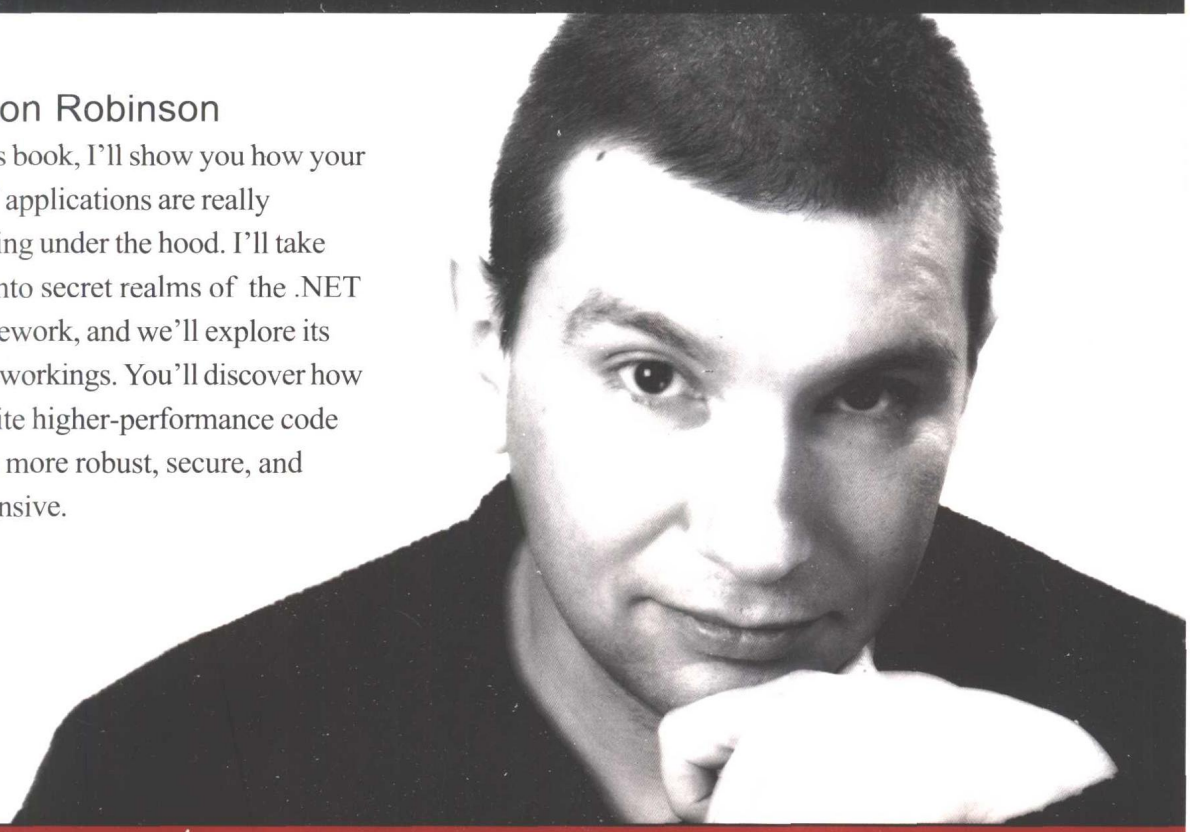
PROGRAMMER TO PROGRAMMER™



Simon Robinson

In this book, I'll show you how your .NET applications are really working under the hood. I'll take you into secret realms of the .NET Framework, and we'll explore its inner workings. You'll discover how to write higher-performance code that's more robust, secure, and responsive.

Advanced .NET Programming

高级.NET 程序设计

Simon Robinson 著
冉晓旻 王军 译

清华大学出版社

高级.NET程序设计

Simon Robinson 著

冉晓旻 王军 译

清华大学出版社

北京

内 容 简 介

本书详细、专业地讲述了.NET 应用程序的工作原理，深入探讨了.NET 的一些高级技术，其中包括中间语言、CLR 工作原理、应用程序操作性能的优化和系统资源使用情况的剖析、线程同步技术、高级 Windows Forms 技术、如何使用 WMI 管理计算机中的资源、如何动态生成代码以及.NET 中的代码访问安全性和密码术等内容。

