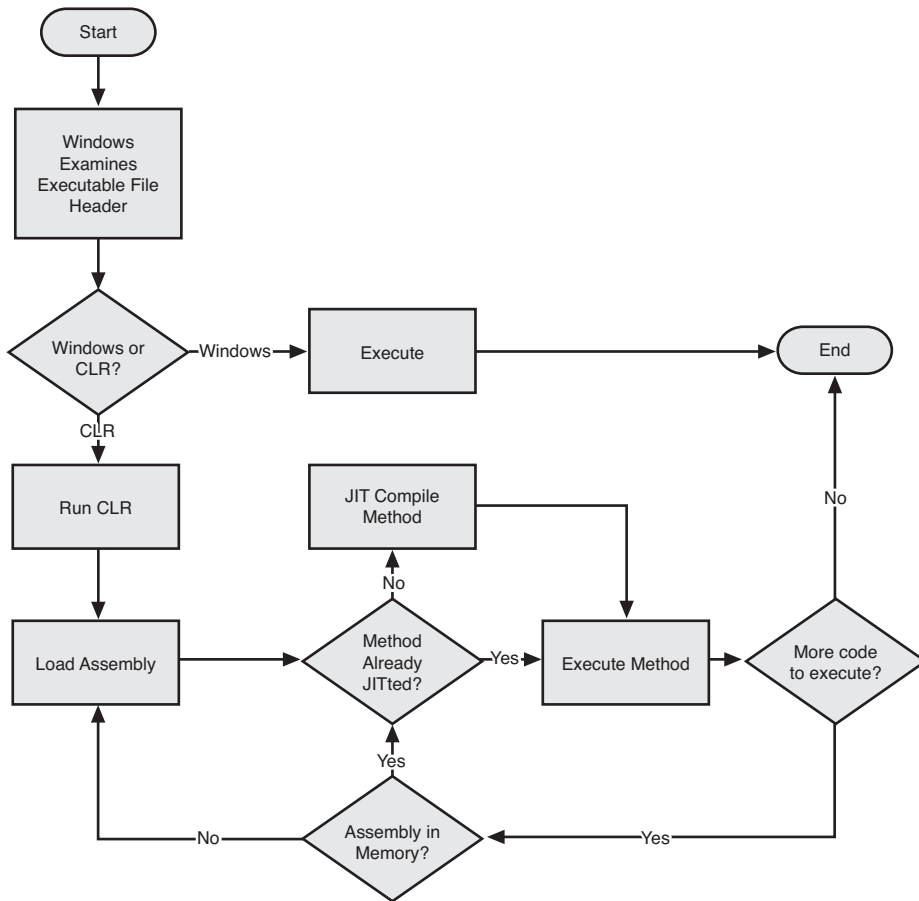


FIGURE 1.4 The CLR execution process (summarized).

As illustrated in Figure 1.4, Windows (the OS) will be running at Start; the CLR won't begin execution until Windows starts it. When an application executes, OS inspects the file to see whether it has a special header to indicate that it is a .NET application. If not, Windows continues to run the application.

If an application is for .NET, Windows starts up the CLR and passes the application to the CLR for execution. The CLR loads the executable assembly, finds the entry point, and begins its execution process.

The executable assembly could reference other assemblies, such as dynamic link libraries (DLLs), so the CLR will load those. However, this is on an as-needed basis. An assembly won't be loaded until the CLR needs access to the assembly's code. It's possible that the

code in some assemblies won't be executed, so there isn't a need to use resources unless absolutely necessary.

As mentioned previously, the C# compiler produces IL as part of an assembly's output. To execute the code, the CLR must translate the IL to binary code that the operating system understands. This is the responsibility of the JIT compiler.

As its name implies, the JIT compiler only compiles code before the first time that it executes. After the IL is compiled to machine code by the JIT compiler, the CLR holds the compiled code in a working set. The next time that the code must execute, the CLR checks its working set and runs the code directly if it is already compiled. It is possible that the working set could be paged out of memory during program execution, for various reasons that are necessary for efficient operation of the CLR on the particular machine it is running on. If more memory is available than the size of the working set, the CLR can hold on to the code. Additionally, in the case of Web applications where scalability is an issue, the working set can be swapped out due to periodic recycling or heavier load on the server, resulting in additional load time for subsequent requests.

The JIT compiler operates at the method level. If you aren't familiar with the term *method*, it is essentially the same as a function or procedure in other languages. Therefore, when the CLR begins execution, the JIT compiler compiles the entry point (the *Main* method in C#). Each subsequent method is JIT compiled just before execution. If a method being JIT compiled contains calls to methods in another assembly, the CLR loads that assembly (if not already loaded).

This process of checking the working set, JIT compilation, assembly loading, and execution continues until the program ends.

The meaning to you in the CLR execution process is in the form of application design and understanding performance characteristics. In the case of assembly loading, you have some control over when certain code is loaded. For example, if you have code that is seldom used or necessary only in specialized cases, you could separate it into its own DLL, which will keep the CLR from loading it when not in use. Similarly, separating seldomly executed logic into a separate method ensures the code doesn't JIT until it's called.

Another detail you might be concerned with is application performance. As described earlier, code is loaded and JIT compiled. Another DLL adds load time, which may or may not make a difference to you, but it is certainly something to be aware of. By the way, after code has been JIT compiled, it executes as fast as any other binary code in memory.

One of the CLR features listed in Table 1.1 is .NET Framework Class Library (FCL) support. The next section goes beyond FCL support for the CLR and gives an overview of what else the FCL includes.

## The .NET Framework Class Library (FCL)

.NET has an extensive library, offering literally thousands of reusable types. Organized into namespaces, the FCL contains code supporting all the .NET technologies, such as Windows Forms, Windows Presentation Foundation, ASP.NET, ADO.NET, Windows

Workflow, and Windows Communication Foundation. In addition, the FCL has numerous cross-language technologies, including file I/O, networking, text management, and diagnostics. As mentioned earlier, the FCL has CLR support in the areas of built-in types, exception handling, security, and threading. Table 1.2 shows some common FCL libraries.

## WHAT IS A TYPE?

Types are used to define the meaning of variables in your code. They could be built-in types such as `int`, `double`, or `string`. You can also have custom types such as `Customer`, `Employee`, or `BankAccount`. Each type has optional data/behavior associated with it.

Much of this book is dedicated to explaining the use of types, whether built-in, custom, or those belonging to the .NET FCL. Chapter 4, “Understanding Reference Types and Value Types,” includes a more in-depth discussion on how C# supports the .NET type system.

TABLE 1.2 Common .NET Framework Class Library Namespaces

<code>System</code>	<code>System.Runtime</code>
<code>System.Collections</code>	<code>System.Security</code>
<code>System.Configuration</code>	<code>System.ServiceModel</code>
<code>System.Data</code>	<code>System.Text</code>
<code>System.Diagnostics</code>	<code>System.Threading</code>
<code>System.Drawing</code>	<code>System.Web</code>
<code>System.IO</code>	<code>System.Windows</code>
<code>System.Linq</code>	<code>System.Workflow.*</code>
<code>System.Net</code>	<code>System.Xml</code>

The namespaces in Table 1.2 are a sampling from the many available in the .NET Framework. They’re representative of the types they contain. For example, you can find Windows Presentation Foundation (WPF) libraries in the `System.Windows` namespace, Windows Communication Foundation (WCF) is in the `System.ServiceModel` namespace, and Language Integrated Query (LINQ) types can be found in the `System.Linq` namespace.

Another aspect of Table 1.2 is that I included only two levels in the namespace hierarchy, `System.*`. In fact, there are multiple namespace levels, depending on which technology you view. For example, if you want to write code using the Windows Workflow (WF) runtime, you look in the `System.Workflow.Runtime` namespace. Generally, you can find the more common types at the higher namespace levels.

One of the benefits you should remember about the FCL is the amount of code reuse it offers. As you read through this book, you’ll see many examples of how the FCL forms the basis for code you can write. For example, you learn how to create your own exception

object in Chapter 13, “Naming and Organizing Types with Namespaces,” which requires that you use the Exception types from the FCL. Even if you encounter situations that don’t require your use of FCL code, you can still use it. An example of when you would want to reuse FCL code is in Chapter 17, “Parameterizing Type with Generics and Writing Iterators,” where you learn how to use existing generic collection classes. The FCL was built and intended for reuse, and you can often be much more productive by using FCL types rather than building your own from scratch.

Another important feature of the FCL is language neutrality. Just like the CLR, it doesn’t matter which .NET language you program in—the FCL is reusable by all .NET programming languages, which are discussed in the next section.

## C# and Other .NET Languages

.NET supports multiple programming languages, which are assisted by both the CLR and the FCL. Literally dozens of languages target the .NET CLR as a platform. Table 1.3 lists some of these languages.

TABLE 1.3    Languages Targeting the .NET CLR

A#	Fortran	Phalanger (PHP)
APL	IronPython	Python
C++	IronRuby	RPG
C#	J#	Silverfrost FTN95
COBOL	Jscript	Scheme
Component Pascal	LSharp	SmallScript
Delphi	Mercury	Smalltalk
Delta Forth	Mondrian	TMT Pascal
Eiffel.NET	Oberon	VB.NET
F#	Perl	Zonnon

Table 1.3 is not a comprehensive list because there are new languages being created for .NET on a regular basis. An assumption one could make from this growing list is that .NET is a successful multilanguage platform.

As you learned earlier in this chapter, the C# compiler emits IL. However, the C# compiler is not alone—all compilers for languages in Table 1.2 emit IL, too. By having a CLR that consumes IL, anyone can build a compiler that emits IL and join the .NET family of languages.

In the next section, you learn how the CLR supports multiple languages via a Common Type System (CTS), the relationship of the languages via a Common Language Specification (CLS), and how languages are supported via the FCL.

## The Common Type System (CTS)

To support multiple programming languages on a single CLR and have the ability to reuse the FCL, the types of each programming language must be compatible. This binary compatibility between language types is called the Common Type System (CTS).

The built-in types are represented as types in the FCL. This means that a C# `int` is the same as a VB.NET Integer type and their .NET type is `System.Int32`, which is a 32-bit integer named `Int32` in the `System` namespace of the FCL. You'll learn more about C# types, type classification, and how C# types map to the CTS in Chapter 4.

## The Common Language Specification (CLS)

Although the CLR understands all types in the CTS, each language targeting the CLR will not implement all types. Languages must often be true to their origins and will not lose their features or add new features that aren't compatible with how they are used.

However, one of the benefits of having a CLR with a CTS that understands IL, and an FCL that supports all languages, is the ability to write code in one language that is consumable by other languages. Imagine you are a third-party component vendor and your language of choice is C#. It would be desirable that programmers in any .NET language (for example, IronRuby or Delphi) would be able to purchase and use your components.

For programming languages to communicate effectively, targeting IL is not enough. There must be a common set of standards to which every .NET language must adhere. This common set of language features is called the Common Language Specification (CLS).

Most .NET compilers can produce both CLS-compliant and non-CLS-compliant code, and it is up to the developer to choose which language features to use. For example, C# supports unsigned types, which are non-CLS compliant. For CLS compliance, you can still use unsigned types within your code so long as you don't expose them in the public interface of your code, where code written in other languages can see.

## Summary

.NET is composed of a CLR and the .NET FCL, and supports multiple languages. The CLR offers several features that free you from the low-level plumbing work required in other environments. The FCL is a large library of code that supports additional technologies such as Windows Presentation Foundation, Windows Communication Foundation, Windows Workflow, ASP.NET, and many more. The FCL also contains much code that you can reuse in your own applications. Through its support of IL, a CTS, and a CLS, many languages target the .NET platform. Therefore, you can write a reusable library with C# code that can be consumed by code written in other programming languages.



Remember that understanding the .NET platform, which includes CLR, FCL, and multiple-language support, has implications in the way you design and write your code.

Throughout this book, you'll encounter many instances where the concepts in this chapter lay the foundation of the tasks you need to accomplish. You might want to refer back to this chapter for an occasional refresher.

This chapter has been purposefully as short as possible to cover only the platform issues most essential to building C# applications. If you're like me, you'll be eager to jump into some code. The next chapter does that by introducing you to essential syntax of the C# programming language.

# Index

2D graphics, creating, Graphics object, 536-539

## A

**abstract classes, 288-290**

interface differences, 290

**abstraction, OOP (object-oriented programming), 177**

**access modifiers**

encapsulation, 185

internal access, 187-188

private access, 186

protected access, 187

protected internal access, 188

public access, 185

objects, 188-189

**accessing**

ADO.NET Data Services via HTTP URIs,  
493-498

ASRNET AJAX controls, 628-635

Entity SQL, 480

**Add menu commands, New Item, 189**

**add method, modifying, 266-267, 271**

**Add New Item window (VS2008), 28**

**Add Silverlight Application window, 643**

**adding colors, console applications, 511-514**

**addition operator (+), 53**

**ADO.NET, 14, 441, 459**

architecture, 441-445

components, 441-443

connected modes, 443-444

connections, creating, 445-447

data

manipulating, 450-452

viewing, 447-450

data providers, 444-445

database login security, 446-447

- DataSet object
  - LINQ, 458-459
  - reading data with, 453-454
  - saving modifications to database, 454-457
- disconnected mode, 443-444
- disconnected data, working with, 453-457
- entities
  - creating, 482-486
  - EDM (Entity Data Model), 476-480
  - mapping, 482-483
  - querying with Entity SQL, 480-481
- namespaces, 444-445
- prefixes, 444-445
- providers, 444-445
- stored procedures, calling, 452
- ADO.NET Data Services, 491**
  - accessing via HTTP URLs, 493-498
  - client library, writing code with, 499-504
  - entities
    - adding, 501
    - deleting, 502-503
    - querying, 499-501
    - sorting, 497
    - updating, 501-502
  - entity associations, traversing, 497-498
  - entity items, selecting, 493-495
  - entity results, filtering, 495-497
  - entity sets, viewing, 493
  - projects, adding to, 492-493
  - WebDataGen.exe-generated classes, using, 503-504
- ADO.NET Entity Framework, 476, 489**
  - data abstractions, creating, 475-489
  - entities, 476
- AdRotator server control (ASP.NET), 593**
- advanced options, compilers, 969-970**
- aggregate operators, 439**
- AJAX (Asynchronous JavaScript and XML), 619-620**
  - ASP.NET AJAX web applications, 620-621
    - calling web services, 635-640
    - controls, 625-635
    - life cycles, 621-623
    - loading custom script libraries, 623-625
  - Silverlight, relationship, 642
- alias directive, namespaces, 276-278**
- alias qualifiers, namespaces, 283-284**
- aliases, 95**
- alignment, WPF applications**
  - default alignment, 552
  - explicit alignment, 552-553
- allocation, memory, 328-329**
- animating UI elements, Silverlight, 655-657**
- Anonymous method, 670**
- anonymous methods**
  - assigning, 256-258
  - fields, capturing, 257
  - local variables, capturing, 257
- Another Implementation of the IBroker Interface listing (14.4), 298-301**
- API hierarchy, reflection, 350**
- APIs (application programming interfaces), Reflection.Emit API, 359-363**
- AppDomain execution environment, 696**
- Append method, 123**
- AppendFormat method, 123**
- application domain security policy level, 936**
- Application state (ASP.NET), 596**
- applications**
  - architectures
    - N-Layer, 781-795
    - N-Tier, 781-784
  - console applications, 507
    - adding color to, 511-514
    - command-line input handling, 510-511
    - PasswordGenerator, 508
    - program interaction, 45-48
    - user interaction, 508-510
  - debugging, 147-159
  - desktop applications, deploying, 955-958
  - RAD (rapid application development), 516
    - drag-and-drop problems, 779-781
  - regular expressions, 127-129
  - RIAs (rich internet applications), 583, 641-657
  - VS2008 applications
    - building, 28-31
    - running, 28-31
  - WCF applications, creating, 726-727
  - web applications, 586-588
    - adding interactivity to, 619-620
  - ASPRNET, 588-617

