Microsoft

多核编程全接触

框架设计（影印版）

3

THIRD EDITION

CLR via C#

Jeffrey Richter

东南大学出版社

Wintellect
Know how.

图书在版编目（CIP）数据

框架设计：第3版：英文/（美）瑞奇特（Richter, J.）著. —影印本. —南京：东南大学出版社，2010.10
书名原文：CLR Via C#, 3e
ISBN 978-7-5641-2414-4

I. ①框… II. ①瑞… III. ①C语言－程序设计－英文 IV. ①TP312

中国版本图书馆CIP数据核字（2010）第169647号

江苏省版权局著作权合同登记
图字：10-2010-296号

© Southeast University Press [2010].

Authorized the English edition of ***CLR via C#, 3rd Edition*** ©[Jeffrey Richter]. This English edition is published and sold by permission of O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

框架设计 第三版（影印版）

出版发行：东南大学出版社
地　　址：南京四牌楼2号　　邮编：210096
出 版 人：江 汉
网　　址：http://www.seupress.com
电子邮件：press@seu.edu.cn
印　　刷：扬中市印刷有限公司
开　　本：787毫米×980毫米　12开本
印　　张：74.5印张
字　　数：1356千字
版　　次：2010年10月第1版
印　　次：2010年10月第1次印刷
书　　号：ISBN 978-7-5641-2414-4
印　　数：1~1800册
定　　价：99.00元（册）

本社图书若有印装质量问题，请直接与读者服务部联系。电话（传真）：025-83792328

Foreword

At first, when Jeff asked me to write the foreword for his book, I was so flattered! He must really respect me, I thought. Ladies, this is a common thought process error—trust me, he doesn't respect you. It turns out that I was about #14 on his list of potential foreword writers and he had to settle for me. Apparently, none of the other candidates (Bill Gates, Steve Ballmer, Catherine Zeta-Jones, . . .) were that into him. At least he bought me dinner.

But no one can tell you more about this book than I can. I mean, Catherine could give you a mobile makeover, but I know all kinds of stuff about reflection and exceptions and C# language updates because he has been talking on and on about it for years. This is standard dinner conversation in our house! Other people talk about the weather or stuff they heard at the water cooler, but we talk about .NET. Even Aidan, our six-year-old, asks questions about Jeff's book. Mostly about when he will be done writing it so they can play something "cool." Grant (age 2) doesn't talk yet, but his first word will probably be "Sequential."

In fact, if you want to know how this all started, it goes something like this. About 10 years ago, Jeff went to a "Secret Summit" at Microsoft. They pulled in a bunch of industry experts (Really, how do you get this title? Believe me, this isn't Jeff's college degree), and unveiled the new COM. Late that night in bed (in our house, this is what we discuss in bed), he talked about how COM is dead. And he was enchanted. Lovestruck, actually. In a matter of days he was hanging around the halls of Building 42 on Microsoft's Redmond campus, hoping to learn more about this wonderful .NET. The affair hasn't ended, and this book is what he has to show for it.

For years, Jeff has told me about threading. He really likes this topic. One time, in New Orleans, we went on a two-hour walk, alone, holding hands, and he spoke the whole time about how he had enough content for a threading book: The art of threading. How misunderstood threading in Windows is. It breaks his heart, all those threads out there. Where do they all go? Why were they created if no one had a plan for them? These are the questions of the universe to Jeff, the deeper meanings in life. Finally, in this book, he has written it down. It is all here. Believe me folks, if you want to know about threading, no one has thought about it more or worked with it more than Jeff has. And all those wasted hours of his life (he can't get them back) are here at your disposal. Please read it. Then send him an e-mail about how that information changed your life. Otherwise, he is just another tragic literary figure whose life ended without meaning or fulfillment. He will drink himself to death on diet soda.