本书适用于有一定编程基础并对 C#有所了解的.NET 开发人员。此外，读者还必须熟悉.NET 的基本概念和主要的类库。

EISBN: 1-86100-629-2
Advanced .NET Programming
Simon Robinson

图书在版编目(CIP)数据

高级.NET 程序设计/(美)罗宾逊著；冉晓旻，王军译.—北京：清华大学出版社，2003
书名原文：Advanced .NET Programming
ISBN 7-302-06789-9
Ⅰ. 高… Ⅱ. ①罗…②冉…③王… Ⅲ. C 语言—程序设计 Ⅳ. TP312
中国版本图书馆 CIP 数据核字(2003)第 047341 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：曹康
文稿编辑：李阳
封面设计：康博
版式设计：康博
印 刷 者：北京市清华园胶印厂
发 行 者：新华书店总店北京发行所
开　　本：787×1092　1/16　**印张**：30.5　**字数**：780 千字
版　　次：2003 年 7 月第 1 版　2003 年 7 月第 1 次印刷
书　　号：ISBN 7-302-06789-9/TP・5050
印　　数：1～4000
定　　价：62.00 元

出版者的话

近年来，国内计算机类图书出版业得到了空前的发展，面向初级用户的应用类软件图书铺天盖地，但是真正有深度和内涵的高端图书不多。已经掌握计算机和网络基础知识的人们，尤其是 IT 专业人士迫切需要“阳春白雪”。IT 图书市场呼唤精品！

为了满足这种市场需求，清华大学出版社从世界出版业知名品牌 Wrox 出版公司引进了受到无数 IT 专业人士青睐，被奉为 IT 出版界经典之作的 Professional 系列丛书。这套讲述最新编程技术与开发环境的高级编程丛书，从头到尾都贯穿了 Wrox 出版公司“由程序员为程序员而著(Programmer to Programmer)”的出版理念，每一本书无不是出自软件大师之手。实际上，Wrox 公司的图书作者都是世界顶级 IT 公司(如 Microsoft，IBM，Oracle 以及 HP 等)的资深程序员，他们的作品既深入研究编程机理，传授最新编程技术，又站在程序员的角度，指导程序员拓展编程思路，学习实用开发技巧，从而风靡世界各地，被 IT 专业人士和程序员视为职业生涯中的必读之作。

为了保证该系列丛书的质量，清华大学出版社迅速组织了一批位于 IT 开发领域前沿的专家学者进行翻译，经过编辑人员的进一步加工整理后，现陆续奉献给广大读者。

读者可以从 www.wrox.com 网站下载所需的源代码并获得相关的技术支持。同时，也欢迎广大读者参与 p2p.wrox.com 网站上的在线讨论，与世界各地的编程人员交流读书感受和编程体验。

前　言

本书教您如何最有效地利用.NET 的特性。它基于这样的原则——要想充分利用.NET 特性来编写优秀、高性能、健壮的应用程序，最佳途径就是去了解该项技术的核心内幕。本书的部分章节会带您一起去探索.NET 的内部世界，特别是其公共中间语言(Common Intermediate Language，CIL)部分，还有部分章节以实践为基础，教您如何使用具体的技术，如线程、动态代码生成和 WMI 等。

本书不是一本纯理论书籍，同时它也不是针对具体问题的书籍(只讲述如何编写代码来完成任务，但不涉及其原因以及工作原理)。相反，我们在本书中尝试了一种新的方式，把实际中特定于某些技术的应用和 CLR 的基本原理结合起来，相信这正是高级.NET 开发人员所需要的。目前市场上有大量关于.NET 技术内幕的书。本书的技术容量超越了 MSDN 文档，也超越了其他大部分的.NET 书籍。但是，我们决不会抽象地去研究某些特性。其根本原因是，理解 CLR 的实现细节在某种程度上有助于您编写出更出色的代码。本书安排了许多章节，展示如何更好地编写特定领域的应用程序(如 Windows Forms)、如何利用.NET 特性(如安全性)、如何更好地调试代码或优化代码，以获得更优的性能。

