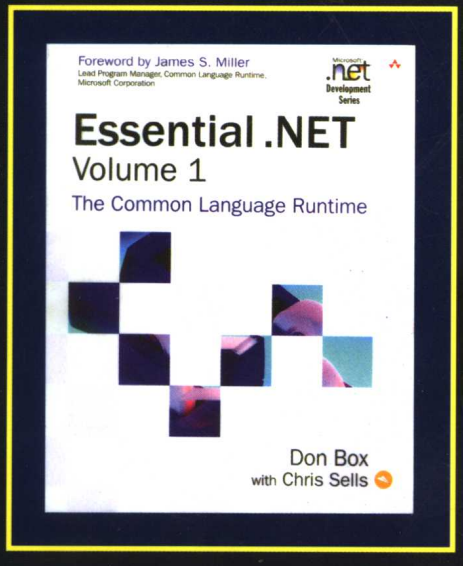
原 版 风 暴 系 列

PEARSON
Addison Wesley

Essential .NET
Volume 1: The Common Language Runtime

.NET 本质论
第一卷：公共语言运行库
（影印版）

[美] Don Box , Chris Sells 著



- 畅销书作者 Don Box 最新作品
- 微软公司 CLR 首席程序经理大力推荐
- 帮助读者快速完整地理解 CLR 工作机制

Pearson Education
中国电力出版社
www.infopower.com.cn

原 版 风 暴 系 列

Essential .NET
Volume 1: The Common Language Runtime

.NET 本质论
第一卷：公共语言运行库

（影印版）

[美] Don Box, Chris Sells 著

中国电力出版社

Essential .NET Volume 1: The Common Language Runtime
(ISBN 0-201-73411-7)
Don Box , Chris Sells

北京市版权局著作合同登记号：图字：01-2003-2439

图书在版编目（CIP）数据

.NET 本质论　第一卷：公共语言运行库 /（美）伯克斯，（美）赛欧司著. —影印本.
—北京：中国电力出版社，2003.11
（原版风暴系列）
ISBN 7-5083-1806-4
Ⅰ.N...　Ⅱ.①伯...②赛...　Ⅲ.计算机网络－程序设计－英文　Ⅳ.TP393
中国版本图书馆 CIP 数据核字（2003）第 086083 号

丛 书 名：原版风暴系列
书　　名：.NET 本质论　第一卷：公共语言运行库（影印版）
编　　著：（美）Don Box , Chris Sells
责任编辑：朱恩从
出版发行：**中国电力出版社**
　　地址：北京市三里河路6号　　邮政编码：100044
　　电话：（010）88515918　　传　　真：（010）88518169
印　　刷：汇鑫印务有限公司
开　　本：787×1092　1/16　　印　　张：27
书　　号：ISBN 7-5083-1806-4
版　　次：2003年11月北京第一版　　2003年11月第一次印刷
定　　价：48.00 元

Essential .NET
Volume 1

The Common Language Runtime

- Don Box
  with Chris Sells

Foreword

I WORKED FOR THE World Wide Web Consortium in its early days at the Massachusetts Institute of Technology (MIT). We decided we needed to learn about both COM and CORBA, at that time the leading frameworks for object-oriented programming over the network.

I called our Microsoft representative and asked who he could send to teach the Consortium staff the fundamentals of COM. I explained that the Consortium staff were very technical and were interested in the "why" of things at least as much as the "how." I explained that we were designing the next generation Web protocols and knew just about everything there was to know about application protocols on the Internet, and that we'd built some very large object-oriented programs in C++ and Objective C.

I fully expected him to suggest someone from Microsoft. Instead he said, "The best person in the world is Don Box." And that's how I met Don, one of the finest teachers I know. Don not only explained well and ran labs well and wrote well, but he never broke down under endless detailed questioning, ranging from the high-level "Why would you build the architecture that way?" to the low-level "Where does that bit ever get set?" So Don taught me (along with hundreds of thousands of others) what COM is, why COM is, and what the problems with COM are. Along the way, just off-hand 'cause he thought we might be interested, he also did an amazingly good comparison to the CORBA technology and to the Web technology we were designing and building.

