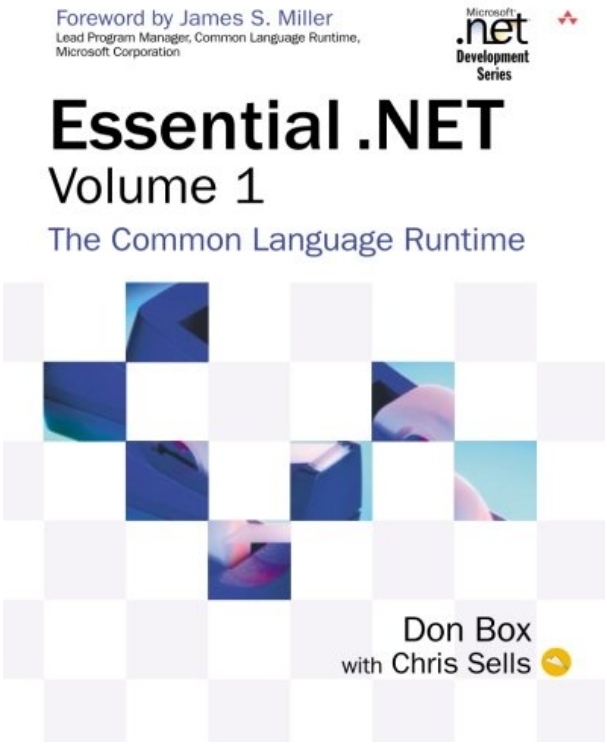
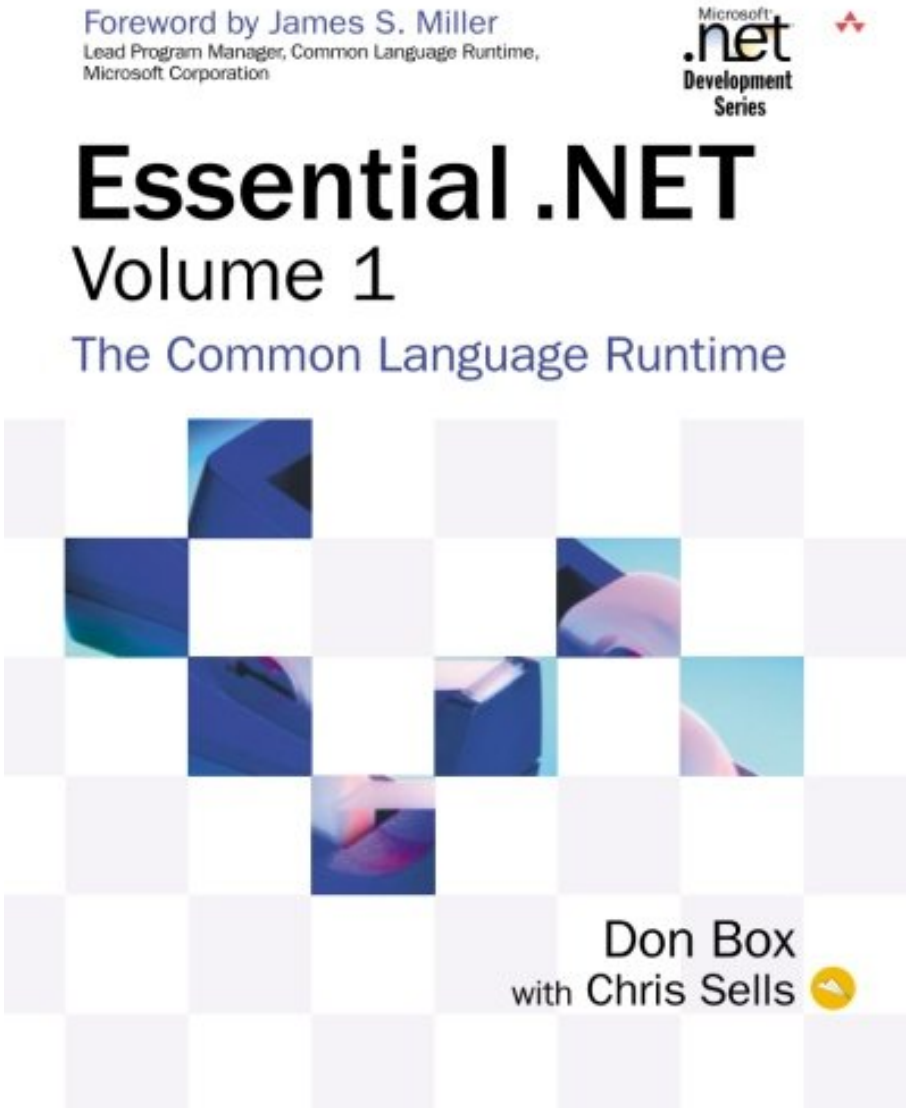
ESSENTIAL .NET, VOLUME I: THE COMMON LANGUAGE RUNTIME BY DON BOX, CHRIS SELLS