本书是关于.NET 编程的一本高级教程。它适用于那些熟悉.NET 应用的开发原理，并且希望深入学习的读者。读者可以通过本书了解 JIT 编译器在评估代码是否具备类型安全特性时，所需了解的信息。或许通过研究编译器所生成的 CIL 代码，您能够对优化有所了解，从而编写出性能更好的代码。您也可能希望在代码中使用一些高级.NET 功能(如动态代码生成)，或者希望了解关于建立安全性和部署代码的信息，那么可将本书作为参考书籍。

本书一开始就显示了它的“高级”特性，第 1 章和第 2 章介绍了中间语言(IL)。如果您确实希望从本书中受益，那就必须掌握知识点——通过掌握 IL，就可以了解到某些高级语言(如 VB、C#和 C++)的基本原理。

正是由于本书是一本高级教程，所以除了偶尔地概述相关背景知识外，不会对.NET 的基础知识进行讲解。如果您还不了解 JIT 编译器的概念，或者不清楚值类型和引用类型的区别，那么本书并不适合您——因为我假设您已经学过所有的基础知识。同样，我假设您已经熟练掌握了至少一种.NET 语言，如 Managed C++、Visual Basic .NET 或 C#。为了阅读本书，您还应该习惯基于实现-继承方式的面向对象编程。

例如，我们在本书中涉及了程序集的相关内容。但本书并不没有讲述 Microsoft 为什么引入程序集的概念、什么是元数据，程序集如何解决版本化问题，以及如何将使用程序集所需的信息集中到同一个地方的问题。您可以参考市场上大量的相关书籍。本书关于程序集的章节首先给出程序集的二进制格式——回顾 IL 代码和元数据的二进制格式，以及如何提高性能。接下来介绍如何通过编程的方法从程序集中提取信息和元数据，最后告诉您如何使用程序集和程序集中的资源，确保对应用程序进行正确的本地化，以适应全球化的销售策略。

本书主要内容

下面依次讲解了本书各章的主要内容。

中间语言(第 1、2 章)

本书前两章介绍了中间语言。在这两章中不可能涵盖中间语言的所有知识，只研究了基本概念，包括编程原理，如使用计算栈、声明和实例化值类型和引用类型，以及与非托管代码的交互操作。这两章的重点在于通过中间语言来加强对 CLR 的理解，并促使我们改进所编写的 C++/C#/VB 代码。在第 2 章的结尾部分，把由 C++，C#和 VB 编译器生成的 IL 代码进行了比较。您可以从 Wrox Web 站点下载一个关于 IL 指令集的综合指南，该指南可以作为对这两章未涉及的 IL 指令内容的补充。

CLR 内幕(第 3 章)

第 3 章讲述 CLR 工作原理的很多问题，这些问题是初级.NET 书籍通常没有涉及的内容。以下是其中特别关注的几个主题：

- JIT 编译器——研究 JIT 编译器处理代码的方式。
- ECMA 标准及其与 Microsoft .NET 实现方式的关系。
- 类型安全——类型安全就像是一个黑箱，无论代码安全与否，多数人几乎无法了解代码通过验证或无法通过验证的原因。在这一节中，我们将研究用来验证代码的算法，解释测试代码的方式。
- 托管和非托管代码 ——讨论托管和非托管代码如何协同工作，以及托管与非托管的真正差异在哪里。这一节将帮助 C++开发人员理解 C++编译器对于托管和非托管代码的一些限制，以及如何充分发挥跨边界应用程序的功效。

程序集和本地化功能(第 4 章)

第 4 章详细介绍程序集，包括以下内容：

- 程序集的基本结构，以及如何将元数据和 IL 嵌入程序集。这里的重点是介绍如何根据程序集的设计宗旨来改善性能。我们还将分析公有和私有程序集。
- 程序集和模块之间的关系。
- 程序集的程序化操作。
- 如何在程序集中嵌入资源，如何使用这些资源本地化应用程序。也就是说本书还将概述.NET 对全球化特性的支持。

无用单元收集(第 5 章)

毫无疑问，您一定对.NET 无用单元收集的基本原理非常熟悉。第 5 章会带您一起探讨无用单元收集器的一些细节。您会了解如何以性能为原则设计无用单元收集算法，无用单元收集器如何与代码中的线程交互作用。我们还会概述一些与无用单元收集相关的高级主题，如弱引用。