I left the MIT and the Web Consortium to come to Microsoft to work on a top-secret project called "COM+ Services." My business card read "Program Manager, Garbage Collection and Related Rubbish" because all we could say (even internally at Microsoft) was that it had something to do with programming languages and distributed systems, and it had a garbage collector. I'm still on that same project at Microsoft, working almost four years to help design and build what is now the Common Language Runtime. I've worked on the IL design, four different JITs, the metadata, the garbage collector, the execution engine, and the ECMA (and soon, hopefully, ISO) standards.

I was sure that I'd be able to turn around and teach Don something I'd learned in this process (I'm not a bad teacher myself). But Don beat me to the punch. He told me to read this book. And (darn it) he's taught me stuff I didn't know about my own product! And I bet he'll teach you something, too.

Dr. James S. Miller
Microsoft Corporation
Lead Program Manager
Common Language Runtime

Preface

What Happened?

In 1998, Microsoft held a Professional Developer's Conference (PDC) in San Diego. COM luminary Charlie Kindel stood up in a general session and proclaimed "no more GUIDs—no more HRESULTs—no more IUnknown." He and Mary Kirtland proceeded to show the basic architecture of the CLR, then known as the COM+ Runtime. Later in the session, Nat Brown and David Stutz stood up and demonstrated cross-language inheritance using Visual Basic and Java. Attendees actually went home with CDs containing primitive versions of compilers that could reproduce this very odd demonstration. It is now February 2002, and this technology has finally shipped in release form.

There are two days that will forever demarcate the evolution of the Microsoft platform. On July 27, 1993, Windows NT 3.1 was released, marking the end of the DOS era. On February 13, 2002, the Common Language Runtime (CLR) was released as part of the .NET Framework, marking the end of the COM era.

The .NET Framework is a platform for software integration. Fundamentally, the .NET Framework provides two core integration technologies. The Common Language Runtime (CLR) is used to integrate software within a single operating system process. XML Web Services are used to integrate software at Internet scale. Both rely on similar ideas, that is,

strongly typed contracts and encapsulation. Fundamentally, though, they are two distinct technologies that one can elect to adopt independently of one another. It is completely reasonable to adopt XML Web Services prior to the CLR (in fact, many production Web services have already done this). It is also reasonable to adopt the CLR in the absence of XML Web Services in order to access CLR-specific features such as code access security or superior memory management facilities. Going forward, however, both the CLR and XML Web Services will be central to the Microsoft development platform, and it is only a matter of time before both of these technologies play a role in everyone's development experience.

The CLR and XML Web Services are both focused on strongly typed contracts between components. Both technologies require developers to describe component interactions in terms of type definitions or contracts. In both technologies, these contracts share two key ideas that tend to permeate their use: metadata and virtualization.

Both the CLR and XML Web Services rely on high-fidelity, ubiquitous, and extensible metadata to convey programmer intention. Metadata conveys the basic structure and type relationships to the developers who will consume a CLR component or XML Web Service.

Equally important, ubiquitous metadata informs the tools and underlying platform of what the component developers had in mind when they were authoring the code.

This metadata-directed "clairvoyance" allows the platform to provide richer support than would be possible if the component were completely opaque. For example, various aspects of object-to-XML mapping are captured in metadata for use by the CLR's XML serializer. How the developer intended the XML to look is conveyed through declarative metadata extensions rather than through explicit labor-intensive coding.

The second key idea that permeates CLR and XML Web Service contracts is the notion of virtualization. Both technologies emphasize the separation of semantic intentions from physical implementation details. Specifically, the metadata for both technologies work at an abstract structural level rather than in terms of low-level data representations and implementation techniques. When developers specify intercomponent contracts at this "virtual" level, the underlying platform is free to express

the contracts in the most appropriate manner available. For example, by expressing Web Service contracts in terms of an abstract data model, the plumbing is free to use an efficient binary data representation for performance or to use the text-based XML 1.0 representation for maximum interoperability.

Because contracts are virtualized, this specific detail of the contract can be bound at runtime based on post-development characteristics.