ESSENTIAL .NET, VOLUME I: THE COMMON LANGUAGE RUNTIME BY DON BOX, CHRIS SELLS PDF

So, when you require quickly that book **Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells**, it doesn't should await some days to receive the book Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells You can straight obtain guide to conserve in your device. Even you love reading this Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells all over you have time, you could appreciate it to read Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells It is undoubtedly practical for you that intend to get the more valuable time for reading. Why do not you spend five mins and also invest little money to get guide Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells here? Never let the new thing goes away from you.

From the Back Cover

"Don taught me stuff I didn't know about my own product! And I bet he'll teach you something, too."
—From the Foreword by James Miller, Lead Program Manager, Common Language Runtime, Microsoft Corporation

Essential .NET, Volume 1 , provides everything developers need to take full advantage of the power of Microsoft .NET. This book describes, in depth, the glue of the .NET Framework: the Common Language Runtime (CLR). Box and Sells explain the inner workings of the CLR—the rationale behind its design, the problems it solves, and the role of type in CLR programming—and show readers how to build better applications using the .NET Framework while gaining a more complete understanding of how the CLR works.

The book is packed with the practical detail and expert advice only Don Box can provide. Topics covered include:

- CLR's evolution
- Assemblies in the .NET Framework
- The CLR type system
- Programming with type
- Objects and values
- Methods
- Explicit method invocation
- Application domains
- Security
- Interoperability

Essential .NET, Volume 1 , is an authoritative guide to the Microsoft .NET Common Language Runtime.

Books in the Microsoft .NET Development Series are written and reviewed by the principal authorities and pioneering developers of the Microsoft .NET technologies, including the Microsoft .NET development team and DevelopMentor. Books in the Microsoft .NET Development Series focus on the design, architecture, and implementation of the Microsoft .NET initiative to empower developers and students everywhere with the knowledge they need to thrive in the Microsoft .NET revolution.

0201734117B10042002

About the Author

Don Box is a leading educator, recognized authority on the Component Object Model (COM), coauthor of the Simple Object Access Protocol (SOAP) specification, and coiner of the term "COM is Love." He recently joined Microsoft as an architect in the Microsoft® .NET Developer and Platform Evangelism Group.

Earlier in his career, Box cofounded DevelopMentor Inc., a component software think tank aimed at educating developers on the use of the COM, Java, and XML. A popular public speaker, Box is known for engaging audiences around the world, combining deep technical insight with often outrageous stunts.

0201734117AB06132002

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 XMLWeb 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 XMLWeb 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 XMLWeb 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 XMLWeb 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 ST3on XMLWeb 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.
- Sum 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.

0201734117P09272002

ESSENTIAL .NET, VOLUME I: THE COMMON LANGUAGE RUNTIME BY DON BOX, CHRIS SELLS PDF

When you are hurried of task due date and have no idea to get motivation, **Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells** book is among your remedies to take. Reserve Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells will provide you the right source as well as thing to get inspirations. It is not just about the tasks for politic company, administration, economics, and also other. Some purchased jobs making some fiction works likewise require inspirations to get over the job. As what you need, this Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells will possibly be your selection.

The way to get this publication *Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells* is very easy. You may not go for some areas and also invest the time to only find guide Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells Actually, you may not always get guide as you want. Yet below, only by search as well as discover Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells, you could get the lists of guides that you truly expect. In some cases, there are lots of publications that are revealed. Those publications certainly will amaze you as this Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells collection.

Are you considering mostly books Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells If you are still confused on which one of the book Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells that ought to be purchased, it is your time to not this website to seek. Today, you will certainly need this Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells as the most referred book and also most required book as resources, in various other time, you could take pleasure in for a few other publications. It will depend upon your willing requirements. Yet, we constantly suggest that books Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells can be a great infestation for your life.