- ASP.NET AJAX web applications, 620-640
- components, 961-962
- model, 583-586
- publishing, 961-966
- Windows Form applications
  - controls, 528-536
  - data binding, 533-536
  - dialog boxes, 539-545
  - files, 520
  - fundamentals, 516-519
  - GDI+, 536-539
  - visual design environment, 519
  - Visual Designer, 521-527
  - VS2008 support, 519-527
  - window communication, 540-543
  - windows, 539-545
- Windows service applications, 679
  - coding, 683-688
  - creating, 680-683
- Workflow application, starting, 797-798
- WPF (Windows Presentation Foundation)
  - applications, 547
    - canvas layout, 553
    - controls, 560-573
    - data binding, 574-578
    - default alignment, 552
    - displaying data lists, 575-578
    - DockPanel layout, 559
    - event handling, 573-574
    - explicit alignment, 552-553
    - Grid layout, 555-559
    - layout management, 551-559
    - StackPanel layout, 554
    - styles, 578-580
    - UniformGrid layout, 555
    - WrapPanel layout, 553-554
    - XAML, 548-551
- architectures**
  - ADO.NET, 441-445
  - applications
    - N-Layer, 781-795
    - N-Tier, 781-784
- arithmetic operators, 52**
  - addition operator (+), 53
  - division operator (/), 52
  - modulus operator (%), 53
  - multiplication operator (\*), 52
  - subtraction operator (-), 53
- arrays, 131-134**
  - array bounds, 137-138
  - generic collections, compared, 371-372
  - initializing, 132
  - jagged arrays, 135-137
  - multidimension arrays, 134-135
  - nongeneric collections, compared, 371-372
  - searching, 138-139
  - single-dimension arrays, 132-134
  - sorting, 138-139
  - System.Array class, 137
    - array bounds, 137-138
    - searching, 138-139
    - sorting, 138-139
- as operator, 60-61**
- ASMX web services, 713**
  - basic web services, 714-716
  - information, viewing, 716-719
  - technologies, 713-714
  - using, 719-723
- ASP.NET, 14**
  - benefits of, 586
  - controls, 593
    - HTML controls, 595-596
    - Menu control, 605
    - Server controls, 593, 595
  - cookies, issuing, 598
  - data binding, 614-617
  - navigation, 603-609
  - page life cycle, code, 590-593
  - page requests, high-level view of, 584
  - pages, 588-593
  - Silverlight, relationship, 642
  - SiteMapPath, 609
  - sites, theming, 609-612
  - state management, 596-603
  - TreeView, implementing, 606-608
  - web applications
    - components, 961-962
    - creating, 586-588
    - model, 583-586
  - web services, 713
    - basic web services, 714-716
    - technologies, 713-714

- using, 719-723
- viewing information, 716-719
- web.sitemap, site layout, 604
- websites, securing, 612-613

**ASP.NET AJAX**

- web applications, 619-620
  - controls, 625-635
  - life cycles, 621-623
  - loading custom script libraries, 623-625
  - setting up, 620-621
- web services, calling, 635-640

**assemblies, 10, 832, 921-922**

- attributes, 923-924
- CLR (Common Language Runtime), 11-14
- code groups, 935-936
- compilers, 971
- concurrent garbage collection, 928
- configuration, 927-930
- deployment, 930
- executable files, compared, 11
- GAC (global assembly cache), 930
- identity, 925
- IL (Intermediate Language), 10
- location elements, 929-930
- manifests, 922
- metadata, 10
- runtime assemblies, building, 359-363
- saving, 362
- scope, 925
- security, 926-927
- version redirection, 928-929
- versioning, 925-926

**Assembly names, verifying, 700****assembly reference, namespaces, compared, 277-278****Assembly type, 356****Assertion Demonstration: AssertDemo.cs listing (42.11), 887****assertions, making, 886-887****assigning anonymous methods, 256-258****assignments**

- reference type, 88-91
- value type, 91-92

**assignment operators, 59****association relationships, creating, VS2008 Class Designer, 753****associations, observing, VS2008 Class Designer, 747-748****associativity, operators, 64-65****Asynchronous JavaScript and XML (AJAX), 619****ATMs (automated teller machines), interfaces, 287-288****attaching to processes, debugging, 156-159****attachments, email, sending, 676****Attribute Targets listing (16.5), 344****attributes, 339, 363**

- assemblies, 923-924
- AttributeUsage, 345-349
- classes, obtaining from, 356-357
- CLSCompliant, 344
- common naming conventions, 348
- conditional attributes, debugging with, 881-882
- creating, 345-349
- field requirements, 348
- Flags, 341
- inheritance, 349
- multiple attributes, using, 341-342
- multiplicity, controlling, 349
- Obsolete, 343
- OperationContract attributes, declaring, 730
- parameters, 342
  - named parameters, 343
  - positional parameters, 342-343
- property requirements, 348
- reflection, 356-363
- ServiceContract attribute, declaring, 730
- StructLayout, 343
- targets, 344-348
  - using, 340-342

**AttributeUsage attribute, 345-349****auto-implemented properties, objects, 168****autogenerated web service proxies, creating, 719-721****Autogenerated Web Service Proxy:**

- BasicWebService.cs listing (33.3), 720-721

**automatic expression conversions, 220****automatic memory management, objects, 328-331****automatic promotions, handling, 220****Autos window (VS2008), 151**

## B

### base classes

- constraints, 385-386
- Contact, 190
- inheritance, 178-180
- members
  - calling, 179-180
  - hiding, 180-181

**Basic Remoting Client: BasicRemotingClient.cs**  
listing (32.3), 699-700

**behavior element, web services, 736-737**

**Berners-Lee, Tim, 583**

### binary operators, 52

- addition operator (+), 53
- division operator (/), 52
- modulus operator (%), 53
- multiplication operator (\*), 52
- subtraction operator (-), 53

**BinarySearch method, 138-139**

**binding ComboBox, 576**

**bitwise AND operator (&), 56**

**bitwise complement operator (~), 52**

**bitwise exclusive OR operator (^), 57**

**bitwise inclusive OR operator (|), 57**

### blocks

- catch blocks, exception handling, 236-237
- code, 63
- try/catch block, exception handling, 232-234
- try/catch blocks, 667

**bool type, 39**

**Boolean AND operator (&), 57**

**Boolean exclusive OR operator (^), 58**

**Boolean inclusive OR operator (|), 57**

### Boolean logical operators, 56

- bitwise AND operator (&), 56
- bitwise exclusive AND operator (&), 57
- bitwise exclusive OR operator (^), 57
- bitwise inclusive OR operator (&), 57
- Boolean exclusive OR operator (^), 58
- Boolean inclusive OR operator (|), 57
- conditional AND operator (&&), 58
- conditional OR operator (||), 58-59
- side effects, 59

**Boolean switches, debugging, implementing with, 883-884**

**BooleanSwitch entry in Configuration File: BooleanSwitchDemo.config listing (42.6), 884**

**Border control (WPF), 560-561**

**boxing value type variables, 83-85**

**break statements, 75-76**

**breakpoints, code, 158**

setting, 148-149

**Brush object (GDI+), Windows Forms applications, 536-537**

**Build Menu (VS2008), 28**

**Build Project command (Build menu), 29**

**Build Solution command (Build menu), 29**

### building

- generic types, 372-387
- hierarchies, group joins, 414
- runtime assemblies, Reflection.Emit API, 359-363
- VS2008 applications, 28-31

**built-in performance counters, accessing, 888-895**

**built-in types, 15**

**BulletedList server control (ASP.NET), 593**

**Button control (Windows Forms), 528**

**Button control (WPF), 561**

**Button server control (ASP.NET), 593**

## C

### C#

- aliases, 95
- arrays, 131-132
  - array bounds, 137-138
  - jagged arrays, 135-137
  - multidimension arrays, 134-135
  - searching, 138-139
  - single-dimension arrays, 132-134
  - sorting, 138-139
  - System.Array class, 137-139
- case sensitivity, 22
- code
  - blocks, 63
  - commenting, 32-35

- conventions, 37
- definite assignment, 44
- identifiers, 35-36
- keywords, 36-37
- layout, 37
- locating, 584
- scope, 63
- style, 37
- types, 39-44
- variables, 38-39
- enums, 139-142
  - System.Enum struct, 142-145
- labels, 63-64
- loops, 69
  - do loops, 71-72
  - for loops, 72-73
  - foreach loops, 73
  - while loops, 70
- operators, 49
  - as operator, 60-61
  - assignment operators, 59
  - associativity, 64-65
  - binary operators, 52-54
  - checked operator, 62
  - compound assignment operators, 59
  - is operator, 60
  - logical operators, 56-59
  - precedence, 64-65
  - relational operators, 54-56
  - sizeof operator, 61
  - ternary operator, 60
  - typeof operator, 61
  - unary operators, 49-52
  - unchecked operator, 62
- simple programs, writing, 19-23
- statements, 62
  - break statements, 75-76
  - continue statements, 76
  - goto statements, 74-75
  - if statements, 65-67
  - return statements, 76-77
  - switch statements, 67-69
- types
  - .NET framework types, 94-101
  - nullable types, 102-103
  - string, 105-122

- Unified Type System, 80-81
  - boxing, 83-85
  - object type, 82-83
  - unboxing, 83-85
- C# Component Configured for JIT Activation:**
  - CPJit.cs listing (41.18), 874
- C# Component Exposed as a COM Component:**
  - CallFromCom.dll listing (41.14), 869-870
- C# handler, Silverlight, adding to, 648-649**
- C# Program Calling a COM Component:**
  - TalkToCom.cs listing (41.12), 867-868
- C# Program in VS2008 listing (2.2), 27**
- C++ COM Component: ComObj.dll**
  - listing (41.11), 867
- C++ Programming Style, 190**
- Cache state (ASP.NET), 596-598**
- Calendar server control (ASP.NET), 593**
- calling**
  - base member classes, 179-180
  - stored procedures
    - ADO.NET, 452
    - LINQ to SQL, 420-421
  - web services
    - ASP.NET AJAX, 635-640
    - client code, 738-739
- Calling a C# Component Exposed as a COM Component listing (41.15), 870**
- canvas layout, WPF (Windows Presentation Foundation), 553**
- Cargill, Tom, 190**
- case sensitivity, C#, 22**
- Caspol.exe security utility, 944**
- catch blocks, exception handling, 236-237**
- Cert2spc.exe security utility, 944**
- certificates, assemblies, 926-927**
- Certmgr.exe security utility, 944**
- channels**
  - HttpChannel, 708
  - remoting, 706-708
  - TCP channels, 708
- character classes, regular expressions, 126**
- characters, strings, working with, 119-121**
- CheckBox control (Windows Forms), 528**
- CheckBox control (WPF), 561**
- CheckBox server control (ASP.NET), 593**

- CheckBoxList** server control (ASP.NET), 593
- checked operator**, 62
- checked statements**, 245-248
- checked Statements**: checked.cs listing (11.7), 245-246
- CheckBoxList** control (Windows Forms), 528
- Chktrust.exe** security utility, 944
- Class Designer** (VS2008), 743, 754
  - code, visualizing, 743-748
  - object models, creating, 749-754
- class libraries**, Silverlight, 644
  - files, 646-647
- class library** documentation, .NET Framework, 974
- class members**, GC (garbage collector), 335
- Class to Reflect Upon**: Reflected.cs listing (16.7), 350-352
- classes**
  - abstract classes, 288-290
    - interface differences, 290
  - attributes, obtaining from, 356-357
  - base classes
    - Contact, 190
    - inheritance, 178-180
  - Console, 23, 45
  - Customer, 190-191
  - Debug, 879-881, 917
  - interfaces
    - explicit implementation, 304-310
    - implicit implementation, 293-304
  - members, qualification, 179
  - Monitor, 827-828
  - Process, 818-823
  - ResourceManager, 833-834
  - sealed classes, inheritance, 183-184
  - SiteOwner, 190-192
  - static classes, objects, 171-172
  - StringBuilder, 122-123
    - Append method, 123
    - AppendFormat method, 123
    - EnsureCapacity method, 123-124
    - ToString method, 124
  - System.Array class, 137
    - array bounds, 137-138
    - searching, 138-139
    - sorting, 138-139

- System.Object, 175
- System.Object class, 172-175
- System.Object class members, overriding, 197-200
- Trace, 879
- WebDataGen.exe-generated classes, using, 503-504
  - wrapped HTTP classes, 661
- Clean Project** (Build menu), 29
- Clean Solution** command (Build menu), 29
- CLI** (Common Language Infrastructure), 10
- ClickOnce**, 959
  - configuring, 957-958
  - desktop applications, deploying, 955-957
- client code**
  - iterator pattern, using in, 763
  - web services, calling, 738-739
- Client Code Calling a Web Service** listing (34.3), 738-739
- Client Configuration File**: BasicRemotingClient.exe.config listing (32.4), 700
- client library** (ADO.NET Data Services), writing code with, 499-504
- Client Using Data from Custom Performance Counter**: CustomOrderClient.cs listing (42.15), 897-902
- clients**
  - HTTP clients, creating, 669-671
  - socket clients, creating, 665-669
- cloning objects**, 173-174
- closing diagrams**, 747
- CLR** (Common Language Runtime), 9-14, 17, 79
  - debugging, 12
  - exception management, 12
  - execution management, 12
  - execution process, 12-14
  - features, 12
  - garbage collection, 12
  - importance of, 11-12
  - interop, 12
  - JIT (Just-In-Time) compilation, 12-14
  - namespaces, 274
  - security, 12
  - stacks, 87
    - unwinding, 239
  - string handling, intern pools, 121-122



- thread management, 12
- type loading, 12
- type safety, 12

## **CLS (Common Language Specification), 16-17**

### **CLSCompliant attribute, 344**

#### **code**

- ASP.NET page life cycle, 590-593
- assertions, making, 886-887
- blocks, 63
- breakpoints, 158
  - setting, 148-149
- built-in performance counters, accessing, 888, 893-895
- C# code, locating, 584
- client code
  - iterator pattern, 763
  - web service calls, 738-739
- code groups, membership, 936
- code-based security, 933-934
  - code groups, 935-936
  - evidence, 934
  - permission requests, 937-940
  - permissions, 934-935
  - security policy implementation, 940-942
  - security policy levels, 936-937
- commenting, 32-35
- comments
  - multiline comments, 32
  - single-line comments, 32-33
  - XML documentation comments, 33-35
- conventions, 37
- copying, 623
- customized performance counters, building, 897, 902-907
- debug tracing, runtime tracing, 884-886
- debugging
  - Boolean switches, 883-884
  - conditional debugging, 881-884
  - runtime debugging, 879-881
- definite assignment, 44
- dynamically activating, 357-359
- executing CLR (Common Language Runtime), 12-14
- identifiers, 35-36
- keywords, 36-37
- layout, 37

- managed code, 854
- managed compilation, 10
- organizing namespaces, 274
- performance analysis, sampling, 908, 913, 916-917
- programming state, examining, 149-150
- proxies, 720
- reusing, FCL, 15-16
- role-based security, 942
  - GenericPrincipal, 943-944
  - WindowsPrincipal, 942-943
- runtime trace facilities, 879
- safe code, 854
- scope, 63
- stepping through, 147-159
- style, 37
- threads, running in, 824
- timers, implementing, 896-897
- traditional compilation, 10
- types, 39-44
  - bool type, 39
  - floating-point types, 42-43
  - integral types, 39-41
  - string type, 43-44
- unmanaged code, 854
- unsafe code, 853-854
  - configuring, 61
  - fixed statement, 861-864
  - pointers, 855-858
  - recognizing, 854
  - sizeof() operator, 858-859
  - stackalloc operator, 860-861
- variables, 38-39
- visualizing, VS2008 Class Designer, 743-748
- WF (Windows Workflow) runtime, writing in, 15
- writing, ADO.NET Data Services, 499-504

#### **code groups**

- code-based security, 935-936
- membership, 936

#### **code-based security, 933-934**

- code groups, 935-936
- evidence, 934
- permission requests, 937-940
- permissions, 934-935
- security policy, implementing, 940-942
- security policy levels, 936-937

**Code-Behind Demonstrating Page Life Cycle listing (27.2), 592**

**Code-Behind File listing (26.2), 549-550**

#### **coding**

- events, 258-267, 271
- LINQ to Entities, 486-489
- VS2008, 27-28
- Windows service applications, 683-688

#### **coding methods, 201-202**

- defining, 202-203
- local variables, 203-204
- overloading, 210-212
- parameters, 204
  - out parameters, 207-209
  - params parameters, 209-210
  - ref parameters, 205-207
  - value parameters, 204-205

**Collect class member (GC), 335**

**collection process, GC (garbage collector), 330**

**collection types, 372**

**Collection Without IEnumerable/IEnumerator Interfaces listing (36.1), 764-766**

**collections, .NET Framework, 757**

**colors, console applications, adding to, 511-514**

#### **COM components, 853**

- .NET components, exposing as, 869
- early-bound COM component calls, 866-868
- communicating with from .NET, 866-869
- late-bound COM component calls, 868-869

#### **COM+ services, 876**

- .NET support, 871-876
- JIT (Just-In-Time) activation, 874
- object pooling, 875-876
- transactions, 873-874

**COM+ Services Object Pooling Implemented in C#: CPPool.cs listing (41.19), 875-876**

**COM+ Transactional Component in C#: CPTrans.cs listing (41.17), 873-874**

**CombinePath method, 229**

**ComboBox control (Windows Forms), 528**

**ComboBox control (WPF), 561**

- binding, 576

**comma-separated values (CSVs), strings, 118**

**command-line communications, 45-46**

**command-line input, handling, console applications, 510-511**

**command-line options, VS2008, 46**

**commands, Add menu, New Item, 189**

**comments, code, 32-35**

- multiline comments, 32
- single-line comments, 32-33
- XML documentation comments, 33-35