This edition of the book even includes a new chapter about the runtime serializer. Turns out, this is not a new breakfast food for kids. When I figured out it was more computer talk and not something to put on my grocery list, I tuned it out. So I don't know what it says, but it is in here and you should read it (with a glass of milk).

My hope is that now he is finished talking about garbage collection in theory and can get on with actually collecting our garbage and putting it on the curb. Seriously people, how hard is that?

Folks, here is the clincher—this is Jeffrey Richter's magnum opus. This is it. There will be no more books. Of course, we say this every time he finishes one, but this time we really mean it. So, 13 books (give or take) later, this is the best and the last. Get it fast, because there are only a limited number and once they are gone—poof. No more. Just like QVC or something. Back to real life for us, where we can discuss the important things, like what the kids broke today and whose turn is it to change the diapers.

Kristin Trace (Jeffrey's wife)

November 24, 2009

A typical family breakfast at the Richter household

Introduction

It was October 1999 when some people at Microsoft first demonstrated the Microsoft .NET Framework, the common language runtime (CLR), and the C# programming language to me. The moment I saw all of this, I was impressed and I knew that it was going to change the way I wrote software in a very significant way. I was asked to do some consulting for the team and immediately agreed. At first, I thought that the .NET Framework was an abstraction layer over the Win32 API and COM. As I invested more and more of my time into it, however, I realized that it was much bigger. In a way, it is its own operating system. It has its own memory manager, its own security system, its own file loader, its own error handling mechanism, its own application isolation boundaries (AppDomains), its own threading models, and more. This book explains all these topics so that you can effectively design and implement software applications and components for this platform.

I have spent a good part of my life focusing on threading, concurrent execution, parallelism, synchronization, and so on. Today, with multicore computers becoming so prevalent, these subjects are becoming increasingly important. A few years ago, I decided to create a book dedicated to threading topics. However, one thing led to another and I never produced the book. When it came time to revise this book, I decided to incorporate all the threading information in here. So this book covers the .NET Framework's CLR and the C# programming language, and it also has my threading book embedded inside it (see Part V, "Threading").

It is October 2009 as I write this text, making it 10 years now that I've worked with the .NET Framework and C#. Over the 10 years, I have built all kinds of applications and, as a consultant to Microsoft, have contributed quite a bit to the .NET Framework itself. As a partner in my own company, Wintellect (*http://Wintellect.com*), I have worked with numerous customers to help them design software, debug software, performance-tune software, and solve issues they have with the .NET Framework. All these experiences have really helped me learn the spots that people have trouble with when trying to be productive with the .NET Framework. I have tried to sprinkle knowledge from these experiences through all the topics presented in this book.

Who This Book Is For

The purpose of this book is to explain how to develop applications and reusable classes for the .NET Framework. Specifically, this means that I intend to explain how the CLR works and the facilities that it offers. I'll also discuss various parts of the Framework Class Library (FCL). No book could fully explain the FCL—it contains literally thousands of types now, and this number continues to grow at an alarming rate. Therefore, here I'm concentrating on the core types that every developer needs to be aware of. And while this book isn't specifically about Windows Forms, Windows Presentation Foundation (WPF), Silverlight, XML Web services,

此为试读,需要完整PDF请访问: www.ertongbook.com

Web Forms, and so on, the technologies presented in the book are applicable to *all* these application types.

The book addresses Microsoft Visual Studio 2010, .NET Framework version 4.0, and version 4.0 of the C# programming language. Since Microsoft tries to maintain a large degree of backward compatibility when releasing a new version of these technologies, many of the things I discuss in this book apply to earlier versions as well. All the code samples use the C# programming language as a way to demonstrate the behavior of the various facilities. But, since the CLR is usable by many programming languages, the book's content is still quite applicable for the non-C# programmer.

Note You can download the code shown in the book from Wintellect's Web site (*http://Wintellect.com*). In some parts of the book, I describe classes in my own Power Threading Library. This library is available free of charge and can also be downloaded from Wintellect's Web site.