ESSENTIAL .NET, VOLUME I: THE COMMON LANGUAGE RUNTIME BY DON BOX, CHRIS SELLS PDF

The first book in the Microsoft .NET Development Series, this text is designed to enable developers to take advantage of the full power available to them in Microsoft .NET.

- Sales Rank: #173404 in Books
- Color: White
- Published on: 2002-11-14
- Released on: 2002-11-04
- Original language: English
- Number of items: 1
- Dimensions: 9.00" h x 1.00" w x 7.30" l, 1.61 pounds
- Binding: Paperback
- 432 pages

From the Back Cover

"Don taught me stuff I didn't know about my own product! And I bet he'll teach you something, too."
—From the Foreword by James Miller, Lead Program Manager, Common Language Runtime, Microsoft Corporation

Essential .NET, Volume 1 , provides everything developers need to take full advantage of the power of Microsoft .NET. This book describes, in depth, the glue of the .NET Framework: the Common Language Runtime (CLR). Box and Sells explain the inner workings of the CLR—the rationale behind its design, the problems it solves, and the role of type in CLR programming—and show readers how to build better applications using the .NET Framework while gaining a more complete understanding of how the CLR works.

The book is packed with the practical detail and expert advice only Don Box can provide. Topics covered include:

- CLR's evolution
- Assemblies in the .NET Framework
- The CLR type system
- Programming with type
- Objects and values
- Methods
- Explicit method invocation
- Application domains
- Security
- Interoperability

Essential .NET, Volume 1 , is an authoritative guide to the Microsoft .NET Common Language Runtime.

Books in the Microsoft .NET Development Series are written and reviewed by the principal authorities and pioneering developers of the Microsoft .NET technologies, including the Microsoft .NET development team and DevelopMentor. Books in the Microsoft .NET Development Series focus on the design, architecture, and implementation of the Microsoft .NET initiative to empower developers and students everywhere with the knowledge they need to thrive in the Microsoft .NET revolution.

0201734117B10042002

About the Author

Don Box is a leading educator, recognized authority on the Component Object Model (COM), coauthor of the Simple Object Access Protocol (SOAP) specification, and coiner of the term "COM is Love." He recently joined Microsoft as an architect in the Microsoft® .NET Developer and Platform Evangelism Group.

Earlier in his career, Box cofounded DevelopMentor Inc., a component software think tank aimed at educating developers on the use of the COM, Java, and XML. A popular public speaker, Box is known for engaging audiences around the world, combining deep technical insight with often outrageous stunts.

0201734117AB06132002

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 XMLWeb 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 XMLWeb 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 XMLWeb 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 XMLWeb 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 ST3on XMLWeb 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.
- Sum 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.

0201734117P09272002

Most helpful customer reviews

11 of 12 people found the following review helpful.
Hardcore .NET
By Bradley J. Wilson
Don't let the first few chapters of this book fool you: this is a book for hardcore .NET developers. It shouldn't be the first book you buy about .NET, as it goes into incredible detail about the fundamentals of the .NET platform. For example, when you learn about using types on the platform, it's not just a pragmatic approach to writing code: it shows underneath how the system does what it does. This gives you a fuller view of the system, and lets some of the mystery disappear. The knowledge makes you a better "big picture" developer.
Don thinks at a high level, and writes very concisely as a result. By any other author, this book might've been a 1400 page mammoth; I'm amazed at the valuable data he's packed into just over 400 pages.
Some developers may the material in this book unattainable because of the concise and in-depth technical material. Those who do grok it will find it invaluable. This book was well worth the wait for me.

7 of 8 people found the following review helpful.
A dry subject made interesting
By William G. Ryan
There are like a zillion CLR books out there and overall, it's not the type of subject that normally keeps you glued to it. When I got Jeffrey Richther's Microsoft .NET Framework book, I was convinced no one was going to outdo him. Well, it's a close call, but I think they are both Superb books by excellent authors. I've purchased Don's stuff before and really liked it. This book lived up to its expectations.
I think his ability to communicate some of the more obscure areas of the CLR in a very clear matter is what makes this book shine. This book can be understood by anyone because of the writer's gift for writing...but that's not to say it's a novice's book. Wherever you are in the .NET learning curve, there's something for you

in this book.
If you really want to learn the CLR, this is a great place to start.