**common dialog boxes, Windows Forms applications, 543-545**

**Common Language Infrastructure (CLI), 10**

**Common Language Runtime (CLR), 9, 79**

**Common Language Specification (CLS), 16**

**common naming conventions, attributes, 348**

**Common Type System (CTS), 16**

#### **communication**

- objects, 177
- workflows, hosts, 805-810

**communications, windows, Windows Forms applications, 540-543**

**Compare method, 109-110**

**CompareOrdinal method, 109-110**

**CompareValidator server control (ASP.NET), 593**

#### **comparing**

- object references, 172
- strings, 109-110

#### **compilation**

- JIT (Just-In-Time) compilation, CLR (Common Language Runtime), 12
- managed compilation, 10
- programs, 969-971
- traditional compilation, 10
- VS2008 projects, 30

**Compilation Instructions for Chapter 12 listings listing (12.8), 271**

**Compilation Instructions for Listing 41.1 listing (41.2), 858**

**Compilation Instructions for Listing 41.3 listing (41.4), 859**

**Compilation Instructions for Listing 41.5 listing (41.6), 861**

**Compilation Instructions for Listing 41.7 listing (41.8), 864**

**Compilation Instructions for Listing 41.9 listing (41.10), 866**

**Compilation Instructions for Listing 42.1 listing (42.2), 881**

**Compilation Instructions for Listing 42.11 listing (42.12), 887**

**Compilation Instructions for Listing 42.3 listing (42.4), 883**

**Compilation Instructions for Listing 42.5 listing (42.7), 884**

**Compilation Instructions for Listing 42.8 listing (42.10), 886**

**compiler**

- advanced options, 969-970
- assemblies, 971
- warnings, handling, 194

**ComplexNumber, 94**

**components**

- ADO.NET, 441-443
- COM components, .NET, 866-869

**compositional relationships, objects, 177**

**compound assignment operators, 59**

**Concat method, 111-112**

**concatenating strings, 111-112**

**concatenation operators, 438-439**

**concurrent garbage collection, assemblies, 928**

**conditional AND operator (&&), 58**

**conditional debugging, System.Diagnostics namespace, 881-884**

**conditional logical operators, 218**

**conditional OR operator (||), 58-59**

**configuration**

- assemblies, 927-930
- ClickOnce, 957-958
- objects, VS2008 Class Designer, 750
- ServiceInstaller, 690
- ServiceProcessInstaller, 689
- unsafe code, 61
- virtual directories, 963-964
- web servers, 962-963
- web services, 734-737
- Windows services, 687-688

**conflicts, avoiding, namespaces, 275**

**connected mode, ADO.NET, 443-444**

**connection performance, databases, 443**

**connections, ADO.NET, creating, 445-447**

**console applications, 507**

- colors, adding to, 511-514
- command-line input, handling, 510-511
- PasswordGenerator, 508
- programs, interacting with, 45-48
- users, interacting with, 508-510

**Console class, 23, 45**

**console screen communications, 45**

**Console.WriteLine method, 210-211**

**constant fields, objects, 165-166**

**constraints**

- base class constraints, 385-386
- constructor constraints, 387
- interfaces, 384-385
- reference type constraints, 386
- value type constraints, 387

**construction**

- reference types, 93
- value types, 93

**constructor constraints, 387**

**constructors**

- default constructors, object initialization, 323
- instance constructors, objects, 320-322
- overloading, object initialization, 322-323
- private constructors, object initialization, 323-324
- static constructors, object initialization, 325

**consuming web services, 737-739**

**Contact base class, 190**

**containment, encapsulation, 190**

**Contains method, 113-114**

**ContentControl control (WPF), 561**

**Context state (ASP.NET), 598**

**continue statements, 76**

**contracts**

- data contracts, creating, 730-732
- web services, creating, 727-732

**controllers, Windows services, communicating with, 691-693**

**controlling objects, 335, 337**

**controls**

- ASRNET, 593
  - HTML controls, 595-596
  - Menu control, 605
  - Server controls, 593, 595
- ASRNET AJAX controls, 625-635
  - accessing, 628-635
  - Timer control, 628
  - UpdatePanel control, 625-627
  - UpdateProgress control, 627-628

- Canvas, 553
- default events, 525
- DockPanel, 559
- Grid, 555-558
- ListView, binding to, 578
- Silverlight, 644-646
- StackPanel, 554
- UniformGrid, 555
- Windows Forms applications, 528-531
  - DataGridView control, 534-536
  - ListBox control, 534
  - MenuStrip control, 531-533
  - StatusStrip control, 531-533
  - ToolStrip control, 531-533
- WPF (Windows Presentation Foundation), 560
  - Border control, 560-561
  - Button control, 561
  - CheckBox control, 561
  - ComboBox control, 561
  - ContentControl control, 561
  - DockPanel control, 562
  - DocumentViewer control, 562
  - Ellipse control, 563
  - Expander control, 563
  - Frame control, 563
  - Grid control, 564
  - GridSplitter control, 564
  - GroupBox control, 564-565
  - Image control, 565
  - Label control, 565
  - ListBox control, 565
  - ListView control, 565
  - MediaElement control, 565
  - Menu control, 565-566
  - PasswordBox control, 566
  - ProgressBar control, 566
  - RadioButton control, 566-567
  - Rectangle control, 567
  - RichTextBox control, 567
  - ScrollBar control, 567
  - ScrollViewer control, 568
  - Separator control, 568
  - Slider control, 569
  - StackPanel control, 569
  - StatusBar control, 569
  - Tab control, 569
  - TextBlock control, 570
  - TextBox control, 570
  - ToolBar control, 570
  - ToolBarPanel control, 570-571
  - ToolBarTray control, 571
  - TreeView control, 571
  - UniformGrid control, 571
  - ViewBox control, 572
  - WindowsFormsHost control, 572-573
  - WrapPanel control, 573
- WrapPanel, 553-554
- XAML, 550-551
- conventions, code, 37**
- conversion operators, 438**
  - overloads, 218-227
- conversions**
  - automatic expression conversions, 220
  - custom reference type conversion operators, 225-227
  - custom value type conversion operators, 222-225
  - explicit conversions, 219-222
  - implicit conversions, 219-222
- converting resource files, 836-838**
- cookies, issuing, ASP.NET, 598**
- Copy method, 112-113**
- copying**
  - code, 623
  - strings, 112-113
- CopyTo method, 114-115**
- Couple of Interfaces with Identical Members listing (14.6), 304-305**
- Creating a Socket Client listing (30.2), 665-669**
- Creating a Socket Server listing (30.1), 662-665**
- Creating a Socket Server: MultiCast.cs listing (12.1), 253-254**
- Creating an HTTP Client listing (30.3), 669-671**
- credentials, processes, opening, 818-819**
- CSVs (comma-separated values), strings, 118**
- CTS (Common Type System), 16-17**
- curly braces, 21**
- custom actions, launch conditions, 953-954**
- Custom Attribute Example listing (16.6), 346-348**
- custom numeric format strings, 108**

**custom operators, 201**

**Custom Performance Counter Sampling:**  
**CustomSamplingProcessor.cs listing (42.18),**  
**913-917**

**custom reference type conversion operators,**  
**225-227**

**custom script libraries, loading, 623-625**

**custom types, 15**

logical operator overloads, 215-216

mathematical operator overloads, 213-215

**custom value type conversion operators,**  
**222-225**

**Customer class, 190-191**

**customized performance counters, building,**  
**897, 902-907**

**CustomValidator server control (ASP.NET), 593**

## D

### data

disconnected data, working with, 453-457

filtering, LINQ, 412

grouping, LINQ, 413

hiding, 185

joining, LINQ, 413-414

manipulating, ADO.NET, 450-452

updating after running samples, 420

viewing, ADO.NET, 447-450

**data abstractions, creating, ADO.NET Entity**  
**Framework, 475-489**

### data binding

ASP.NET, 614-617

Windows Forms applications, 533-536

WPF (Windows Presentation Foundation)  
 applications, 574-578

**data contracts, creating, 730-732**

**data entities, LINQ to SQL, 418**

**data handling logic, extending, partial methods,**  
**425-427**

**data providers, ADO.NET, 444-445**

**data structures, nongeneric collections, 366**

### databases

connection performance, 443

login security, ADO.NET, 446-447

modifying, stored procedures, 421-424

scalability, 443

### DataContext

declaring, 575-576

defining, 415-418

modifying, 419-420

querying through, 418-419

**DataGrid, Silverlight, 652**

**DataGridView control (Windows Forms), 528,**  
**534-536**

**DataList server control (ASP.NET), 594**

### DataSet object

data, reading, 453-454

saving to database, 454-457

**DataTemplate, 576-577**

### DateTime object

math, 98-99

string types, converting between, 99-101

### DateTime objects

creating, 97

extracting parts of, 97-98

**DateTimePicker control (Windows Forms), 528**

### Debug class, 879, 917

conditional debugging, 881-884

runtime debugging, 880-881

### Debug.Assert method, 917

assertions, making, 886-887

**debugger demo program, 147-148**

**debuggers, programming errors, finding, 152,**  
**155-156**

### debugging

applications, 147-159

assertions, making, 886-887

Boolean switches, 883-884

built-in performance counters, accessing,  
 888, 893-895

CLR (Common Language Runtime), 12

code

breakpoints, 148-149

stepping through, 147-152, 155-159

conditional debugging, System.Diagnostics  
 namespace, 881-884

customized performance counters, building,  
 897, 902-907

performance analysis, sampling, 908, 913,  
 916-917

- processes, attaching to, 156-159
- runtime debugging, 879
  - Debug class, 880-881
  - runtime tracing, System.Diagnostics namespace, 884-886
  - timers, implementing, 896-897
- Debugging Demo Program listing (7.1), 147**
- Debugging with Conditional Attributes: ConditionalDebugDemo.cs listing (42.3), 882**
- Declarative Security Request listing (44.1), 937-938**
- declaring**
  - DataContext, 575-576
  - OperationContract attributes, 730
  - properties, objects, 167
  - ServiceContract attribute, 730
- decrement operator (—), 51**
- default alignment, WPF (Windows Presentation Foundation), 552**
- default constructors, object initialization, 323**
- default events, controls, 525**
- DefineDynamicModule method, 362**
- defining**
  - DataContext, 415, 417-418
  - delegates, 250-251
  - event handlers, 258-259
  - interface types, 291-293
    - events, 293
    - indexers, 293
    - methods, 292
    - properties, 292
  - methods, 202-203
  - types, generic types, 384
- definite assignment, 44**
- delegates, 250, 271**
  - defining, 250-251
  - equality, checking, 255-256
  - function pointers, compared, 251
  - generic delegates, 382-384
  - hooking up, 251-252
  - inference, implementing, 256
  - lambda expressions, 258, 399-404
  - method handlers, creating, 251
  - methods, invoking through, 252
  - multicasting, 252, 254-255
- deleting**
  - data, ADO.NET, 451-452
  - entities, ADO.NET Data Services, 502-503
- deploying**
  - desktop applications, 955-958
  - web servers, 965
  - Windows services, 690-691
- deployment, assemblies, 930**
- design patterns, 755**
  - iterator pattern, 755-756
    - client code, 763
    - foreach loop, 764-767
    - IEnumerable interface, 756-763
    - Reset method, 766
    - simplifying with iterators, 767-768
  - proxy pattern, 768-771
    - proxy object, 771-772
  - template pattern, 772-777
    - .NET Framework, 773
    - implementing, 773-777
- designing**
  - exceptions, 243-245
  - objects, 163-175
- Designing an Exception: NewException.cs listing (11.6), 243-245**
- desktop applications, deploying, 955-958**
- desktop N-Layer multiple-assembly architecture, 792-794**
- desktop N-Layer single-assembly architecture, 785-789**
- destructors**
  - detriments, 332
  - finalizers, compared, 328
- details, processes, reading, 822**
- DetailsView server control (ASP.NET), 594**
- diagrams, closing, 747**
- dialog boxes, Windows Forms applications, 539-545**
  - common dialog boxes, 543-545
  - modal dialog boxes, 539-540
  - modeless dialog boxes, 539-540
  - window communications, 540-543
- disconnected data, working with, ADO.NET, 453-457**

- directives, namespaces, 275**
  - alias directive, 276-278
  - using directive, 275-276
- directories, virtual directories, setting up, 963-964**
- disabling processes, 822-823**
- disconnected mode, ADO.NET, 443-444**
- Dispose pattern, 332-334**
- Dispose Pattern: DisposePattern.cs listing (15.1), 332-333**
- disposing iterators, 395-396**
- division operator (/), 52**
- DLLs (dynamic link libraries), 13**
- do loops, 71-72**
- DockPanel control (WPF), 562**
- DockPanel layout (WPF), 559**
- documents**
  - XML documents, creating, 468, 470
  - XPS documents, creating, 562
- DocumentViewer control (WPF), 562**
- DOM (Document Object Model), XML DOM, 466-468**
- DomainUpDown control (Windows Forms), 528**
- drag-and-drop problems, RAD (rapid application development), 779-781**
- drawing text, Windows Forms applications, 537-539**
- DropDownList server control (ASP.NET), 594**
- Dynamic Assembly Generation listing (16.11), 360-361**
- dynamic link libraries (DLLs), 13**
- dynamically activating code, 357-359**
- Dynamically Activating Code:**
  - DynamicActivation.cs listing (16.10), 358-359

## E

- early-bound COM component calls, 866-868**
- ECMA (European Computer Manufacturers Association), .NET standardization, 10**
- EDM (Entity Data Model), 476-480, 492**
- element operators, 437**
- elements, resource files, 832**

- Ellipse control (WPF), 563**

### email

- attachments, sending, 676
- sending, 675-676

### encapsulation

- access modifiers, 185
  - internal access, 187-188
  - private access, 186
  - protected access, 187
  - protected internal access, 188
  - public access, 185
- containment, 190
- data, hiding, 185
- inheritance, 190
- objects, managing, 177
- OOP (object-oriented programming), 184-190

- endpoint element, web services, 736**

- EndsWith method, 113-114**

- EnsureCapacity method, 123-124**

- enterprise security policy level, 936**

### entities, 476

- accessing, Entity SQL, 480
- ADO.NET Data Services
  - adding, 501
  - sorting, 497
- creating, 482-486
- deleting, ADO.NET Data Services, 502-503
- EDM (Entity Data Model), 476-480
- mapping, 482-483
- querying
  - Entity SQL, 480-481
  - LINQ to Data Services, 503
  - LINQ to Entities, 486-487
  - WebDataQuery, 499-501
- selecting, Entity SQL, 480-481
- updating, ADO.NET Data Services, 501-502

- entity associations, ADO.NET Data Services, traversing, 497-498**

- Entity Data Model (EDM), 476-480, 492**

- entity items, ADO.NET Data Services, selecting, 493-495**

- entity results, ADO.NET Data Services, filtering, 495-497**

- entity sets, ADO.NET Data Services, viewing, 493**
- Entity SQL, entities, querying, 480-481**

**enums, 131, 139-142**

- ints, converting between, 142-143
- strings, converting between, 142-143
- System.Enum struct, 142-145
- type members, iterating through, 144

**Equal method, 110-111****equal operator (==), 54****equality**

- delegates, checking, 255-256
- objects, checking, 173
- strings, checking for, 110-111

**equality operators, 436-437****Equals method, 200**

- System.Object class members, overriding, 198-199

**errors, 231****European Computer Manufacturers Association (ECMA), .NET standardization, 10****Event Accessors: MenuItem.cs listing (12.6), 266-267****Event Declaration: MenuItem.cs**

listing (12.2), 259

**event handlers, defining, 258-259****event handling**

- Silverlight, 648-652
- state workflows, 811-813
- WPF (Windows Presentation Foundation) applications, 573-574

**Event Method Handlers: DelegatesAndEvents.cs listing (12.3), 260-261****Event Method Handlers: SiteManager.cs**

listing (12.4), 261-263

**event-based programming, 249-250, 271**

- anonymous methods, assigning, 256-258
- delegates, 250
  - checking delegate equality, 255-256
  - defining, 250-251
  - hooking up, 251-252
  - implementing inference, 256
  - method handler creation, 251
  - method invocation, 252
  - multicasting, 252-255
- event add/remove methods, modifying, 266-267, 271
- event handlers, defining, 258-259

**events**

- coding, 258-267, 271
- firing, 263-265
- implementing, 261-263
- registering for, 259-261

**events, 271**

- add/remove methods, modifying, 266-271
- coding, 258-267, 271
- controls, default events, 525
- firing, 263-265
- implementing, 261-263
- interface types, defining, 293
- registering for, 259-261

**evidence, code-based security, 934****exception management, CLR (Common Language Runtime), 12****exceptions, 231**

- designing, 243-245
- handling, 232-240
  - different exception types, 236-237
  - finally blocks, 234-235
  - try/catch block, 232-234
- passing, 237-240
- recovering from, 240-243

**executable files**

- assemblies, compared, 11
- compiling to, 10

**executable works, ensuring, 363****executing code, CLR (Common Language Runtime), 12-14****execution management, CLR (Common Language Runtime), 12****existing processes, working with, 821-823****Expander control (WPF), 563****explicit alignment, WPF (Windows Presentation Foundation), 552-553****explicit conversions, implicit conversions, compared, 219-222****explicit implementation, interfaces, 304-310****Explicit Interface Implementation listing (14.7), 306-308****expression trees, 397, 404-406**