Today, Microsoft offers several versions of the CLR. There is the desktop/server version, which runs on 32-bit x86 versions of Microsoft Windows as well as 64-bit x64 and IA64 versions of Windows. There is the Silverlight version, which is produced from the same source code base as the desktop/server version of the .NET Framework's CLR. Therefore, everything in this book applies to building Silverlight applications, with the exception of some differences in how Silverlight loads assemblies. There is also a "lite" version of the .NET Framework called the .NET Compact Framework, which is available for Windows Mobile phones and other devices running the Windows CE operating system. Much of the information presented in this book is applicable to developing applications for the .NET Compact Framework, but this platform is not the primary focus of this book.

On December 13, 2001, ECMA International (*http://www.ecma-international.org/*) accepted the C# programming language, portions of the CLR, and portions of the FCL as standards. The standards documents that resulted from this have allowed other organizations to build ECMA-compliant versions of these technologies for other CPU architectures, as well as other operating systems. In fact, Novell produces Moonlight (*http://www.mono-project.com/Moonlight*), an open-source implementation of Silverlight (*http://Silverlight.net*) that is primarily for Linux and other UNIX/X11-based operating systems. Moonlight is based on the ECMA specifications. Much of the content in this book is about these standards; therefore, many will find this book useful for working with any runtime/library implementation that adheres to the ECMA standard.

Note My editors and I have worked hard to bring you the most accurate, up-to-date, in-depth, easy-to-read, painless-to-understand, bug-free information. Even with this fantastic team assembled, however, things inevitably slip through the cracks. If you find any mistakes in this book (especially bugs) or have some constructive feedback, I would greatly appreciate it if you would contact me at *JeffreyR@Wintellect.com*.

Dedication

To Kristin Words cannot express how I feel about our life together. I cherish our family and all our adventures. I'm filled each day with love for you.

To Aidan (age 6) and Grant (age 2) You both have been an inspiration to me and have taught me to play and have fun. Watching the two of you grow up has been so rewarding and enjoyable for me. I am lucky to be able to partake in your lives. I love and appreciate you more than you could ever know.

Acknowledgments

I couldn't have written this book without the help and technical assistance of many people. In particular, I'd like to thank my family. The amount of time and effort that goes into writing a book is hard to measure. All I know is that I could not have produced this book without the support of my wife, Kristin, and my two sons, Aidan and Grant. There were many times when we wanted to spend time together but were unable to due to book obligations. Now that the book project is completed, I really look forward to adventures we will all share together.

For this book revision, I truly had some fantastic people helping me. Christophe Nasarre, who I've worked with on several book projects, has done just a phenomenal job of verifying my work and making sure that I'd said everything the best way it could possibly be said. He has truly had a significant impact on the quality of this book. As always, the Microsoft Press editorial team is a pleasure to work with. I'd like to extend a special thank you to Ben Ryan, Valerie Woolley, and Devon Musgrave. Also, thanks to Jean Findley and Sue McClung for their editing and production support.

Support for This Book

Every effort has been made to ensure the accuracy of this book. As corrections or changes are collected, they will be added to a Microsoft Knowledge Base article accessible via the Microsoft Help and Support site. Microsoft Press provides support for books, including instructions for finding Knowledge Base articles, at the following Web site:

http://www.microsoft.com/learning/support/books/

If you have questions regarding the book that are not answered by visiting the site above or viewing a Knowledge Base article, send them to Microsoft Press via e-mail to mspinput@microsoft.com.

Please note that Microsoft software product support is not offered through these addresses.

We Want to Hear from You

We welcome your feedback about this book. Please share your comments and ideas via the following short survey:

http://www.microsoft.com/learning/booksurvey

Your participation will help Microsoft Press create books that better meet your needs and standards.