7 of 8 people found the following review helpful.
Uncovers the little quirks and secrets of the CLR
By Southern California .NET User Group
This book is worth reading if you keep in mind that its main purpose is to uncover the little quirks and secrets of the CLR. As the author states, it isn't intended to be a tutorial and shouldn't be your first choice if you are new to .NET programming (I'd recommend the excellent Applied .NET Framework Programming by Jeffrey Richter as a good starter book). However, reading Essential .NET could potentially save you lots of time sifting through the MSDN documentation to find out why your program is not behaving exactly the way you think it should (you know, those little, tiny, nasty bugs that prove to be the hardest to find).
As with any book that tries to cover such an extensive ground as the .NET CLR is, there are tradeoffs in the depth and extent with which the author describes each subject. In this case, Box chose to highlight the details of the inner workings of the CLR that we, as programmers, must have present to make efficient and appropriate use of the runtime facilities. Chapters one through five deal with basic concepts that, in my opinion, are best left to an introductory book and are not worth more than skimming through them, although you could always find a golden needle hidden in the haystack. However, on chapters six and after, the book really takes off and you'll probably find new things to learn page after page.
Although the crucial details are clearly exposed, this book is by no means exhaustive, I believe it can be considered more as a base from where you can start researching further about the subject of your interest. For example chapter seven, "Advanced Methods", deals with stack/message transitions, proxies, sinks and contexts. All these concepts are very well covered but I didn't get the eureka! feeling until I read Ingo Rammer's Advanced .NET Remoting and could see those concepts in action and realize their importance.
All in all, a book that deserves a slot in your .NET library (a slot somewhere in between a pair of good tutorials and the in-detail books about the areas of the framework that draw your interest). I would consider it a good investment of your time and money and I also see myself coming back to it (specially back to chapter 6-10) as a refresher. -- Review by Julio G.

See all 30 customer reviews...

ESSENTIAL .NET, VOLUME I: THE COMMON LANGUAGE RUNTIME BY DON BOX, CHRIS SELLS PDF

Also we discuss the books **Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells**; you may not find the published books right here. Numerous compilations are supplied in soft file. It will specifically give you much more benefits. Why? The very first is that you may not have to carry guide anywhere by fulfilling the bag with this Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells It is for guide remains in soft data, so you could wait in device. Then, you could open the gizmo everywhere and also check out guide appropriately. Those are some couple of benefits that can be got. So, take all benefits of getting this soft data publication Essential .NET, Volume I: The Common Language Runtime By Don Box, Chris Sells in this internet site by downloading in web link provided.

From the Back Cover

"Don taught me stuff I didn't know about my own product! And I bet he'll teach you something, too."
—From the Foreword by James Miller, Lead Program Manager, Common Language Runtime, Microsoft Corporation

Essential .NET, Volume 1 , provides everything developers need to take full advantage of the power of Microsoft .NET. This book describes, in depth, the glue of the .NET Framework: the Common Language Runtime (CLR). Box and Sells explain the inner workings of the CLR—the rationale behind its design, the problems it solves, and the role of type in CLR programming—and show readers how to build better applications using the .NET Framework while gaining a more complete understanding of how the CLR works.

The book is packed with the practical detail and expert advice only Don Box can provide. Topics covered include:

- CLR's evolution
- Assemblies in the .NET Framework
- The CLR type system
- Programming with type
- Objects and values
- Methods
- Explicit method invocation
- Application domains
- Security
- Interoperability

Essential .NET, Volume 1 , is an authoritative guide to the Microsoft .NET Common Language Runtime.

Books in the Microsoft .NET Development Series are written and reviewed by the principal authorities and pioneering developers of the Microsoft .NET technologies, including the Microsoft .NET development team and DevelopMentor. Books in the Microsoft .NET Development Series focus on the design, architecture, and implementation of the Microsoft .NET initiative to empower developers and students everywhere with the knowledge they need to thrive in the Microsoft .NET revolution.

0201734117B10042002

About the Author

Don Box is a leading educator, recognized authority on the Component Object Model (COM), coauthor of the Simple Object Access Protocol (SOAP) specification, and coiner of the term "COM is Love." He recently joined Microsoft as an architect in the Microsoft® .NET Developer and Platform Evangelism Group.

Earlier in his career, Box cofounded DevelopMentor Inc., a component software think tank aimed at educating developers on the use of the COM, Java, and XML. A popular public speaker, Box is known for engaging audiences around the world, combining deep technical insight with often outrageous stunts.

0201734117AB06132002

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 XMLWeb 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 XMLWeb 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 XMLWeb 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 XMLWeb 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 ST3on XMLWeb 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.
- Sum 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.

0201734117P09272002