- lambda expressions, converting to, 404-406



**expressions**

- expression trees, 397, 404-406
    - lambda conversions, 404-406
  - grep (Global Regular Expression Print), 127-128
  - lambda expressions, 397-398, 406
    - delegates, 258, 399-404
    - syntax, 398
    - using, 398-399
  - regular expressions, 124-125
    - application, 127-129
    - common character classes, 126
    - operations, 125-126
    - quantifiers, 127
- Extensible HTML (XHTML), 589**
- Extensible Markup Language (XML), 461**
- extension methods, 228-230**
- extern namespaces alias, 284-286**
- ExternalDataExchangeService, implementing, 805-810**
- extracting string information, 114-115**
- extending data handling logic, partial methods, 425-427**

**F****FCL (Framework Class Library), 9, 14-17, 23**

- attributes, creating, 345-349
- code reuse, 15-16
- features, 15
- language neutrality, 16
- namespaces, 14-15, 274-276
- reusable types, 14

**fields**

- attributes, requirements, 348
- capturing, anonymous methods, 257
- objects, 165
  - constant fields, 165-166
  - read only fields, 166

**file system setup projects, configuring, 950****file types, setup projects, 951****files**

- FTP servers
  - downloading from, 673-675
  - uploading to, 671-673

**resource files, 831**

- converting, 836-838
- creating, 831-834
- elements, 832
- graphical resource files, 838-843
- multiple locales, 843-850
- reading, 835-836
- writing, 834-835

**Silverlight class library, 646-647****Windows Forms applications, 520****FileUpload server control (ASP.NET), 594****filtering**

- ADO.NET Data Services entity results, 495-497
- data, LINQ, 412

**filtering operators, 430-431****finalization**

- objects, 327-328
- reference types, 93
- value types, 93

**finalizers, destructors, compared, 328****finally Block: Exceptions2.cs listing (11.2), 234-235****finally blocks, exception handling, 234-235****FindAll method, 398****firing events, 263, 265****Firing Events: Menu.cs listing (12.5), 264-265****FirstProgram.cs, creating, 20-23****fixed statement, 861-864****fixed Statement Demo: FixedStatementDemo.cs listing (41.7), 862-864****Flags attribute, 341****floating-point types, 42-43****folders, Project and Solution folders, 25****fonts, Windows Forms application, 537-539****for loops, 72-73****foreach loops, 73**

- iterator pattern, 764-767

**ForEach method, 398****Form control (Windows Forms), 528****Form1.cs after Adding and Configuring a Button listing (25.5), 525-526****Form1.Designer.cs After Adding and Configuring a Button listing (25.6), 526-527****Format method, 106-109**

formatting strings, 106-109

forms

ASP.NET, 588-593

web forms, 588

FormView server control (ASP.NET), 594

Frame control (WPF), 563

framework class library types, 16

framework types (NET), 94-101

FrontPage Server Extensions, 643

FTP file transfers, performing, 671-675

FTP servers, files

downloading from, 673-675

uploading to, 671-673

FullAddress method, 181-182

function pointers, delegates, compared, 251

functions, 14. *See also* methods

SQL functions, using, 421

fundamentals, Windows Forms application, 516-519

## G

GAC (global assembly cache), 930

garbage collection, CLR (Common Language Runtime), 12

Garbage Collector (GC), 327

GC (Garbage Collector), 327-330

class members, 335

collection process, 330

interacting with, 335-337

optimization, 330-331

running, 329

GDI+, Windows Forms applications, 536-539

Brush object, 536-537

drawing text, 537-539

fonts, 537-539

Graphics object, 536-537

Pen object, 536-537

Generated .resx File: strings.resx listing (40.5), 837-838

GenerateException method, 239

GenerateReport method, 185

generation operators, 435-436

generic collections, 396

arrays, compared, 371-372

nongeneric collections, compared, 371-372

generic delegates, 382-384

generic interfaces, 381-382

generic nodes, implementing, 373-374

generic type objects, 365

generic types

benefits of, 366-372

building, 372-387

singly linked lists, implementing, 373-381

types, defining, 384

GenericPrincipal, role-based security, 943-944

GenericSingleLinkedList collection, 373-376

GenericSingleLinkedListT, 375-376

GetAttribute method, 356-357

GetEnumerator method, 388-390

GetGeneration class member (GC), 335

GetHashCode method, 173, 200

System.Object class members, overriding, 199

GetLowerBound method, 137-138

GetModule method, 356

GetReflectionInfo method, 356

Getting Attributes from a Class listing (16.9), 356-357

Getting Files from an FTP Server listing (30.5), 673-675

GetTotalMember class member (GC), 335

GetUpperBound method, 137-138

global assembly cache (GAC), 930

globally unique identifier (GUIDs), 95-96

goto statements, 63, 74-75

graphical resource files, creating, 838-840, 843

Graphics object, 2D graphics, creating, 536-539

Graphics object (GDI+), Windows Forms applications, 536-537

grep (Global Regular Expression Print) expressions, 127-128

Grid control (WPF), 564

Grid layout, WPF (Windows Presentation Foundation), 555-559

GridSplitter control (WPF), 564

GridView server control (ASP.NET), 594

group joins, hierarchies, building with, 414  
 GroupBox control (Windows Forms), 528  
 GroupBox control (WPF), 564-565  
 grouping data, LINQ (Language Integrated Query), 413  
 grouping operators, 434-435  
 GUIDs (globally unique identifiers), System.Guid, 95-96

## H

handlers, event handlers, defining, 258-259

handling

- automatic promotions, 220
- command-line input, console applications, 510-511
- compiler warnings, 194
- events
  - Silverlight, 648-652
  - state workflows, 811-813
  - WPF (Windows Presentation Foundation), 573-574
- exceptions, 232-240
  - different exception types, 236-237
  - finally blocks, 234-235
  - try/catch block, 232-234

hash values, objects, obtaining, 173

help, .NET Framework, receiving, 973-975

Help index, VS2008, 974

HiddenField server control (ASP.NET), 594

hiding

- base member classes, 180-181
- data, 185

hierarchies, building, group joins, 414

Hospital Database Schema listing (19.1), 415-417

host utility, setup, 703-706

Host Utility Configuration File:

- RemotingHost.exe.config listing (32.9), 706

Host Utility Demo: RemotingHost.cs  
 listing (32.8), 705

Hosted Server Demo: HostedServer.cs  
 listing (32.5), 703

hosts, workflows, communicating to, 805-810

HTML (Hypertext Markup Language), 589

HTML controls (ASP.NET), 595-596

HTTP (HyperText Transfer Protocol)

- clients, creating, 699-671
- message formats, 719
- network communications, working with, 669-671
- wrapped HTTP classes, 661

HTTP URIs, ADO.NET Data Services, accessing via, 493-498

HttpChannel, TCP channels, compared, 708

HyperLink server control (ASP.NET), 594

Hypertext Markup Language (HTML), 589

## I

IBroker Interface Definition listing (14.1), 293-294

identifiers

- code, 35-36
- yield, 389

identity, assemblies, 925

identity attributes, assemblies, 923

IDEs (integrated development environments), 19

IDL (Interface Definition Language), 339

IEnumerable interface, implementing, 756-763

IEnumerableT, 376-381

IEnumerator interface

- IEnumerator-derived type construction, 759-760
- moving, 761-762
- resetting, 760-761
- values, reading from, 762-763

IEnumeratorT, 376-381

if statements, 65

- if-else-else statement, 66-67
- if-then-else statement, 65-66

if-else-else statement, 66-67

if-then-else statement, 65-66

IIS (Internet Information Server), 643, 670

IL (Microsoft Intermediate Language), 10, 17

IListT, 375-376

Image control (WPF), 565

**Image server control (ASP.NET), 594**  
**ImageButton server control (ASP.NET), 594**  
**ImageMap server control (ASP.NET), 594**  
**impedance mismatches, 410**  
**Imperative Security Request listing (44.2), 939-940**  
**implementation**  
     explicit implementation, interfaces, 304-305, 308-310  
     generic nodes, 373-374  
     IEnumerableT, 376-381  
     IEnumeratorT, 376-381  
     IListT, 375-376  
     implicit implementation, interfaces, 293-304  
     interfaces, single-class interface implementation, 293-297  
     iterators, 388-396  
     singly linked lists, generic types, 373-381  
**Implementation of Explicit Interface Members listing (14.8), 308-309**  
**Implementation of Single Interface Inheritance listing (14.3), 296-297**  
**Implementation of the IBroker Interface listing (14.2), 294-295**  
**implementing**  
     events, 261-263  
     interfaces, 291  
     logic, WCF web services, 732-734  
**Implementing a ServiceController listing (31.7), 692-693**  
**Implementing Debugging with a Boolean Switch: BooleanSwitchDemo.cs listing (42.5), 883-884**  
**Implementing OnStart listing (31.4), 685**  
**Implementing OnStop listing (31.6), 687**  
**Implicit and Explicit Class Conversions listing (10.2), 226**  
**Implicit and Explicit Struct Conversions listing (10.1), 222-225**  
**implicit conversions, explicit conversions, compared, 219-222**  
**implicit implementation, interfaces, 293-304**  
**interfaces, IEnumeratorT, 376-381**  
**increment operator (++), 50-51**  
**indexer iterators, 391-393**

**indexers**  
     interface types, defining, 293  
     objects, 169-170  
     polymorphism, 196-197  
**IndexOf method, 114-115**  
**IndexOutOfRangeException, 239**  
**inference, delegates, implementing, 256**  
**informational attributes, assemblies, 923-924**  
**inheritance, 81**  
     attributes, 349  
     encapsulation, 190  
     interfaces, 312-315  
     namespaces, 274  
     object initialization, 324-325  
     objects, 315  
     observing, VS2008 Class Designer, 747-748  
     OOP (object-oriented programming), 178-184  
         base classes, 178-180  
         sealed classes, 183-184  
         versioning, 180-183  
     reference types, 92-93  
     single-implementation inheritance, 93  
     value types, 92-93  
**inheritance relationships, objects, 177**  
**initialization**  
     arrays, 132  
     objects, 320  
         default constructors, 323  
         inheritance, 324-325  
         instance constructors, 320-322  
         order of instantiation, 324-325  
         overloading constructors, 322-323  
         private constructors, 323-324  
         static constructors, 325  
**initializers, objects, 326-327**  
**Insert method, 117**  
**inserting data, ADO.NET, 450-451**  
**installation, Windows services, 688-691**  
**InstallUtil, 690, 693**  
**instance constructors, object initialization, 320-322**  
**instances, objects, 164-165**  
**integral types, 39-41**

- integrated development environments (IDEs), 19
- IntelliSense, overloads, finding, 106
- IntelliSense
  - overrides, 195
  - partial methods, 228
- Interacting with the Garbage Collector:
  - CollectGenerations.cs listing (15.2), 335-337
- interaction, programs, console applications, 45-48
- Interface Definition Language (IDL), 339
- interface differences, abstract classes, 290
- Interface Mapping Example listing (14.9), 310-312
- interface types, defining, 291-293
- interfaces, 186, 315
  - ATMs (automated teller machines), 287-288
  - constraints, 384-385
  - explicit implementation, 304-310
  - generic interfaces, 381-382
  - IEnumerable interface, implementing, 756-763
  - IEnumerableT, 376-381
  - IListT, 375-376
  - implementing, 291
    - single-class interface implementation, 293-297
  - implicit implementation, 293-304
  - inheritance, 312-315
  - mapping, 310-312
  - observing, VS2008 Class Designer, 747-748
  - polymorphic behavior, simulating, 298-304
  - types, defining, 291-293
  - WCF web services, creating, 727-730
- Interfaces Inheriting Other Interfaces
  - listing (14.10), 313-314
- Intern method, 121-122
- intern pools, 122
  - CLR string handling, 121-122
- internal access modifiers, encapsulation, 187-188
- Internet Information Server (IIS), 643, 670
- Interop, CLR (Common Language Runtime), 12
- ints, enums, converting between, 142-143
- invocation results, web services, 717

- invoking methods through delegates, 252
- is operator, 60
- IsError positional parameter, 343
- IService1.cs Skeleton Code listing (34.1), 727-729
- IsValidUri method, 202, 212
- item template defaults, VS2008, 189
- items, Windows service projects, 680-683
- iteration, enum type members, 144
- iterator pattern, 756
  - client code, using in, 763
  - foreach loop, 764-767
  - IEnumerable interface, implementing, 756-763
  - Reset method, 766
  - simplifying with iterators, 767-768
- iterators, 366, 396
  - disposing, 395-396
  - GetEnumerator method, 388-390
  - implementing, 388-396
  - indexer iterators, 391-393
  - iterator pattern, simplifying, 767-768
  - iterator types, 388
  - method iterators, 390-391
  - operator iterators, 393-394
  - property iterators, 391
  - sequence of values, 394-395

## J–K

- jagged arrays, 135-137
- Java packages, namespaces, 274
- JavaScript, Silverlight, relationship, 642
- JIT (Just-In-Time) activation, COM+ services, 874
- JIT (Just-In-Time) compilation, CLR (Common Language Runtime), 12-14
- JIT compiler (CLR), 14
- Join method, 118-119
- join operators, 433-434
- joining
  - data, LINQ (Language Integrated Query), 413-414
  - strings, 118-119

**KeepAlive class member (GC), 335**

**keywords**

code, 36-37

yield, 389

**killing processes, 822-823**

## L

**Label control (Windows Forms), 528**

**Label control (WPF), 565**

**Label server control (ASP.NET), 594**

**labels, 63-64**

**lambda expressions, 397-398, 406**

delegates, 258, 399-404

expression trees, converting to, 404-406

syntax, 398

using, 398-399

**Language Integrated Query (LINQ). See LINQ (Language Integrated Query)**

**language neutrality, FCL (Framework Class Library), 16**

**languages, .NET support, 16**

**LastIndexOf method, 114-115**

**lastName parameter, 398**

**late-bound COM component calls, 868-869**

**Late-Bound COM Component Invocation:**

**TalkToComLater.cs listing (41.13), 868-869**

**launch conditions**

custom actions, 953-954

setup projects, 953

**launching new processes, 818-821**

**layout, 37**

WPF (Windows Presentation Foundation)