Note We hope that you will give us detailed feedback via our survey. If you have questions about our publishing program, upcoming titles, or Microsoft Press in general, we encourage you to interact with us via Twitter at *http://twitter.com/MicrosoftPress*. For support issues, use only the e-mail address shown above.

Table of Contents

Foreward xiii
Introduction xv

Part I **CLR Basics**

1 **The CLR's Execution Model** 1
Compiling Source Code into Managed Modules 1
Combining Managed Modules into Assemblies 5
Loading the Common Language Runtime 6
Executing Your Assembly's Code 9
IL and Verification 15
Unsafe Code 16
The Native Code Generator Tool: NGen.exe 18
The Framework Class Library 20
The Common Type System 22
The Common Language Specification 25
Interoperability with Unmanaged Code 29

2 **Building, Packaging, Deploying, and Administering Applications and Types** 31
.NET Framework Deployment Goals 32
Building Types into a Module 33
Response Files 34
A Brief Look at Metadata 36
Combining Modules to Form an Assembly 43
Adding Assemblies to a Project by Using the Visual Studio IDE 49
Using the Assembly Linker 50
Adding Resource Files to an Assembly 52
Assembly Version Resource Information 53
Version Numbers 57
Culture 58
Simple Application Deployment (Privately Deployed Assemblies) 59
Simple Administrative Control (Configuration) 61

What do you think of this book? We want to hear from you!

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

www.microsoft.com/learning/booksurvey/

3 Shared Assemblies and Strongly Named Assemblies 65

Two Kinds of Assemblies, Two Kinds of Deployment 66
Giving an Assembly a Strong Name 67
The Global Assembly Cache 73
Building an Assembly That References a Strongly Named Assembly 75
Strongly Named Assemblies Are Tamper-Resistant 76
Delayed Signing 77
Privately Deploying Strongly Named Assemblies 80
How the Runtime Resolves Type References 81
Advanced Administrative Control (Configuration) 84
Publisher Policy Control 87

Part II Designing Types

4 Type Fundamentals 91

All Types Are Derived from **System.Object** 91
Casting Between Types 93
Casting with the C# **is** and **as** Operators 95
Namespaces and Assemblies 97
How Things Relate at Runtime 102

5 Primitive, Reference, and Value Types 113

Programming Language Primitive Types 113
Checked and Unchecked Primitive Type Operations 117
Reference Types and Value Types 121
Boxing and Unboxing Value Types 127
Changing Fields in a Boxed Value Type by Using Interfaces (and Why You Shouldn't Do This) 140
Object Equality and Identity 143
Object Hash Codes 146
The **dynamic** Primitive Type 148

6 Type and Member Basics 155

The Different Kinds of Type Members 155
Type Visibility 158
Friend Assemblies 159
Member Accessibility 160
Static Classes 162
Partial Classes, Structures, and Interfaces 164
Components, Polymorphism, and Versioning 165
How the CLR Calls Virtual Methods, Properties, and Events 167
Using Type Visibility and Member Accessibility Intelligently 172
Dealing with Virtual Methods When Versioning Types 175

7 Constants and Fields 181

Constants 181
Fields 183

8 Methods 187
Instance Constructors and Classes (Reference Types) 187
Instance Constructors and Structures (Value Types) 191
Type Constructors 194
Type Constructor Performance 198
Operator Overload Methods 200
Operators and Programming Language Interoperability 203
Conversion Operator Methods 204
Extension Methods 207
Rules and Guidelines 210
Extending Various Types with Extension Methods 211
The Extension Attribute 213
Partial Methods 213
Rules and Guidelines 216

9 Parameters 219
Optional and Named Parameters 219
Rules and Guidelines 220
The **DefaultParameterValue** and **Optional** Attributes 222
Implicitly Typed Local Variables 223
Passing Parameters by Reference to a Method 225
Passing a Variable Number of Arguments to a Method 231
Parameter and Return Type Guidelines 233
Const-ness 235