Because this volume focuses exclusively on the CLR, a working definition of the CLR is in order. The CLR is fundamentally a loader that brings your components to life inside an operating system process. The CLR replaces COM's `CoCreateInstance` and Win32's `LoadLibrary` as the primary loader for code.

The CLR loader provides a number of services beyond what COM and Win32 offered before it. The CLR loader is version-aware and provides flexible configuration of version policies and code repositories. The CLR loader is security-aware and is a critical part of the enforcement of security policy. The CLR loader is type-aware and provides a rich runtime environment for the explicit management and creation of types independent of programming language. In short, the CLR loader is an advanced component technology that supplants COM as Microsoft's primary in-memory integration strategy.

The CLR is made accessible through compilers that emit the CLR's new file format. Program language wonks view the CLR as providing key building blocks for compiler writers, building blocks that reduce the complexity of compiler implementations. In contrast, systems wonks often view programming languages as facades or "skins" over the underlying constructs of the CLR. The author falls firmly into the latter camp. However, programming languages are a necessary lens through which even low-level systems plumbers view the CLR. To that end, examples in this book are written in various programming languages because binary dumps of metadata and code are arcane to the point of being incomprehensible.

About This Book

I try very hard to make a book readable and accessible to a wide array of readers, but invariably, my terse writing style tends to make a "Don Box book" a challenge to get through. Experience has shown me that I am horrible at writing tutorials or primers. What I can do reasonably well is convey how I see the world in book form. To that end, it is not uncommon to need to read a Don Box book more than once to get the intended benefits.

As the previous paragraph implied, this book is by no means a tutorial. If you try to learn .NET Framework programming from a standing start using this book, the results may not be pretty. For readers looking for a good tutorial on .NET programming techniques or the C# language, please read Stan Lippman's *C# Primer* (Addison-Wesley, 2002) or Jeffery Richter's *Applied .NET Framework Programming* (Microsoft Press, 2002) before taking on this book.

This book is divided into two volumes. Volume 1 focuses on the Common Language Runtime. Volume 2 will focus [ST3]on XML Web Services. Although the two technologies share a fair number of core concepts, the thought of covering them both in a single book made my head spin.

This book was written against Version 1 of the CLR. Some of the internal techniques used by the CLR may evolve over time and may in fact change radically. In particular, the details of virtual method dispatch are very subject to change. They are included in this book largely as an homage to COM developers wondering where the vptr went. That stated, the basic concepts that are the focus of this book are likely to remain stable for years to come.

Throughout the book, I use assertions in code to reinforce the expected state of a program. In the CLR, assertions are performed using `System.Diagnostics.Debug.Assert`, which accepts a Boolean expression as its argument. If the expression evaluates to false, then the assertion has failed and the program will halt with a distinguished error message. For readability, all code in this book uses the short form, `Debug.Assert`, which assumes that the `System.Diagnostics` namespace prefix has been imported.

My perspective on .NET is fairly agnostic with respect to language. In my daily life, I use C# for about 50 percent of my CLR-based programming.

I use C++ for about 40 percent, and I resort to ILASM for the remaining 10 percent. That stated, most programming examples in this book use C# if for no other reason than it is often the most concise syntax for representing a particular concept or technique. Although some chapters may seem language-focused, none of them really is. The vast majority of this book could have used C++, but, given the tremendous popularity of C#, I elected to use C# to make this book as accessible as possible.

This book focuses on the Common Language Runtime and is divided into 10 chapters:

- Chapter 1—The CLR as a Better COM: This chapter frames the discussion of the CLR as a replacement for the Component Object Model (COM) by looking at the issues that faced COM developers and explaining how the CLR addresses those issues through virtualization and ubiquitous, extensible metadata.
- Chapter 2—Components: Ultimately, the CLR is a replacement for the OS and COM loaders. This chapter looks at how code is packaged and how code is loaded, both of which are done significantly differently than in the Win32 and COM worlds.
- Chapter 3—Type Basics: Components are containers for the code and metadata that make up type definitions. This chapter focuses on the CLR's common type system (CTS), including what constitutes a type and how types relate. This is the first chapter that contains significant chunks of source code.
- Chapter 4—Programming with Type: The CLR makes type a first-class concept in the programming model. This chapter is dedicated to the explicit use of type in CLR programs, with an emphasis on the role of metadata and runtime type information.
- Chapter 5—Instances: The CLR programming model is based on types, objects, and values. Chapter 4 focused on type; this chapter focuses on objects and values. Specifically, this chapter outlines the difference between these two instancing models, including how values and objects differ with respect to memory management.
- Chapter 6—Methods: All component interaction occurs through method invocation. The CLR provides a broad spectrum of techniques for making