applications

canvas layout, 553

default alignment, 552

DockPanel layout, 559

explicit alignment, 552-553

Grid layout, 555-559

managing, 551-559

StackPanel layout, 554

UniformGrid layout, 555

WrapPanel layout, 553-554

**less than operator (<), 55**

**less than or equal operator (<=), 55**

**levels**

namespaces, 15

policies, code-based security, 936-937

**libraries, compiling, 969-971**

**life cycles**

AJAX pages, 621-623

ASP.NET page life cycle, 590-593

**lifetimes**

managing, remoting, 709-711

objects, 319

automatic memory management,  
328-331

GC (Garbage Collector) interaction,  
335, 337

resource cleanup, 331-332, 334

**LinkButton server control (ASP.NET), 594**

**LinkLabel control (Windows Forms), 529**

**LINQ (Language Integrated Query), 228, 409, 439**

data

filtering, 412

grouping, 413

joining, 413-414

DataSet object, 458-459

group joins, building hierarchies with, 414

objects, accessing, 410-414

projections, extracting, 411-412

query operators, 427

aggregate operators, 439

concatenation operators, 438-439

conversion operators, 438

element operators, 437

equality operators, 436-437

filtering operators, 430-431

generation operators, 435-436

grouping operators, 434-435

join operators, 433-434

partitioning operators, 433

projection operators, 432

quantifier operators, 431-432

set operators, 428-430

sorting operators, 427-428

query results, ordering, 412-413

syntax, 410-411

types, 15

**LINQ to Data Services, entities, querying, 503**

**LINQ to Entities, coding with, 486-489**

### **LINQ to SQL**

data entities, 418

databases, modifying, 421-424

DataContext

defining, 415-418

modifying, 419-420

querying through, 418-419

relational data, querying, 414-427

SQL functions, using, 421

stored procedures, calling, 420-421

### **LINQ to XML**

documents, creating, 468-470

namespaces, 470-471

objects, 468

XML, manipulating, 468-472

XML documents

modifying, 472

querying, 471

reading, 471

**ListBox, binding to, 576-577**

**ListBox control (Windows Forms), 529, 534**

**ListBox control (WPF), 565**

**ListBox server control (ASP.NET), 594**

### **listings**

2.1 (Simple C# Program), 20

2.2 (C# Program in VS2008), 27

5.1 (Regular Expressions Application), 127

7.1 (Debugging Demo Program), 147

7.2 (Program with Bugs), 152-155

10.1 (Implicit and Explicit Struct Conversions), 222-225

10.2 (Implicit and Explicit Class Conversions), 226

11.1 (Simple Exception: Exceptions.cs), 233

11.2 (finally Block: Exceptions2.cs), 234-235

11.3 (Multiple catch Blocks: Exceptions3.cs), 236-247

11.4 (Passing Exceptions: Exceptions4.cs), 237-239

11.5 (Recovering from Exceptions: ExceptionTester.cs), 240-242

11.6 (Designing an Exception: NewException.cs), 243-245

11.7 (checked Statements: checked.cs), 245-246

11.8 (unchecked Statements: unchecked.cs), 246-247

12.1 (Creating a Socket Server: MultiCast.cs), 253-254

12.2 (Event Declaration: MenuItem.cs), 259

12.3 (Event Method Handlers: DelegatesAndEvents.cs), 260-261

12.4 (Event Method Handlers: SiteManager.cs), 261-263

12.5 (Firing Events: Menu.cs), 264-265

12.6 (Event Accessors: MenuItem.cs), 266-267

12.7 (Rest of the Program: WebSites.cs), 268, 271

12.8 (Compilation Instructions for Chapter 12 listings), 271

14.1 (IBroker Interface Definition), 293-294

14.2 (Implementation of the IBroker Interface), 294-295

14.3 (Implementation of Single Interface Inheritance), 296-297

14.4 (Another Implementation of the IBroker Interface), 298-301

14.5 (Using Two Classes with the Same Interface), 302-303

14.6 (Couple of Interfaces with Identical Members), 304-305

14.7 (Explicit Interface Implementation), 306-308

14.8 (Implementation of Explicit Interface Members), 308-309

14.9 (Interface Mapping Example), 310-312

14.10 (Interfaces Inheriting Other Interfaces), 313-314

15.1 (Dispose Pattern: DisposePattern.cs), 332-333

15.2 (Interacting with the Garbage Collector: CollectGenerations.cs), 335-337

16.1 (Using a Single Attribute), 340-341

16.2 (Using Multiple Attributes), 341

16.3 (Positional Parameters), 342

16.4 (Named Parameters), 343

16.5 (Attribute Targets), 344

16.6 (Custom Attribute Example), 346-348

16.7 (Class to Reflect Upon: Reflected.cs), 350-352

- 16.8 (Performing Reflection: Reflecting.cs), 352-355
- 16.9 (Getting Attributes from a Class), 356-357
- 16.10 (Dynamically Activating Code: DynamicActivation.cs), 358-359
- 16.11 (Dynamic Assembly Generation), 360-361
- 17.1 (You Can Use Objects Polymorphically with Generics), 370-371
- 17.2 (GenericSingleLinkedListT, 375-376
- 19.1 (Hospital Database Schema), 415, 417
- 21.1 (Writing an XML Document with XmlTextWriter), 462-464
- 21.2 (Reading an XML Document with XmlTextReader), 465
- 24.1 (Simple Console Application), 508
- 24.2 (Retrieving User Input from the Command-Line), 510
- 25.1 (Windows Forms Application Showing Fundamentals), 516-518
- 25.2 (Windows Forms Program.cs File), 521
- 25.3 (Windows Forms Form1.cs File), 522
- 25.4 (Windows Forms Form1.Designer.cs File), 523-525
- 25.5 (Form1.cs after Adding and Configuring a Button), 525-526
- 25.6 (Form1.Designer.cs After Adding and Configuring a Button), 526-527
- 26.1 (Skeleton XAML for a WPF Application), 549
- 26.2 (Code-Behind File), 549-550
- 26.3 (Using the Grid Layout Control), 556-558
- 27.1 (Web Form with Multiple Controls), 591-592
- 27.2 (Code-Behind Demonstrating Page Life Cycle), 592
- 29.1 (Silverlight Control on a Web Form), 645-646
- 29.2 (Silverlight User Control), 646
- 30.1 (Creating a Socket Server), 662-665
- 30.2 (Creating a Socket Client), 665-669
- 30.3 (Creating an HTTP Client), 669-671
- 30.4 (Uploading a File to an FTP Server), 671-673
- 30.5 (Getting Files from an FTP Server), 673-675
- 31.1 (Partial Class for a Windows Service), 681-682
- 31.2 (Windows Service Entry Point), 682-683
- 31.3 (Windows Service Code), 684
- 31.4 (Implementing OnStart), 685
- 31.5 (MoneyServer Implementation), 686-687
- 31.6 (Implementing OnStop), 687
- 31.7 (Implementing a ServiceController), 692-693
- 32.1 (Remoting Server Demo: BasicRemotingServer.cs), 697
- 32.2 (Remote Server Component Configuration File: web.config), 697-698
- 32.3 (Basic Remoting Client: BasicRemotingClient.cs), 699-700
- 32.4 (Client Configuration File: BasicRemotingClient.exe.config), 700
- 32.5 (Hosted Server Demo: HostedServer.cs), 703
- 32.6 (Remoting Client Demo: HostedClient.cs), 704
- 32.7 (Remoting Client Configuration File: HostedClient.exe.config), 704-705
- 32.8 (Host Utility Demo: RemotingHost.cs), 705
- 32.9 (Host Utility Configuration File: RemotingHost.exe.config), 706
- 32.10 (Remote Leasing Demo: LeasingDemo.cs), 709-711
- 33.1 (Web Service Header: BasicWebService.asmx), 714
- 33.2 (Web Service Code: BasicWebService.asmx.cs), 715
- 33.3 (Autogenerated Web Service Proxy: BasicWebService.cs), 720-721
- 33.4 (Using a Web Service: WebServiceClient.cs), 722
- 34.1 (IService1.cs Skeleton Code), 727-729
- 34.2 (WCF Web Service Implementation), 732-734
- 34.3 (Client Code Calling a Web Service), 738-739
- 36.1 (Collection Without IEnumerable/IEnumerator Interfaces), 764-766
- 36.2 (Proxy for Encapsulating TCP/IP Calls to a Server), 768, 770



- 40.1 (txt Resource File: strings.txt), 832
- 40.2 (Using Resources: StringRes.cs), 833
- 40.3 (Writing a Resource File: ResWrite.cs), 834-835
- 40.4 (Reading a Resource file: RegRead.cs), 835-836
- 40.5 (Generated .resx File: strings.resx), 837-838
- 40.6 (Using Graphical Resources: GraphRes.cs), 840-842
- 40.7 (Localized Program: MultiCulture.cs), 845-848
- 41.1 (Pointer Arithmetic: PointerArithmetic.cs), 856-858
- 41.2 (Compilation Instructions for Listing 41.1), 858
- 41.3 (Using the sizeof() Operator), 858-859
- 41.4 (Compilation Instructions for Listing 41.3), 859
- 41.5 (stackalloc Demonstration: StackAllocDemo.cs), 860-861
- 41.6 (Compilation Instructions for Listing 41.5), 861
- 41.7 (fixed Statement Demo: FixedStatementDemo.cs), 862-864
- 41.8 (Compilation Instructions for Listing 41.7), 864
- 41.9 (Platform Invoke Demo: PinvokeDemo.cs), 864-865
- 41.10 (Compilation Instructions for Listing 41.9), 866
- 41.11 (C++ COM Component: ComObj.dll), 867
- 41.12 (C# Program Calling a COM Component: TalkToCom.cs), 867-868
- 41.13 (Late-Bound COM Component Invocation: TalkToComLater.cs), 868-869
- 41.14 (C# Component Exposed as a COM Component: CallFromCom.dll), 869-870
- 41.15 (Calling a C# Component Exposed as a COM Component), 870
- 41.16 (Minimal C# COM+ Component), 871-872
- 41.17 (COM+ Transactional Component in C#: CPTrans.cs), 873-874
- 41.18 (C# Component Configured for JIT Activation: CPJit.cs), 874
- 41.19 (COM+ Services Object Pooling Implemented in C#: CPPool.cs), 875-876
- 42.1 (Simple Debugging Example: PlainDebugDemo.cs), 880-881
- 42.2 (Compilation Instructions for Listing 42.1), 881
- 42.3 (Debugging with Conditional Attributes: ConditionalDebugDemo.cs), 882
- 42.4 (Compilation Instructions for Listing 42.3), 883
- 42.5 (Implementing Debugging with a Boolean Switch: BooleanSwitchDemo.cs), 883-884
- 42.6 (BooleanSwitch entry in Configuration File: BooleanSwitchDemo.config), 884
- 42.7 (Compilation Instructions for Listing 42.5), 884
- 42.8 (TraceSwitch Class Demo: TraceSwitchDemo.cs), 885
- 42.9 (TraceSwitch entry in Config File: TraceSwitchDemo.config), 886
- 42.10 (Compilation Instructions for Listing 42.8), 886
- 42.11 (Assertion Demonstration: AssertDemo.cs), 887
- 42.12 (Compilation Instructions for Listing 42.11), 887
- 42.13 (System Performance Counter Demo: OrderClient.cs), 888-893
- 42.14 (Server Component of System Performance Counter Demo: OrderProcessor.cs), 893
- 42.15 (Server Component of System Performance Counter Demo: OrderProcessor.cs), 894-895
- 42.16 (Client Using Data from Custom Performance Counter: CustomOrderClient.cs), 897-902
- 42.17 (Server Implementing a Custom Performance Counter: CustomOrderProcessor.cs), 902-907
- 42.18 (Sampling Client: SampleClient.cs), 908-913
- 42.19 (Custom Performance Counter Sampling: CustomSamplingProcessor.cs), 913-917
- 44.1 (Declarative Security Request), 937-938
- 44.2 (Imperative Security Request), 939-940
- 44.3 (Role-Based Security with WindowsPrincipal), 942-943
- 44.4 (Role-Based Security with GenericPrincipal), 943-944

- ListItem objects, 175
- lists, processes, obtaining, 821-822
- lists of data, WPF (Windows Presentation Foundation) applications, displaying, 575-578
- ListView control (Windows Forms), 529
- ListView control (WPF), 565
  - binding to, 578
- ListView server control (ASP.NET), 594
- loading custom script libraries, 623-625
- local variables
  - anonymous methods, capturing, 257
  - methods, 203-204
    - definite assignment, 44
- locales, multiple locales, 843-844
  - implementing, 844-849
  - obtaining resources, 849-850
- localization, 831
  - multiple locales, 843-844
    - implementing, 844-849
    - obtaining resources, 849-850
  - resource files, 831
    - converting, 836-838
    - creating, 831-834
    - graphical resource files, 838-843
    - reading, 835-836
    - writing, 834-835
- Localized Program: MultiCulture.cs listing (40.7), 845-848
- Locals window (VS2008), 149
- location elements, assemblies, 929-930
- lock statement, thread synchronization, 826-828
  - Monitor class, 827-828
- logic, WCF web services, implementing, 732-734
- logical complement operator (!), 51
- logical operator overloads, custom types, 215-216
- logical operators, 56
  - bitwise AND operator (&), 56
  - bitwise exclusive OR operator (^), 57
  - bitwise inclusive OR operator (|), 57
  - Boolean AND operator (&), 57
  - Boolean exclusive OR operator (^), 58
  - Boolean inclusive OR operator (|), 57

- conditional AND operator (&&), 58
- conditional logical operators, 218
- conditional OR operator (||), 58-59
- side effects, 59
- login security, databases, ADO.NET, 446-447
- loops, 69
  - do loops, 71-72
  - for loops, 72-73
  - foreach loop, iterator pattern, 764-767
  - foreach loops, 73
  - while loops, 70
- low-level work (plumbing), 11

## M

- machine security policy level, 936
- Main method, 14, 21-22, 30, 239
- Makecert.exe security utility, 944
- managed code, 854
- managed compilation, 10
- managed software, 9-10
- managing
  - layout, WPF (Windows Presentation Foundation), 551-559
  - process streams, 820-821
- manifest attributes, assemblies, 924
- manifests, assemblies, 922
- mapping interfaces, 310-312
- matching curly braces, 21
- mathematical operator overloads, custom types, 213-215
- MaxGeneration class member (GC), 335
- McIlroy, Doug, 128
- media, playing, Silverlight, 652-655
- MediaElement (Silverlight), manipulating, 653-655
- MediaElement control (WPF), 565
- MediaPlayer, WebForms, adding to, 652-653
- members
  - base classes
    - calling, 179-180
    - hiding, 180-181
  - classes, qualification, 179
  - namespaces, 281

- objects, 163-164
  - fields, 165-166
  - indexers, 169-170
  - instances, 164-165
  - methods, 166-167
  - properties, 167-169
  - static classes, 171-172
  - static members, 164-165
  - System.Object class, 172-175
  - System.Object class, overriding, 197-200
- membership, code groups, 936**
- MemberwiseClone method, 174**
- memory allocation, 328-329**
  - reference type, 86
  - value type, 86-87
- Menu control (ASP.NET), 605**
- Menu control (WPF), 565-566**
- MessageBox control (Windows Forms), 529-533**
- metadata, assemblies, 10**
- method handlers, delegate method handlers, creating, 251**
- method iterators, 390-391**
- methods, 14, 397**
  - add, 266-267, 271
  - Anonymous, 670
  - anonymous methods, assigning, 256-258
  - Append, 123
  - AppendFormat, 123
  - BinarySearch, 138-139
  - coding methods, 201-202
  - CombinePath, 229
  - Compare, 109-110
  - CompareOrdinal, 109-110
  - Concat, 111-112
  - Console.WriteLine, 210-211
  - Contains, 113-114
  - Copy, 112-113
  - CopyTo, 114-115
  - Debug.Assert, 886-887, 917
  - DefineDynamicModule, 362
  - defining, 202-203
  - delegates, invoking through, 252
  - EndsWith, 113-114
  - EnsureCapacity, 123-124
  - Equals, 110-111, 198-200
  - extension methods, 228-230
  - FindAll, 398
  - ForEach, 398
  - Format, 106-109
  - FullAddress, 181-182
  - GenerateException, 239
  - GenerateReport, 185
  - GetAttribute, 356-357
  - GetEnumerator, 388-390
  - GetHashCode, 173, 199-200
  - GetLowerBound, 137-138
  - GetModule, 356
  - GetReflectionInfo, 356
  - GetUpperBound, 137-138
  - IndexOf, 114-115
  - Insert, 117
  - interface types, defining, 292
  - Intern, 121-122
  - IsValidUrl, 202, 212
  - Join, 118-119
  - LastIndexOf, 114-115
  - local variables, 203-204
    - definite assignment, 44
  - Main, 14, 21-22, 30, 239
  - MemberwiseClone, 174
  - objects, 166-167
  - OnPaint, 537, 539
  - OnStart, 684-687
  - OnStop, 687
  - overloading, 210-212
  - overriding, Windows services, 683-684
  - PadLeft, 115-116
  - PadRight, 115-116
  - parameters, 204
    - out parameters, 207-209
    - params parameters, 209-210
    - ref parameters, 205-207
    - value parameters, 204-205
  - partial methods, 227-228, 425-427
    - IntelliSense, 228
  - PredictLocation, 206
  - ReadLine(), 45
  - Remove, 117, 266-267, 271
  - Replace, 118
  - Reset, 766
  - Save, 363
  - SendAlert, 191, 194

- Split, 118-119
- Start, 818
- StartsWith, 113-114
- ToCharArray, 119-121
- ToLower, 118
- ToString, 124, 175, 199-200
- ToUpper, 118
- Trim, 115-116
- UpdateSite, 205
- ValidateUrl, 211-212
- WriteLine, 23
- Microsoft .NET, 9**
- Microsoft Intermediate Language (MSIL), 10**
- Minimal C# COM+ Component listing (41.16), 871-872**
- minus operator (-), 50**
- MMC configuration tool, runtime configuration, assemblies, 930**
- modal dialog boxes, Windows Forms applications, 539-540**
- modeless dialog boxes, Windows Forms applications, 539-540**
- models, web applications, 583-586**
- modifying strings, 117-118**
- modules, skipping to types, 356**
- modulus operator (%), 53**
- MoneyServer Implementation listing (31.5), 686-687**
- Monitor class, thread synchronization, 827-828**
- Mono, 10**
- MonthCalendar control (Windows Forms), 529**
- Moore's law, 895**
- moving IEnumerator interface, 761-762**
- multicasting delegates, 252-255**
- multicore processors, 817**
- multidimension arrays, 134-135**
- multiline comments, code, 32**
- multiple attributes, using, 341-342**
- Multiple catch Blocks: Exceptions3.cs listing (11.3), 236-247**
- multiple locales (localization), 843-844**
  - implementing, 844-849
  - resources, obtaining, 849-850
- multiple-assembly N-Layer architectures, 792-795**

- multiplication operator (\*), 52**
- multiplicity, attributes, controlling, 349**
- multithreading, 823**
  - ThreadPool object, 825-826
  - threads
    - creating, 823-824
    - passing parameters to, 824-825
    - running code in, 824
- MultiView server control (ASP.NET), 594**
- MyCalc variable, 251**