10 Properties 237
Parameterless Properties 237
Automatically Implemented Properties 241
Defining Properties Intelligently 242
Object and Collection Initializers 245
Anonymous Types 247
The **System.Tuple** Type 250
Parameterful Properties 252
The Performance of Calling Property Accessor Methods 257
Property Accessor Accessibility 258
Generic Property Accessor Methods 258

11 Events 259
Designing a Type That Exposes an Event 260
Step #1: Define a type that will hold any additional information that should be sent to receivers of the event notification 261
Step #2: Define the event member 262
Step #3: Define a method responsible for raising the event to notify registered objects that the event has occurred 263
Step #4: Define a method that translates the input into the desired event 266
How the Compiler Implements an Event 266

此为试读，需要完整PDF请访问：www.ertongbook.com

Designing a Type That Listens for an Event 269
Explicitly Implementing an Event 271

12 Generics 275
Generics in the Framework Class Library 280
Wintellect's Power Collections Library 281
Generics Infrastructure 282
Open and Closed Types 283
Generic Types and Inheritance 285
Generic Type Identity 287
Code Explosion 288
Generic Interfaces 289
Generic Delegates 290
Delegate and Interface Contravariant and Covariant Generic Type Arguments 291
Generic Methods 293
Generic Methods and Type Inference 294
Generics and Other Members 296
Verifiability and Constraints 296
Primary Constraints 299
Secondary Constraints 300
Constructor Constraints 301
Other Verifiability Issues 302

13 Interfaces 307
Class and Interface Inheritance 308
Defining an Interface 308
Inheriting an Interface 310
More About Calling Interface Methods 312
Implicit and Explicit Interface Method Implementations (What's Happening Behind the Scenes) 314
Generic Interfaces 315
Generics and Interface Constraints 318
Implementing Multiple Interfaces That Have the Same Method Name and Signature 319
Improving Compile-Time Type Safety with Explicit Interface Method Implementations 320
Be Careful with Explicit Interface Method Implementations 322
Design: Base Class or Interface? 325

Part III Essential Types

14 Chars, Strings, and Working with Text 327
Characters 327
The `System.String` Type 330
Constructing Strings 330
Strings Are Immutable 333
Comparing Strings 334

String Interning ... 340
String Pooling ... 343
Examining a String's Characters and Text Elements ... 343
Other String Operations ... 346
Constructing a String Efficiently ... 346
Constructing a **StringBuilder** Object ... 347
StringBuilder Members ... 348
Obtaining a String Representation of an Object: **ToString** ... 350
Specific Formats and Cultures ... 351
Formatting Multiple Objects into a Single String ... 355
Providing Your Own Custom Formatter ... 356
Parsing a String to Obtain an Object: **Parse** ... 359
Encodings: Converting Between Characters and Bytes ... 361
Encoding and Decoding Streams of Characters and Bytes ... 367
Base-64 String Encoding and Decoding ... 368
Secure Strings ... 369

15 Enumerated Types and Bit Flags ... 373
Enumerated Types ... 373
Bit Flags ... 379
Adding Methods to Enumerated Types ... 383

16 Arrays ... 385
Initializing Array Elements ... 388
Casting Arrays ... 390
All Arrays Are Implicitly Derived from **System.Array** ... 392
All Arrays Implicitly Implement **IEnumerable**, **ICollection**, and **IList** ... 393
Passing and Returning Arrays ... 394
Creating Non-Zero–Lower Bound Arrays ... 395
Array Access Performance ... 396
Unsafe Array Access and Fixed-Size Array ... 401