method invocation an explicit act. This chapter looks at those techniques, starting with method initialization through JIT compilation and ending with method termination via strongly typed exceptions.

- Chapter 7—Advanced Methods: The CLR provides a rich architecture for intercepting method calls. This chapter dissects the CLR's interception facilities and its support for aspect-oriented programming. These facilities are one of the more innovative aspects of the CLR.
- Chapter 8—Domains: The CLR uses AppDomains rather than OS processes to scope the execution of code. To that end, this chapter looks at the role of AppDomains both as a replacement for the underlying OS's process model as well as an AppDomain's interactions between the assembly resolver or loader. Readers with Java roots will find the closest analog to a Java class loader here.
- Chapter 9—Security: One of the primary benefits of the CLR is that it provides a secure execution environment. This chapter looks at how the CLR loader supports the granting of privileges to code and how those privileges are enforced.
- Chapter 10—CLR Externals: The first nine chapters of this book are focused on what it means to write programs to the CLR's programming model. This concluding chapter looks at how one steps outside of that programming model to deal with the world outside of the CLR.

Acknowledgments

As always, my wife and children made unreasonable sacrifices for me to complete this book. In acknowledgment of this, I am taking at least a year off before I tackle another book project.

This book would never have been completed had it not been for Chris Sells. Chris was my writing partner on this project and pulled me through some very rocky times during 2001, both personally and professionally. Although Chris contributed no paragraphs to this book, his efforts are visible on every page.

Thanks go out to Jim Miller of the CLR team at Microsoft. Jim was nice enough to write the foreword for this book, and for that I am personally grateful. More importantly, however, Jim was the lead author on what I

consider the most important writing on the CLR, which is the five-part ECMA specification for the CLI. These five Microsoft Word documents reside in the Tools Developer's Guide folder in the .NET Framework SDK and give more insight into the CLR than anything else I have read. Readers familiar with my COM background may recall that I considered the COM specification to be must-reading. For the CLR, I believe the ECMA specifications are even more vital.

A lot of the insights I gained while writing this book were a result of conversations that took place on the DOTNET mailing list at http://discuss.develop.com/. This list is the center of the universe for developers and architects who are active in the CLR programming community. Specific listers who helped my thinking include Brian Harry, Jim Miller, Dennis Angeline, Steven Pratchner, Brent Rector, John Lam, Mike Woodring, Keith Brown, Peter Drayton, Brad Wilson, Jay Freeman, and Sam Gentile.

This book would have considerably more errors had it not been for my reviewers. Thanks go out to Dan Sullivan, Peter Drayton, Mike Woodring, Chris Sells, Stuart Celarier, Jay Freeman, Steve Vinoski, Stan Lippman, Robert Husted, Peter Jones, Mike Giroux, Vishwas Lele, Dan Green, Paul Gunn, and Fumiaki Yoshimatsu. Arguably the most critical review was from Brian Kernighan early on in the project. That review made me reevaluate where I wanted the book to go and how I was going to get it there. Had it not been for Brian's review, I am not certain I would have survived the process. Thanks, Brian.

As always, it takes a village to be Don Box. For this book, that village consisted of Helga Thomsen, Sandy Deason, Barbara Box, Judith Swerling, David Baum, David Stromberg, Martin Gudgin, Mike Woodring, Fritz Onion, Shannon Ahern Ikeda, Ron Sumida, Tim Ewald, and Aaron Skonnard.

Don Box
February 2002
Yarrow Point, WA

Contents

Figures