## N

- N-Layer architecture, 782-784**
  - N-Layer/Multiple-Assembly architectures, 792-795
  - N-Layer/Single-Assembly architectures, 785-792
  - N-Tier architecture, compared, 781-782
- N-Layer/Single-Assembly architectures, 785-792**
- N-Layer/Multiple-Assembly architectures, 792-795**
- N-Tier architecture, 783-784**
  - N-Layer architecture, compared, 781-782
- named parameters, attributes, 343**
- Named Parameters listing (16.4), 343**
- NameMatch delegate parameter, 400**
- names**
  - assembly names, verifying, 700
  - Windows service project spaces, 680
- namespaces, 273-274, 286**
  - ADO.NET, 444-445
  - alias qualifiers, 283-284
  - assembly reference, compared, 277-278
  - benefits of, 284
  - CLR (Common Language Runtime), 274
  - code, organizing, 274
  - conflicts, avoiding, 275
  - creating, 278-281
  - directives, 275
    - alias directive, 276-278
    - using directive, 275-276
  - extern namespaces alias, 284-286
  - FCL (Framework Class Library), 14-15, 274-276

- inheritance, 274
- Java packages, 274
- levels, 15
- LINQ to XML, 470-471
- managing, VS2008, 286
- members, 281
- nested namespaces, 279
- scope, 282-283
- System, 273
- System.Diagnostics, 879, 918
  - assertions, 886-887
  - built-in performance counters, 888-895
  - conditional debugging, 881-884
  - Debug class, 880-881
  - runtime tracing, 884-886
- System.Linq, 15
- System.Linq.Expression, 404
- System.ServiceModel, 15
- System.Windows, 15
- System.Workflow.Runtime, 15
- visibility, 282-283
- when to use, 281

#### **naming conventions, attributes, 348**

#### **native code, 854**

#### **natural relationships, objects, 177**

#### **navigation, ASP.NET applications, 603-609**

#### **nested namespaces, 279**

#### **.NET, 9-11, 17-18**

- COM components, communicating with, 866-869
- COM+ services, support for, 871-876
- components, exposing as COM
  - components, 869
- FTP file transfers, performing, 671-673, 675
- HTTP, working with, 669-671
- programming languages, 16
- SMTP mail, sending, 675-676
- sockets, implementing, 661-669
- standardization, 10

#### **.NET Framework**

- class library documentation, 974
- collections, 757
- framework class library types, 16
- framework types, 94-101
- help, receiving, 973-975
- processes, support, 818-823

- search engines, 975
- security utilities, 944
- template pattern, 773-777
- websites, 975

#### **network communications**

- FTP file transfers, performing, 671, 673, 675
- HTTP, working with, 669-671
- remoting, 695
  - basic remoting, 695-706
  - channels, 706-708
  - diagrams, 695
  - lifetime management, 709-711
  - remoting servers, 696-698
  - setup, 701-706
- SMTP mail, sending, 675-676
- sockets, implementing, 661-669

#### **New Item command (Add menu), 189**

#### **new processes, launching, 818-821**

#### **New Project Wizard (VS2008), running, 23-26**

#### **new threads, creating, 823-824**

#### **nongeneric collections, 366, 396**

- arrays, compared, 371-372
- generic collections, compared, 371-372

#### **not equal operator (!=), 54-55**

#### **nullable types, 102-103**

#### **NumericUpDown control (Windows Forms), 529**

## **O**

#### **object models, creating, VS2008 Class Designer, 749-754**

#### **object pooling, COM+ services, 875-876**

#### **object type, 82-83**

#### **object-oriented programming (OOP), 177**

#### **objects, 164, 177**

- 2D objects, creating, 536-539
- access modifiers, 188-189
- automatic memory management, 328-331
- cloning, 173-174
- communication, 177
- compositional relationships, 177
- configuring, VS2008 Class Designer, 750
- controlling, 335, 337

- data access, LINQ (Language Integrated Query), 410-414
- DataContext, modifying, 419-420
- DataSet
  - LINQ to, 458-459
  - reading data, 453-454
- DateTime objects
  - creating, 97
  - extracting parts of, 97-98
  - math, 98-99
  - string types, 99-101
- design patterns, 755
  - iterator pattern, 756-768
  - proxy pattern, 768-772
  - template pattern, 772-777
- designing, 163-175
- DOM (Document Object Model), XML DOM, 466-468
- encapsulation, managing, 177
- equality, checking, 173
- fields, 165-166
- finalization, 327-328
- GC (Garbage Collector), interacting with, 335-337
- GDI+, 536-537
- generic type object, 365
- Graphics object, 536-539
- hash values, obtaining, 173
- identifying, 177
- indexers, 169-170
- inheritance, 81, 315
- inheritance relationships, 177
- initialization, 320
  - default constructors, 323
  - inheritance, 324-325
  - instance constructors, 320-322
  - order of instantiation, 324-325
  - overloading constructors, 322-323
  - private constructors, 323-324
  - static constructors, 325
- initializers, 326-327
- lifetime, 319
- LINQ to XML objects, 468
- ListItem, 175
- members, 163-164
  - instances, 164-165
  - methods, 166-167
  - static members, 164-165
- natural relationships, 177
- partial types, 170
- PerformanceCounter, 888, 893-895
- polymorphism, 177
- properties, 167
  - auto-implemented properties, 168
  - declaring, 167
  - using, 167-168
  - VS2008 property snippet, 169
- proxy object, 771-772
- references, comparing, 172
- resource cleanup, 331-334
- size, considerations, 94
- static classes, 171-172
- strings, using as, 174-175
- System.Object class, 172-175
- ThreadPool, 825-826
- TimeSpan object, 98-99
- types, checking, 172
- viewing, VS2008 Class Designer, 744-746
- Obsolete attribute, 343**
- OnPaint method, overriding, 537, 539**
- OnStart method, implementing, 684-687**
- OnStop method, implementing, 687**
- OOP (object-oriented programming), 200**
  - abstraction, 177
  - classes, abstract classes, 288-290
  - coding methods, 202
    - defining, 202-203
    - local variables, 203-204
    - overloading, 210-212
    - parameters, 204-210
  - encapsulation, 184-190
    - access modifiers, 185-188
    - containment, 190
    - inheritance, 190
  - event-based programming, 249-250, 271
    - anonymous methods, 256-258
    - coding events, 258-271
    - defining event handlers, 258-259
    - delegates, 250-256
    - firing events, 263-265
    - implementing events, 261-263
    - modifying event add/remove methods, 266-271
    - registering for events, 259-261

- exception handling, 232-240
  - different exception types, 236-237
  - finally blocks, 234-235
  - recoveries, 240-243
  - try/catch block, 232-234
- exceptions
  - designing, 243-245
  - passing, 237-240
- inheritance, 178-184
  - base classes, 178-180
  - sealed classes, 183-184
  - versioning, 180-183
- interface types, defining, 291-293
- interfaces, 315
  - explicit implementation, 304-310
  - implementing, 291
  - implicit implementation, 293-304
  - inheritance, 312-314
  - mapping, 310-312
- namespaces, 274, 286
  - alias qualifiers, 283-284
  - avoiding conflicts, 275
  - code organization, 274
  - creating, 278-281
  - directives, 275-278
  - extern namespaces alias, 284-286
  - Java packages, 274
  - members, 281
  - scope, 282-283
  - visibility, 282-283
- operators, overloading, 213-227
- polymorphism, 190-200
  - indexers, 196-197
  - problem solving, 190-196
  - properties, 196
  - System.Object class member overrides, 197-200
- statements
  - checked statements, 245-248
  - unchecked statements, 245-248
- Open Standards Institute (OSI), .NET standardization, 10**
- OperationContract attributes, declaring, 730**
- operator iterators, 393-394**
- operators, 49**
  - as operator, 60-61
  - assignment operators, 59
  - associativity, 64-65
  - binary operators, 52
    - addition operator (+), 53
    - division operator (/), 52
    - modulus operator (%), 53
    - multiplication operator (\*), 52
    - subtraction operator (-), 53
  - checked operator, 62
  - compound assignment operators, 59
  - conversion operators, overloads, 218-227
  - custom operators, 201
  - custom reference type conversion operators, 225-227
  - custom value type conversion operators, 222-225
  - is operator, 60
  - logical operator overloads, custom types, 215-216
  - logical operators, 56
    - bitwise AND operator (&), 56
    - bitwise exclusive OR operator (^), 57
    - bitwise inclusive OR operator (|), 57
    - Boolean AND operator (&), 57
    - Boolean exclusive OR operator (^), 58
    - Boolean inclusive OR operator (|), 57
    - conditional AND operator (&&), 58
    - conditional logical operators, 218
    - conditional OR operator (||), 58-59
    - side effects, 59
  - mathematical operator overloads, custom types, 213-215
  - overloading, 213-218
  - postfix, 217
  - precedence, 64-65
  - prefix, 217
  - query operators, 427
    - aggregate operators, 439
    - concatenation operators, 438-439
    - conversion operators, 438
    - element operators, 437
    - equality operators, 436-437
    - filtering operators, 430-431
    - generation operators, 435-436
    - grouping operators, 434-435
    - join operators, 433-434
    - partitioning operators, 433
    - projection operators, 432

- quantifier operators, 431-432
- set operators, 428-430
- sorting operators, 427-428
- relational operators, 54
  - equal operator (==), 54
  - less than operator (<), 55
  - less than or equal operator (<=), 55
  - not equal operator (!=), 54-55
- sizeof( ), 61, 858-859
- stackalloc, 860-861
- ternary operator, 60
- typeof operator, 61
- unary operators, 49
  - bitwise complement operator (~), 52
  - decrement operator (--), 51
  - increment operator (++), 50-51
  - logical complement operator (!), 51
  - minus operator (-), 50
  - plus operator (+), 49
- unchecked operator, 62
- optimization, GC (Garbage Collector), 330-331**
- optional parameters, simulating, 212**
- order of instantiation, object initialization, 324-325**
- organizing code, namespaces, 274**
- OSI (Open Standards Institute), .NET standardization, 10**
- out parameters, methods, 207-209**
- output, strings, 115-116**
- overflow checking, 62**
- overloading**
  - constructors, object initialization, 322-323
  - methods, 210-212
  - operators, 213-218
  - overriding, compared, 210
- overloads**
  - conversion operators, 218-227
  - explicit conversions, 219-222
  - finding, IntelliSense, 106
  - implicit conversions, 219-222
- override modifier, compiler warnings, handling, 194**
- overriding**
  - IntelliSense, 195
  - methods, Windows services, 683-684
  - OnPaint method, 537, 539

- overloading, compared, 210
- System.Object class members, 197-200
  - Equals method, 198-199
  - GetHashCode method, 199
  - ToString method, 199-200

## P

- padding output, strings, 115-116**
- PadLeft method, 115-116**
- PadRight method, 115-116**
- page requests, ASP.NET, high-level view of, 584**
- Panel control (Windows Forms), 529**
- Panel server control (ASP.NET), 594**
- parameters**
  - attributes, 342
    - named parameters, 343
    - positional parameters, 342-343
  - lastName, 398
  - methods, 204
    - out parameters, 207-209
    - params parameters, 209-210
    - ref parameters, 205-207
    - value parameters, 204-205
  - optional parameters, simulating, 212
  - params parameter, methods, 209-210
  - threads, passing to, 824-825
  - values as variables, 452
- params parameters, methods, 209-210**
- Partial Class for a Windows Service listing (31.1), 681-682**
- partial methods, 227-228**
  - data handling logic, extending, 425-427
  - IntelliSense, 228
- partial types, objects, 170**
- partitioning operators, 433**
- passing exceptions, 237-240**
- Passing Exceptions: Exceptions4.cs listing (11.4), 237-239**
- PasswordBox control (WPF), 566**
- PasswordGenerator console application, 508**
- patterns**
  - Dispose pattern, 332-334
  - proxy pattern, 768-771
  - template pattern, 772-777



**Pen object (GDI+), Windows Forms applications, 536-537**

**perceived performance, comprehending, 585-586**

**performance analysis, sampling, 908, 913, 916-917**

**performance counters**

- built-in performance counters, accessing, 888, 893-895
- customized performance counters, building, 897, 902-907
- sampling performance counters, 908, 913, 916-917

**PerformanceCounter object, 888-895**

**Performing Reflection: Reflecting.cs listing (16.8), 352-355**

**permissions**

- code-based security, 934-935
- requests, code-based security, 937-940

**Permview.exe security utility, 944**

**Preverifx.exe security utility, 944**

**PictureBox control (Windows Forms), 529**

**PInvoke (Platform Invoke), 853, 864-866**

**Platform Invoke Demo: PInvokeDemo.cs listing (41.9), 864-865**

**playing media, Silverlight, 652-655**

**plumbing (low-level work), 11**

**plus operator (+), 49**

**Pointer Arithmetic: PointerArithmetic.cs listing (41.1), 856-858**

**pointers, 855-858**

**policies**

- levels, code-based security, 936-937
- security policy, implementing, 940-942

**polymorphic behavior, interfaces, simulating, 298-304**

**polymorphism, 190**

- indexers, 196-197
- objects, 177
- OOP (object-oriented programming), 190-200
- problem solving, 190-196
- properties, 196
- System.Object class members, overriding, 197-200

**positional parameters, attributes, 342-343**

**Positional Parameters listing (16.3), 342**

**positioning colors, console applications, 511-514**

**postfix operators, 217**

**precedence, operators, 64-65**

**PredictLocation method, 206**

**prefix operators, 217**

**prefixes, ADO.NET, 444-445**

**private access modifiers, encapsulation, 186**

**private constructors, object initialization, 323-324**

**procedures, 14**

**processes, 829**

- .NET support, 818-823
- attaching to, 156-159
- details, reading, 822
- existing processes, working with, 821-823
- killing, 822-823
- launching new processes, 818-821
- lists, obtaining, 821-822
- locating specific, 822
- streams, managing, 820-821
- threads, 823, 830
  - creating, 823-824
  - passing parameters to, 824-825
  - reader threads, 828-829
  - running code in, 824
  - synchronization, 826-829
  - ThreadPool object, 825-826
  - writer threads, 828-829

**processors, multicore processors, 817**

**ProcessStartInfo, setting, 820**

**program state, examining, 149-150**

**Program with Bugs listing (7.2), 152-155**

**Programmatic Remoting Channel Registration: RemotingProxyClient.cs listing (31.10), 708**

**programming**

- event-based programming, 249-250, 271
  - anonymous methods, 256-258
  - coding events, 258-267, 271
  - defining event handlers, 258-259
  - delegates, 250-252, 254-256
  - firing events, 263, 265
  - implementing events, 261, 263
  - modifying event add/remove methods, 266-267, 271
  - registering for events, 259-261

- objects, designing, 163-175
- OOP (object-oriented programming)
  - base classes, 178-180
  - checked statements, 245-248
  - classes, 180
  - encapsulation, 184-190
  - exception design, 243-245
  - exception handling, 232-243
  - inheritance, 178-184
  - namespaces, 274-286
  - polymorphism, 190-200
  - serialized classes, 183-184
  - unchecked statements, 245-248
  - versioning, 180-183
- programming errors, finding, debuggers, 152-156**
- programming languages, NET support, 16**
- programs**
  - assertions, making, 886-887
  - built-in performance counters, accessing, 888, 893-895
  - C# programs, writing simple, 19-23
  - compiling, 969-971
  - customized performance counters, building, 897, 902-907
  - debugger demo program, 147-148
  - debugging
    - Boolean switches, 883-884
    - conditional debugging, 881-884
    - runtime debugging, 879-881
  - information, discovering, 350-356
  - interacting with, 45-48
  - localization, 831
    - multiple locales, 843-845, 848-850
    - resource files, 831-843
  - performance analysis, sampling, 908, 913, 916-917
  - runtime trace facilities, 879
  - timers, implementing, 896-897
  - tracing, runtime tracing, 884, 886
  - values, returning, 47-48
- ProgressBar control (Windows Forms), 529**
- ProgressBar control (WPF), 566**
- Project folder, 25**
- projection operators, 432**
- projections, extracting, LINQ (Language Integrated Query), 411-412**
- projects**
  - ADO.NET Data Services, adding to, 492-493
  - setup projects
    - configuring, 950-954
    - creating, 947-954
  - Silverlight projects
    - creating in VS2008, 642-647
    - parts, 644-647
  - VS2008 projects
    - compiling, 30
    - creating, 23-32
  - WF (Windows Workflow) projects, starting, 797-798
  - Windows service projects, items, 680-683
- projects (VS2008), 26**
- promotions, automatic promotions, handling, 220**
- properties**
  - attributes, requirements, 348
  - interface types, defining, 292
  - objects, 167
    - auto-implemented properties, 168
    - declaring, 167
    - using, 167-168
  - VS2008 property snippet, 169
  - polymorphism, 196
- property accessors, reflection, 359**
- property iterators, 391**
- PropertyGrid control (Windows Forms), 529**
- protected access modifiers, encapsulation, 187**
- protected internal access modifiers, encapsulation, 188**
- providers, ADO.NET, 444-445**
- proxies, 737**
  - code, 720
- Proxy for Encapsulating TCP/IP Calls to a Server listing (36.2), 768-770**
- proxy object, 771-772**
- proxy pattern, 768-771**
  - proxy object, 771-772
- public access modifiers, encapsulation, 185**
- publishing, web applications, 961-966**
  - VS2008, 965-966