17 Delegates ... 405
A First Look at Delegates ... 405
Using Delegates to Call Back Static Methods ... 408
Using Delegates to Call Back Instance Methods ... 409
Demystifying Delegates ... 410
Using Delegates to Call Back Many Methods (Chaining) ... 415
C#'s Support for Delegate Chains ... 419
Having More Control over Delegate Chain Invocation ... 419
Enough with the Delegate Definitions Already (Generic Delegates) ... 422
C#'s Syntactical Sugar for Delegates ... 423
Syntactical Shortcut #1: No Need to Construct a Delegate Object ... 424
Syntactical Shortcut #2: No Need to Define a Callback Method ... 424
Syntactical Shortcut #3: No Need to Wrap Local Variables in a Class Manually to Pass Them to a Callback Method ... 428
Delegates and Reflection ... 431

18 Custom Attributes 435
Using Custom Attributes 435
Defining Your Own Attribute Class 439
Attribute Constructor and Field/Property Data Types 443
Detecting the Use of a Custom Attribute 444
Matching Two Attribute Instances Against Each Other 448
Detecting the Use of a Custom Attribute Without Creating Attribute-Derived Objects 451
Conditional Attribute Classes 454

19 Nullable Value Types 457
C#'s Support for Nullable Value Types 459
C#'s Null-Coalescing Operator 462
The CLR Has Special Support for Nullable Value Types 463
Boxing Nullable Value Types 463
Unboxing Nullable Value Types 463
Calling **GetType** via a Nullable Value Type 464
Calling Interface Methods via a Nullable Value Type 464

Part IV Core Facilities

20 Exceptions and State Management 465
Defining "Exception" 466
Exception-Handling Mechanics 467
The **try** Block 468
The **catch** Block 469
The **finally** Block 470
The **System.Exception** Class 474
FCL-Defined Exception Classes 478
Throwing an Exception 480
Defining Your Own Exception Class 481
Trading Reliability for Productivity 484
Guidelines and Best Practices 492
Use **finally** Blocks Liberally 492
Don't **Catch** Everything 494
Recovering Gracefully from an Exception 495
Backing Out of a Partially Completed Operation When an Unrecoverable Exception Occurs—Maintaining State 496
Hiding an Implementation Detail to Maintain a "Contract" 497
Unhandled Exceptions 500
Debugging Exceptions 504
Exception-Handling Performance Considerations 506
Constrained Execution Regions (CERs) 509
Code Contracts 512

21 Automatic Memory Management (Garbage Collection) 519

Understanding the Basics of Working in a Garbage-Collected Platform 520
Allocating Resources from the Managed Heap 521
The Garbage Collection Algorithm 523
Garbage Collections and Debugging 527
Using Finalization to Release Native Resources 530
Guaranteed Finalization Using **CriticalFinalizerObject** Types 532
Interoperating with Unmanaged Code by Using **SafeHandle** Types 535
Using Finalization with Managed Resources 537
What Causes **Finalize** Methods to Be Called? 540
Finalization Internals 541
The Dispose Pattern: Forcing an Object to Clean Up 544
Using a Type That Implements the Dispose Pattern 548
C#'s **using** Statement 551
An Interesting Dependency Issue 554
Monitoring and Controlling the Lifetime of Objects Manually 555
Resurrection 566
Generations 568
Other Garbage Collection Features for Use with Native Resources 574
Predicting the Success of an Operation that Requires a Lot of Memory 578
Programmatic Control of the Garbage Collector 580
Thread Hijacking 583
Garbage Collection Modes 585
Large Objects 588
Monitoring Garbage Collections 589

22 CLR Hosting and AppDomains 591

CLR Hosting 592
AppDomains 594
Accessing Objects Across AppDomain Boundaries 597
AppDomain Unloading 609
AppDomain Monitoring 610
AppDomain First-Chance Exception Notifications 612
How Hosts Use AppDomains 612
Executable Applications 612
Microsoft Silverlight Rich Internet Applications 613
Microsoft ASP.NET Web Forms and XML Web Services Applications 613
Microsoft SQL Server 614
Your Own Imagination 614
Advanced Host Control 615
Managing the CLR by Using Managed Code 615
Writing a Robust Host Application 616
How a Host Gets Its Thread Back 617