## Q

**qualification, class members, 179**

**quantifier operators, 431-432**

**quantifiers, regular expressions, 127**

**query operators (LINQ), 427**

- aggregate operators, 439
- concatenation operators, 438-439
- conversion operators, 438
- element operators, 437
- equality operators, 436-437
- filtering operators, 430-431
- generation operators, 435-436
- grouping operators, 434-435
- join operators, 433-434
- partitioning operators, 433
- projection operators, 432
- quantifier operators, 431-432
- set operators, 428-430
- sorting operators, 427-428

**query results, ordering, LINQ (Language Integrated Query), 412-413**

**querying**

- entities
  - Entity SQL, 480-481
  - LINQ to Data Services, 503
  - LINQ to Entities, 486-487
  - WebDataQuery, 499-501
- relational data, LINQ to SQL, 414-427
- through DataContext, 418-419

## R

**RAD (rapid application development), 516**

- drag-and-drop problems, 779-781

**RadioButton control (Windows Forms), 529**

**RadioButton control (WPF), 566-567**

**RadioButton server control (ASP.NET), 594**

**RadioButtonList server control (ASP.NET), 594**

**RangeValidator server control (ASP.NET), 594**

**rapid application development (RAD), 516, 779**

**reader threads, access, 828-829**

**reading**

- process details, 822
- resource files, 835-836
- values, IEnumerator interface, 762-763
- XML, 465-466
  - XPathDocument, 466-467

**Reading an XML Document with XmlTextReader listing (21.2), 465**

**ReadLine() method, 45**

**readonly fields, objects, 166**

**Rebuild Project (Build menu), 29**

**Rebuild Solution command (Build menu), 29**

**Recovering from Exceptions: ExceptionTester.cs listing (11.5), 240-242**

**Rectangle control (WPF), 567**

**ref parameters, methods, 205-207**

**reference types, 79-80**

- assignments, 88-91
- constraints, 386
- construction, 93
- conversions, 225-227
- finalization, 93
- inheritance, 92-93
- memory allocation, 86
- object size considerations, 94
- objects, arrays, 134
- value types, compared, 92-94

**references, objects, comparing, 172**

**reflection, 339, 349, 363**

- API hierarchy, 350
- attributes, 356-363
- performing, 352-356
- program information, discovering, 350-356
- property accessors, 359

**Reflection.Emit API, runtime assemblies, building with, 359-363**

**registering for events, 259-261**

**registry settings, creating, 951**

**regular expressions, 124-125**

- application, 127-129
- common character classes, 126
- operations, 125-126
- quantifiers, 127

**Regular Expressions Application listing (5.1), 127**

**RegularExpressionValidator** server control (ASP.NET), 594

**relational data, querying, LINQ to SQL, 414-427**

**relational operators, 54**

- equal operator (==), 54
- less than operator (<), 55
- less than or equal operator (<=), 55
- not equal operator (!=), 54-55

**relationships, objects, 177**

**Remote Leasing Demo: LeasingDemo.cs**  
listing (32.11), 709-711

**Remote Server Component Configuration File:**  
web.config listing (32.2), 697-698

**remoting, 695**

- basic remoting, 695-706
- channels, 706-708
- diagram, 695
- lifetime management, 709-711
- remoting server, 696-698
- setup, 701-706

**Remoting Client Configuration File:**  
HostedClient.exe.config listing (32.7), 704-705

**Remoting Client Demo: HostedClient.cs**  
listing (32.6), 704

**Remoting Server Demo:**  
BasicRemotingServer.cs listing (32.1), 697

**Remove method, 117**

**Remove method, modifying, 266-267, 271**

**Repeater** server control (ASP.NET), 594

**Replace method, 118**

**Representational State Transfer (REST) web**  
service APIs, 493

**requests, permissions, code-based security,**  
937-940

**RequiredFieldValidator** server control (ASP.NET), 594

**ReRegisterForFinalize** class member (GC), 335

**ResEditor** utility, graphical resource files,  
creating, 839-840

**Reset** method, iterator pattern, 766

**resetting IEnumerator** interface, 760-761

**ResGen** utility, resource files, creating, 831-834

**resource cleanup**

- ensuring, finally blocks, 234-235
- objects, 331-334

**resource files, 831**

- converting, 836-838
- creating, 831-834
- elements, 832
- graphical resource files, creating, 838-843
- multiple locales, 843-844
  - implementing, 844-849
  - obtaining resources, 849-850
- reading, 835-836
- writing, 834-835

**ResourceManager** class, 833-834

**REST (Representational State Transfer) web**  
service APIs, 493

**Rest of the Program: WebSites.cs** listing (12.7),  
268-271

**ResXGen** utility, graphical resource files,  
creating, 839

**Retrieving User Input from the Command-Line**  
listing (24.2), 510

**return statements, 76-77**

**returning values, 47-48**

**reusable types, FCL (Framework Class**  
Library), 14

**reusing code, FCL (Framework Class Library),**  
15-16

**RIAs (rich internet applications), 583**

- creating, Silverlight, 641-657

**RichTextBox** control (Windows Forms), 530

**RichTextBox** control (WPF), 567

**role-based security, 933, 942**

- GenericPrincipal, 943-944
- WindowsPrincipal, 942-943

**Role-Based Security with GenericPrincipal**  
listing (44.4), 943-944

**Role-Based Security with WindowsPrincipal**  
listing (44.3), 942-943

**running**

- GC (Garbage Collector), 329
- New Project Wizard (VS2008), 23-26
- Silverlight, 647
- VS2008 applications, 28-31
- VS2008 Setup Project Wizard, 947-948

**runtime assemblies, building with**  
**Reflection.Emit** API, 359-363

**runtime configuration, assemblies, 928-930**

**runtime debugging, 879**

Debug class, 880-881

**runtime trace facilities, 879**

**runtime tracing, System.Diagnostics namespace, 884-886**

## S

**safe code, 854**

**samples, data updates, 420**

**sampling performance analysis, 908, 913, 916-917**

**Sampling Client: SampleClient.cs**  
listing (42.17), 908, 913

**Save method, 363**

**saving assemblies, 362**

**scalability, 584-585**  
databases, 443

**scope**

assemblies, 925  
code, 63  
namespaces, 282-283

**script libraries, loading, 623-625**

**ScrollBar control (Windows Forms), 530**

**ScrollBar control (WPF), 567**

**ScrollView control (WPF), 568**

**sealed classes, inheritance, 183-184**

**search engines, .NET Framework, 975**

**searching arrays, 138-139**

**securing, ASP.NET sites, 612-613**

**security**

assemblies, 926-927  
CLR (Common Language Runtime), 12  
code-based security, 933-934  
code groups, 935-936  
evidence, 934  
permission requests, 937-940  
permissions, 934-935  
security policy implementation, 940-942  
security policy levels, 936-937  
role-based security, 933, 942  
GenericPrincipal, 943-944  
WindowsPrincipal, 942-943  
security utilities, 944

**security policy, implementing, 940-942**

**security utilities, 944**

**Secutil.exe security utility, 944**

**selecting**

ADO.NET Data Services entity items, 493-495  
entities, Entity SQL, 480-481

**semicolons, statements, 70**

**SendAlert method, 191, 194**

**sending**

email attachments, 676  
SMTP mail, 675-676

**Separator control (WPF), 568**

**sequence of values, iterators, 394-395**

**serialization, XML, 731**

**Server Component of System Performance**  
Counter Demo: OrderProcessor.cs  
listing (42.14), 893-895

**Server controls (ASP.NET), 593-595**

**Server Implementing a Custom Performance**  
Counter: CustomOrderProcessor.cs  
listing (42.16), 902-907

**servers**

remoting servers, 696-698  
socket servers, creating, 252-254, 662-665  
web servers  
deploying, 965  
setting up, 962-963

**service element, web services, 736**

**service references, creating, 737-738**

**ServiceContract attribute, declaring, 730**

**ServiceController, implementing, 691-693**

**ServiceInstaller, configuring, 690**

**ServiceProcessInstaller, configuring, 689**

**services, uninstalling before redeploying, 691**

**Session state (ASP.NET), 599**

**set operators, 428-430**

**Setreg.exe security utility, 944**

**setting breakpoints, code, 148-149**

**Setup Project Wizard (VS2008), running, 947-948**

**setup projects, 954**

configuring, VS2008, 950-954  
creating, VS2008, 947-954  
file types, 951

- launch conditions, 953
- UIs (user interfaces), 952
- Setup Wizard Include Files window, 948**
- Setup Wizard Project Outputs window, 948**
- Setup Wizard Project Type window, 948**
- Shared Source Common Language Infrastructure (SSCLI), 10**
- side effects, Boolean logical operators, 59
- Signcode.exe security utility, 944**
- Silverlight, 641, 657**
  - Add Silverlight Application window, 643
  - ASP.NET, relationship, 642
  - C# handler, adding, 648-649
  - class libraries, 644
    - files, 646-647
  - control, 644, 646
  - data, working with, 649-650, 652
  - DataGrid, 652
  - events, handling, 648-652
  - JavaScript, relationship, 642
  - media, playing, 652-655
  - MediaElement, manipulating, 653-655
  - obtaining, 642
  - projects
    - creating in VS2008, 642-647
    - parts, 644-647
  - RIAs (rich Internet applications), creating, 641-657
  - running, 647
  - UI elements, animating, 655-657
  - WPF (Windows Presentation Foundation), 641-642
  - XAML (Extensible Application Markup Language), 641-642
- Silverlight Control on a Web Form listing (29.1), 645-646**
- Silverlight User Control listing (29.2), 646**
- Simple C# Program listing (2.1), 20**
- simple C# programs, writing, 19-23**
- Simple Console Application listing (24.1), 508**
- Simple Debugging Example:**
  - PlainDebugDemo.cs listing (42.1), 880-881**
- Simple Exception: Exceptions.cs listing (11.1), 233**
- simplifying iterator pattern with iterators, 767-768**
- simulation**
  - optional parameters, 212
  - polymorphic behavior, interfaces, 298-304
- single-assembly N-Layer architectures**
  - desktop N-Layer single-assembly architecture, 785-789
  - web application notes N-Layer single-assembly, 789-792
- single-dimension arrays, 132-134**
- single-implementation inheritance, 93**
- single-line comments, code, 32-33**
- singly linked lists, implementing, generic types, 373-381**
- site layout, web.sitemap file (ASP.NET), 604**
- SiteMapPath, ASP.NET, 609**
- SiteOwner class, 190-192**
- sites, ASP.NET**
  - securing, 612-613
  - theming, 609-612
- sizeof operator, 61**
- sizeof() operator, 858-859**
- Skeleton XAML for a WPF Application listing (26.1), 549**
- skins, websites, creating, 610-611**
- Slider control (WPF), 569**
- SMTP mail, sending, 675-676**
- Sn.exe security utility, 944**
- SOAP, message formats, 717**
- socket clients, creating, 665-669**
- socket servers, creating, 252-254, 662-665**
- sockets, implementing, 661-669**
- software**
  - managed software, 9-10
  - scalability, 584-585
- Solution folder, 25**
- solutions (VS2008), 26**
- sorting**
  - ADO.NET Data Services entities, 497
  - arrays, 138-139
- sorting operators, 427-428**
- spaces, Windows service project names, 680**
- Split method, 118-119**
- Splitter control (Windows Forms), 530**
- splitting strings, 118-119**

- SQL functions, using, LINQ to SQL, 421
- SSCLI (Shared Source Common Language Infrastructure), 10
- stackalloc Demonstration: StackAllocDemo.cs listing (41.5), 860-861
- stackalloc operator, 860-861
- StackPanel control (WPF), 569
- StackPanel layout, WPF (Windows Presentation Foundation), 554
- stacks, 87
  - unwinding, CLR, 239
- standard numeric format strings, 108
- standard query operators (LINQ), 427
  - aggregate operators, 439
  - concatenation operators, 438-439
  - conversion operators, 438
  - element operators, 437
  - equality operators, 436-437
  - filtering operators, 430-431
  - generation operators, 435-436
  - grouping operators, 434-435
  - join operators, 433-434
  - partitioning operators, 433
  - projection operators, 432
  - quantifier operators, 431-432
  - set operators, 428-430
  - sorting operators, 427-428
- standardization, .NET, 10
- Start method, processes, launching, 818
- StartsWith method, 113-114
- startup configuration, assemblies, 927-928
- state management, 584-585
  - ASP.NET, 596-603
- state workflows
  - building, 803-813
  - event handling, 811-813
- statements, 62
  - break statements, 75-76
  - checked statements, 245-248
  - continue statements, 76
  - fixed, 861-864
  - goto statements, 74-75
  - if statements, 65-67
  - lock, 826-828

- loop statements, 69
  - do loops, 71-72
  - for loops, 72-73
  - foreach loops, 73
  - while loops, 70
- return statements, 76-77
- semicolons, 70
- switch statements, 67-69
- unchecked statements, 245-248
- using statement, 334
- static classes, objects, 171-172
- static constructors, object initialization, 325
- static members, objects, 164-165
- static types, 23
- StatusBar control (Windows Forms), 530
- StatusBar control (WPF), 569
- StatusStrip control, Windows Forms applications, 531-533
- stepping through code, 147-159
- Storeadm.exe security utility, 944
- stored procedures
  - calling
    - ADO.NET, 452
    - LINQ to SQL, 420-421
  - databases, modifying, 421-424
- streaming XML data, 462
- streams, processes, managing, 820-821
- string type, 43-44, 105-122
- string types, DateTime objects, converting between, 99-101
- StringBuilder class, 122-123
  - Append method, 123
  - AppendFormat method, 123
  - EnsureCapacity method, 123-124
  - ToString method, 124
- StringBuilder types, 105
- strings
  - characters, working with, 119-121
  - CLR string handling, intern pools, 121-122
  - comparing, 109-110
  - concatenating, 111-112
  - content, inspecting, 113-114
  - copying, 112-113
  - CSVs (comma-separated values), 118
  - custom numeric format strings, 108

- enums, converting between, 142-143
- equality, checking for, 110-111
- formatting, 106-109
- information, extracting, 114-115
- joining, 118-119
- modifying, 117-118
- objects, using as, 174-175
- output, padding and trimming, 115-116
- regular expressions, 124-125
  - application, 127-129
  - common character classes, 126
  - operations, 125-126
  - quantifiers, 127
- splitting, 118-119
- standard numeric format strings, 108
- StringBuilder class, 122-123
  - Append method, 123
  - AppendFormat method, 123
  - EnsureCapacity method, 123-124
  - ToString method, 124
- strong name attributes, assemblies, 924-926**
- StructLayout attribute, 343**
- structs**
  - interfaces
    - explicit implementation, 304-310
    - implicit implementation, 293-304
  - System.Enum struct, 142-145
- style, code, 37**
- style sheets, creating, 612**
- styles, WPF (Windows Presentation Foundation)**
  - applications, 578-580
- Substitution server control (ASP.NET), 595**
- subtraction operator (-), 53**
- SuppressFinalize class member (GC), 335**
- switch statements, 67-69**
- switches, Boolean switches, debugging with, 883-884**
- synchronization, threads, 826**
  - access, 828-829
  - lock statement, 826-828
  - Monitor class, 827-828
  - reader threads, 828-829
  - writer threads, 828-829

- syntax**
  - code, variables, 38-39
  - exception handling, try/catch block, 232-234
  - lambda expressions, 398
  - LINQ (Language Integrated Query), 410-411
- system libraries, runtime debugging and tracing facilities, 879**
- System namespace, 273**
- System Performance Counter Demo:**
  - OrderClient.cs listing (42.13), 888-893**
- System.Array class, 137**
  - array bounds, 137-138
  - searching, 138-139
  - sorting, 138-139
- System.DateTime type, 97-101**
- System.Diagnostics namespace, 918**
  - assertions, 886-887
  - built-in performance counters, accessing, 888, 893-895
  - conditional debugging, 881-884
  - customized performance counters, building, 897, 902-907
  - Debug class, 880-881
  - runtime debugging classes, 879
  - runtime tracing, 884-886
  - sampling, performance analysis, 908, 913-917
  - timers, implementing, 896-897
- System.Enum struct, 142-143**
  - type members, 144-145
- System.Guid, 95-96**
- System.Linq namespace, 15**
- System.Linq.Expression namespace, 404**
- System.Object class, 172-175**
  - hash values, obtaining, 173
  - members, overriding, 197-200
  - object equality, checking, 173
  - object references, comparing, 172
  - object types, checking, 172
  - objects
    - cloning, 173-174
    - using as strings, 174-175
- System.ServiceModel namespace, 15**
- System.Windows namespace, 15**
- System.Workflow.Runtime namespace, 15**



# T

**Tab control (WPF), 569**

**TabControl control (Windows Forms), 530**

**Table server control (ASP.NET), 595**

**targets, attributes, 344-348**

**TCP channels, HttpChannel, compared, 708**

**template pattern, 772-777**

.NET Framework, 773

implementing, 773-777

**ternary operator, 60**

**testing web services, 717**

**text, Windows Forms application, drawing, 537-539**

**TextBlock control (WPF), 570**

**TextBox control (Windows Forms), 530**

**TextBox control (WPF), 570**

**TextBox server control (ASP.NET), 595**

**theming, ASP.NET sites, 609-612**

**Thread.CurrentThread, 849**

**ThreadPool object, 825-826**

**threads, 823, 830**

code, running in, 824

creating, 823-824

managing, CLR (Common Language Runtime), 12

parameters, passing to, 824-825

reader threads, access, 828-829

synchronization, 826

access, 828-829

lock statement, 826-828

Monitor class, 827-828

ThreadPool object, 825-826

writer threads, access, 828-829

**Timer control (ASP.NET AJAX), 628**

**Timer control (Windows Forms), 530**

**timers, implementing, 896-897**

**TimeSpan object, 98-99**

**ToCharArray methods, 119-121**

**ToLower method, 118**

**ToolBar control (Windows Forms), 530**

**ToolBar control (WPF), 570**

**ToolBarPanel control (WPF), 570-571**

**ToolBarTray control (WPF), 571**

**ToolStrip control, Windows Forms applications, 531-533**

**ToolTip control (Windows Forms), 530**

**ToString method, 124, 175, 200**

System.Object class members, overriding, 199-200

**ToUpper method, 118**

**Trace class, 879**

**TraceSwitch class, runtime tracing, 884, 886**

**TraceSwitch Class Demo: TraceSwitchDemo.cs listing (42.8), 885**

**TraceSwitch entry in Config File:**

TraceSwitchDemo.config listing (42.9), 886

**TrackBar control (Windows Forms), 531**

**traditional compilation, 10**

**transactions, COM+ services, 873-874**

**traversing, ADO.NET Data Services entity associations, 497-498**

**TrayIcon control (Windows Forms), 531**

**TreeView, ASP.NET, implementing, 606-608**

**TreeView control (Windows Forms), 531**

**TreeView control (WPF), 571**

**Trim method, 115-116**

**trimming output, strings, 115-116**

**try/catch block, exception handling, 232-234**

**try/catch blocks, 667**

**txt Resource File: strings.txt listing (40.1), 832**

**type loading, CLR (Common Language Runtime), 12**

**type safety, CLR (Common Language Runtime), 12**

**typeof operator, 61**

**types, 15, 23**

.NET framework types, 94-101

Assembly, 356

built-in types, 15

code, 39-44

bool type, 39

floating-point types, 42-43

integral types, 39-41

string type, 43-44

collection types, 372

CTS (Common Type System), 16-17

- custom types, 15
  - logical operator overloads, 215-216
  - mathematical operator overloads, 213-215
- enum types, 139-142
  - System.Enum struct, 142-145
- framework class library types, 16
- generic type object, 365
- generic types
  - benefits of, 366-372
  - building, 372-387
  - defining, 384
- interface types, defining, 291-293
- iterator types, 388
- LINQ (Language Integrated Query) types, 15
- modules, skipping, 356
- nullable types, 102-103
- objects
  - checking, 172
  - System.Object class, 172-175
- partial types, objects, 170
- reference type, assignment, 88-91
- reference types, 79-80, 92-94
  - construction, 93
  - finalization, 93
  - inheritance, 92-93
  - memory allocation, 86
  - object size considerations, 94
- reusable types, FCL (Framework Class Library), 14
- static types, 23
- string, 105-122
- string types, 99-101
- StringBuilder, 105
- System.DateTime type, 97-101
- Unified Type System, 80-81
  - boxing, 83-85
  - object type, 82-83
  - unboxing, 83-85
- value types, 79-80, 92-94
  - assignment, 91-92
  - construction, 93
  - finalization, 93
  - inheritance, 92-93
  - object size considerations, 94
- values types, memory allocation, 86-87

## U

- UIs (user interfaces)**
  - elements, animating, 655-657
  - setup projects, 952
- UML (Unified Modeling Language), 81**
- unary operators, 49**
  - bitwise complement operator (~), 52
  - decrement operator (--), 51
  - increment operator (++), 50-51
  - logical complement operator (!), 51
  - minus operator (-), 50
  - plus operator (+), 49
- unboxing value type variables, 83-85**
- unchecked operator, 62**
- unchecked statements, 245-248**
- unchecked Statements: unchecked.cs listing (11.8), 246-247**
- Unified Modeling Language (UML), 81**
- Unified Type System, 80-81**
  - boxing, 83-85
  - object type, 82-83
  - unboxing, 83-85
- Uniform control (WPF), 571**
- UniformGrid layout, WPF (Windows Presentation Foundation), 555**
- uninstalling services before redeploying, 691**
- unmanaged code, 854**
- unsafe code, 853-854**
  - configuring, 61
  - fixed statement, 861-864
  - pointers, 855-858
  - recognizing, 854
  - sizeof( ) operator, 858-859
  - stackalloc operator, 860-861
- unwinding stacks, CLR, 239**
- UpdatePanel control (ASP.NET AJAX), 625-627**
- UpdateProgress control (ASP.NET AJAX), 627-628**
- UpdateSite method, 205**
- updating**
  - data
    - ADO.NET, 451
    - after running samples, 420
    - entities, ADO.NET Data Services, 501-502

**Uploading a File to an FTP Server listing (30.4), 671-673**  
**user security policy level, 936**  
**users, interacting with, console applications, 508-510**  
**Using a Single Attribute listing (16.1), 340-341**  
**Using a Web Service: WebServiceClient.cs listing (33.4), 722**  
**using directive, namespaces, 275-276**  
**Using Graphical Resources: GraphRes.cs listing (40.6), 840-842**  
**Using Multiple Attributes listing (16.2), 341**  
**Using Resources: StringRes.cs listing (40.2), 833**  
**using statement, 334**  
**Using the Grid Layout Control listing (26.3), 556-558**  
**Using the sizeof() Operator listing (41.3), 858-859**  
**Using Two Classes with the Same Interface listing (14.5), 302-303**

## V

**ValidateUrl method, 211-212**  
**ValidationSummary server control (ASP.NET), 595**  
**value parameters, methods, 204-205**  
**value type, assignment, 91-92**  
**value type constraints, 387**  
**value type variables**  
**value types, 79-80**  
     assignments, 91-92  
     construction, 93, 387  
     finalization, 93  
     inheritance, 92-93  
     memory allocation, 86-87  
     object size considerations, 94  
     reference types, compared, 92-94  
     variables  
         boxing, 83-85  
         unboxing, 83-85

**values**  
     parameters as variables, 452  
     programs, returning, 47-48  
     reading, IEnumerator interface, 762-763  
**variables**  
     code, 38-39  
     local variables, methods, 203-204  
     MyCalc, 251  
     parameter values as, 452  
     types, 15  
     value type variables  
         boxing, 83-85  
         unboxing, 83-85  
**version redirection, assemblies, 928-929**  
**versioning**  
     assemblies, 925-926  
     inheritance, 180-183  
**View server control (ASP.NET), 595**  
**ViewBox control (WPF), 572**  
**viewing**  
     ADO.NET Data Services entity sets, 493  
     data, ADO.NET, 447-450  
     objects, VS2008 Class Designer, 744-746  
**ViewState (ASP.NET), 599**  
**virtual directories, setting up, 963-964**  
**visibility, namespaces, 282-283**  
**visual design environments, Windows Forms applications, 519**  
**Visual Designer, Windows Forms applications, 521-527**  
**Visual Studio 2008 (VS2008), 23**  
**visualizing code, VS2008 Class Designer, 743-748**  
**VS2005, 643**  
**VS2008**  
     Add New Item window, 28  
     applications  
         building, 28-31  
         debugging, 147-159  
         running, 28-31  
     ASP.NET projects, creating, 586-588  
     Autos window, 151  
     Build Menu, 28  
     Class Designer, 743, 754  
         object model creation, 749-754  
         visualizing code, 743-748

- code
  - commenting, 32-35
  - stepping through, 150-151
- coding in, 27-28
- command-line options, 46
- EDM (Entity Data Model), 476-480
- Help index, 974
- item template defaults, 189
- Locals window, 149
- namespaces, managing, 286
- New Project Wizard, running, 23-26
- projects, 26
  - compiling, 30
  - creating, 23-32
- property snippet, objects, 169
- Setup Project Wizard, running, 947-948
- setup projects
  - configuring, 950-954
  - creating, 947-954
- Silverlight, creating projects in, 642-647
- solutions, 26
- Watch window, 150
- WCF applications, creating, 726-727
- web applications, publishing, 965-966
- web references, creating, 723
- web services, building, 716
- Windows Forms applications, 519-527
  - files, 520
  - visual design environment, 519
  - Visual Designer, 521-527
- Windows service projects
  - coding, 683-688
  - creating, 680-683
- Workflow application, starting, 797-798

**VS2008 (Visual Studio 2008), 19**

## W

- WaitForPendingFinalizers class member (GC), 335**
- warnings, compilers, handling, 194**
- Watch window (VS2008), 150**
- WCF (Windows Communications Foundation), 15, 492**
  - applications, creating, 726-727

- web services
  - configuring, 734-737
  - consuming, 737-739
  - contracts, 727-732
  - creating, 725-739
  - implementing logic, 732-734
  - interface creation, 727-730

**WCF Web Service Implementation listing (34.2), 732-734**

**web application notes N-Layer single-assembly architecture, 789-792**

**web applications, 619-620**

- ASP.NET AJAX web applications
  - controls, 625-635
  - life cycles, 621-623
  - loading custom script libraries, 623-625
  - setting up, 620-621
- ASP.NET applications, 588-593
  - controls, 593-596
  - data binding, 614-617
  - navigation, 603-609
  - securing, 612-613
  - state management, 596-603
  - theming, 609-612
- ASP.NET projects, starting, 586-588
- components, 961-962
- model, 583-586
- publishing, 961-966
- RIAs (rich Internet applications), creating, 641-657
- websites, compared, 643

**Web Form with Multiple Controls listing (27.1), 591-592**

**web forms, 588**

**web project for N-Layer multiple-assembly architecture, 794-795**

**web references, creating, VS2008, 723**

**web servers**

- deploying, 965
- setting up, 701-702, 962-963

**Web Service Code: BasicWebService.asmx.cs listing (33.2), 715**

**Web Service Header: BasicWebService.asmx listing (33.1), 714**

**web services, 713**

- ASP.NET AJAX, calling, 635-640
- basic web services, 714-716

- behavior element, 736-737
- building, VS2008, 716
- client code, writing code to call, 738-739
- configuring, 734-737
- consuming, 737-739
- contracts, creating, 727-732
- creating, WCF, 725-739
- endpoint element, 736
- HTTP message services, 719
- Information, viewing, 716-719
- invocation results, 717
- logic, implementing, 732-734
- service element, 736
- SOAP message format, 717
- technologies, 713-714
- testing, 717
- using, 719-723
- WCF web services, interface creation, 727-730
- web service proxies, creating, 719, 721
- web.sitemap file (ASP.NET), site layout, 604**
- WebDataGen.exe-generated classes, using, 503-504**
- WebDataQuery, entities, querying, 499-501**
- WebForms, MediaPlayer, adding to, 652-653**
- websites**
  - .NET Framework, 975
  - securing, ASP.NET, 612-613
  - web applications, compared, 643
- Welcome screen (Setup Project Wizard), 947**
- WF (Windows Workflow), 797**
  - projects, starting, 797-798
  - workflows
    - creating, 798-801
    - executing, 802-803
    - state workflows, 803-813
- WF (Windows Workflow) runtime, writing code in, 15**
- while loops, 70**
- windows, Windows Forms applications, 539-545**
  - communication, 540-543
- Windows Communications Foundation (WCF).**
  - See **WCF (Windows Communications Foundation)**
- Windows Forms, 14**
- Windows Forms Application Showing Fundamentals listing (25.1), 516-518**

## **Windows Forms applications**

- controls, 528, 531
  - DataGridView control, 534-536
  - ListBox control, 534
  - MenuStrip control, 531-533
  - StatusStrip control, 531-533
  - ToolStrip control, 531-533
- data binding, 533-536
- dialog boxes, 539-545
  - common dialog boxes, 543-545
  - modal dialog boxes, 539-540
  - modeless dialog boxes, 539-540
  - window communication, 540-543
- fundamentals, 516-519
- GDI+, 536-539
  - Brush object, 536-537
  - drawing text, 537-539
  - fonts, 537-539
  - Graphics object, 536-537
  - Pen object, 536-537
- VS2008 support, 519-527
  - files, 520
  - visual design environment, 519
  - Visual Designer, 521-527
- windows, 539-545
- Windows Forms Form1.cs File listing (25.3), 522**
- Windows Forms Form1.Designer.cs File listing (25.4), 523-525**
- Windows Forms Program.cs File listing (25.2), 521**
- Windows Presentation Foundation (WPF) libraries, 14-15**
- Windows service, method overrides, 683-684**
- Windows service applications, 679**
  - coding, 683-688
  - creating, VS2008, 680-683
  - items, 680-683
  - spaces, names, 680
- Windows Service Code listing (31.3), 684**
- Windows Service Entry Point listing (31.2), 682-683**
- Windows services**
  - communicating with, controllers, 691-693
  - configuring, 687-688
  - deploying, 690-691
  - installing, 688-691

- OnStart method, implementing, 684-687
- OnStop method, implementing, 687
- Windows Workflow (WF), 797**
  - runtime, writing code in, 15
- WindowsFormsHost control (WPF), 572-573**
- WindowsPrincipal, role-based security, 942-943**
- Wizard server control (ASP.NET), 595**
- wizards, New Project Wizard, running, 23-26**
- workflows**
  - creating, 798-801
  - executing, 802-803
  - hosts, communicating from, 805-810
  - state workflows
    - building, 803-813
    - event handling, 811-813
- WPF (Windows Presentation Foundation)**
  - applications, 547**
  - controls, 560
    - Border control, 560-561
    - Button control, 561
    - CheckBox control, 561
    - ComboBox control, 561
    - ContentControl control, 561
    - DockPanel control, 562
    - DocumentViewer control, 562
    - Ellipse control, 563
    - Expander control, 563
    - Frame control, 563
    - Grid control, 564
    - GridSplitter control, 564
    - GroupBox control, 564-565
    - Image control, 565
    - Label control, 565
    - ListBox control, 565
    - ListView control, 565
    - MediaElement control, 565
    - Menu control, 565-566
    - PasswordBox control, 566
    - ProgressBar control, 566
    - RadioButton control, 566-567
    - Rectangle control, 567
    - RichTextBox control, 567
    - ScrollBar control, 567
    - ScrollViewer control, 568
    - Separator control, 568
    - Slider control, 569
    - StackPanel control, 569
    - StatusBar control, 569
    - Tab control, 569
    - TextBlock control, 570
    - TextBox control, 570
    - ToolBar control, 570
    - ToolBarPanel control, 570-571
    - ToolBarTray control, 571
    - TreeView control, 571
    - UniformGrid control, 571
    - ViewBox control, 572
    - WindowsFormsHost, 572-573
    - WrapPanel control, 573
  - data binding, 574-578
  - event handling, 573-574
  - layout
    - canvas layout, 553
    - default alignment, 552
    - DockPanel layout, 559
    - explicit alignment, 552-553
    - Grid layout, 555-559
    - managing, 551-559
    - StackPanel layout, 554
    - UniformGrid layout, 555
    - WrapPanel layout, 553-554
  - lists of data, displaying, 575-578
  - Silverlight, 641-642
  - styles, using, 578-580
  - XAML, 548-551
- WPF (Windows Presentation Foundation)**
  - libraries, 15**
- WrapPanel control (WPF), 573**
- WrapPanel layout, WPF (Windows Presentation Foundation), 553-554**
- wrapped HTTP classes, 661**
- WriteLine method, 23**
- writer threads, access, 828-829**
- writing**
  - resource files, 834-835
  - simple C# programs, 19-23
  - XAML, 462-464
- Writing a Resource File: ResWrite.cs**
  - listing (40.3), 834-835**
- Writing an XML Document with XmlTextWriter**
  - listing (21.1), 462-464**

## X–Z

### XAML

- controls, 550-551
- Silverlight, 641-642
- WPF (Windows Presentation Foundation) applications, 548-551

### XHTML (Extensible HTML), 589

### XML (Extensible Markup Language), 461

- data, streaming, 462
- documents, creating, 468-470
- LINQ to XML
  - document creation, 468, 470
  - modifying documents, 472
  - namespaces, 470-471
  - objects, 468
  - querying documents, 471
  - reading documents, 471
- manipulating
  - LINQ to XML, 468-472
  - XmlDocument, 467-468
- reading, 465-466
  - XPathDocument, 466-467
- serialization, 731
- writing, 462-464
- XmlDocument, manipulating, 467-468
- XML DOM, 466-468

### XML documentation comments, code, 33-35

### XML documents

- modifying, LINQ to XML, 472
- querying, LINQ to XML, 471
- reading, LINQ to XML, 471

### XML DOM, 466-468

### Xml server control (ASP.NET), 595

### XmlDocument, manipulating, 467-468

### XPathDocument, reading, 466-467

### XPS documents, creating, 562

### yield identifier, 389

### You Can Use Objects Polymorphically with Generics listing (17.1), 370-371