23 Assembly Loading and Reflection 621

Assembly Loading 621
Using Reflection to Build a Dynamically Extensible Application 626
Reflection Performance 627
Discovering Types Defined in an Assembly 628
What Exactly Is a Type Object? 628
Building a Hierarchy of Exception-Derived Types 631
Constructing an Instance of a Type 632
Designing an Application That Supports Add-Ins 634
Using Reflection to Discover a Type's Members 637
Discovering a Type's Members 638
`BindingFlags`: Filtering the Kinds of Members That Are Returned 643
Discovering a Type's Interfaces 644
Invoking a Type's Members 646
Bind Once, Invoke Multiple Times 650
Using Binding Handles to Reduce Your Process's Memory Consumption 658

24 Runtime Serialization 661

Serialization/Deserialization Quick Start 662
Making a Type Serializable 667
Controlling Serialization and Deserialization 668
How Formatters Serialize Type Instances 672
Controlling the Serialized/Deserialized Data 673
How to Define a Type That Implements **`ISerializable`** when the Base Type Doesn't Implement This Interface 678
Streaming Contexts 680
Serializing a Type as a Different Type and Deserializing an Object as a Different Object 682
Serialization Surrogates 684
Surrogate Selector Chains 688
Overriding the Assembly and/or Type When Deserializing an Object 689

Part V Threading

25 Thread Basics 691

Why Does Windows Support Threads? 691
Thread Overhead 692
Stop the Madness 696
CPU Trends 699
NUMA Architecture Machines 700
CLR Threads and Windows Threads 703
Using a Dedicated Thread to Perform an Asynchronous Compute-Bound Operation 704
Reasons to Use Threads 706
Thread Scheduling and Priorities 708
Foreground Threads versus Background Threads 713
What Now? 715

26 Compute-Bound Asynchronous Operations 717

Introducing the CLR's Thread Pool 718
Performing a Simple Compute-Bound Operation 719
Execution Contexts 721
Cooperative Cancellation 722
Tasks 726
Waiting for a Task to Complete and Getting Its Result 727
Cancelling a Task 729
Starting a New Task Automatically When Another Task Completes 731
A Task May Start Child Tasks 733
Inside a Task 733
Task Factories 735
Task Schedulers 737
`Parallel`'s Static **`For`**, **`ForEach`**, and **`Invoke`** Methods 739
Parallel Language Integrated Query 743
Performing a Periodic Compute-Bound Operation 747
So Many Timers, So Little Time 749
How the Thread Pool Manages Its Threads 750
Setting Thread Pool Limits 750
How Worker Threads Are Managed 751
Cache Lines and False Sharing 752

27 I/O-Bound Asynchronous Operations 755

How Windows Performs I/O Operations 755
The CLR's Asynchronous Programming Model (APM) 761
The **`AsyncEnumerator`** Class 765
The APM and Exceptions 769
Applications and Their Threading Models 770
Implementing a Server Asynchronously 773
The APM and Compute-Bound Operations 774
APM Considerations 776
Using the APM Without the Thread Pool 776
Always Call the **`EndXxx`** Method, and Call It Only Once 777
Always Use the Same Object When Calling the **`EndXxx`** Method 778
Using **`ref`**, **`out`**, and **`params`** Arguments with **`BeginXxx`** and **`EndXxx`** Methods 778
You Can't Cancel an Asynchronous I/O-Bound Operation 778
Memory Consumption 778
Some I/O Operations Must Be Done Synchronously 779
`FileStream`-Specific Issues 780
I/O Request Priorities 780
Converting the **`IAsyncResult`** APM to a Task 783
The Event-Based Asynchronous Pattern 784
Converting the EAP to a **`Task`** 786
Comparing the APM and the EAP 788
Programming Model Soup 788

此为试读，需要完整PDF请访问：www.ertongbook.com