改进性能(第 6 章)

性能是设计代码时需要考虑的重要因素，其他章节多次谈到.NET 体系结构的性能。第 6

章是关于性能的专题，包括.NET Framework 中性能的一些设计方式，以及如何利用这些要素编写更高性能的代码。我们特别讲述了以下几个方面的内容：

- 在编写托管代码时(相对于非托管代码)有关性能的注意事项。一般地讲，托管代码的性能非常好，而且在将来会变得更好。但是，它仍然存在 Microsoft .NET 没有公开的一些问题，我们将对这些问题进行探讨。
- JIT 优化。讨论在 JIT 编译时进行优化所带来的好处，以及如何在代码中控制这类优化。
- 对编写代码时有关提高性能的建议。

剖析(第 7 章)

代码性能优化与代码性能测定同时进行。在第 7 章，我们将介绍如何剖析托管代码。主要内容包括：

- .NET 相关的性能计数器和 PerfMon 工具。
- 编写和使用自定义性能计数器。
- 任务管理器和其他剖析工具的高级用法。

动态代码生成(第 8 章)

动态代码生成不只是为编译器开发人员准备的。在某些情况下，此技术可以用于改进性能。例如，它可以用于 System.Text.RegularExpressions 类，以提供高性能的正则表达式分析，这类操作相当简单。在第 8 章我们将涉及代码生成的基本原理，包括：

- 命名空间 System.CodeDom 和 System.CodeDom.Compiler，如何使用它们来控制源代码的生成和编译。还会介绍特定的语言编译器可以利用的工具。
- 命名空间 System.Reflection.Emit 和程序集的动态生成。

线程操作(第 9 章)

在某些方面，通过托管代码来实现线程变得越来越简单，因为现有的类就可以实现诸如线程池等特性，这些特性很重要，但非托管代码实现起来则非常困难。但是，在另一方面，线程操作却变得更加复杂，因为面临更多的线程模型选择——尤其当我们可以轻易地实现线程池或方法的异步执行时。在这一章中，将讨论可用于编写多线程托管代码，以及控制线程间的通信与同步的方法。

管理工具(第 10 章)

.NET 提供了功能强大的工具，用于与运行代码的操作系统和硬件进行交互作用。这些工具有两个来源：各种架构基类，以及允许将.NET 应用程序连接到 WMI 提供程序上的 System.Management 命名空间和 System.Management.Instrumentation 命名空间。使用各种架构基类来访问环境信息相对简单，所以我们将简要介绍一下。但是，WMI 是一个强大的工具，然而却只有少数开发人员能很好地理解并加以应用。所以，第 10 章的大部分篇幅将用来讨论该话题。主要内容包括：

- WMI 的基础概念——如何建立 WMI 提供程序和消费者，使用 WMI 所能完成的任务——包括：从查找系统的数据库连接到通过编程的方法弹出一个 CD 等操作。
- 相关.NET 基类如何与 WMI 交互作用，如何使用 WMI 查找、甚至控制硬件操作的托管

应用程序。

高级 Windows Forms(第 11 章)

我们都曾编写过基本的 Windows Forms 应用程序——在一个窗体中放置若干控件，甚至使用 GDI+来执行自定义绘图。本章会带您深入探讨这个主题：

- 支撑 Windows Forms 和 Windows Forms 事件的 Windows Message 体系结构。通过对此架构的理解，您可以访问一些普通 Windows Forms 事件机制无法直接访问的有用事件。
- 能够吸引用户的更高级的窗口技术，如非矩形窗体和自绘制控件。
- 使用 GDI+绘图的性能问题。

安全性(第 12 章)

.NET 有一个重要承诺——增强的安全特性将使您更好地控制代码所能执行的操作，使您更加放心地下载和运行代码。本章教您如何利用基于证据的.NET 安全机制，包括以下主题：

- 基于证据的安全机制如何运行。
- .NET 安全机制与 W2K/XP 安全机制的交互。
- 使用诸如 mscorcfg 和 caspol 的工具控制安全设置。

密码术(第 13 章)

密码系统用于安全通信，尤其是通过 Internet 进行的通信：确保未授权用户无法查阅或篡改数据。Microsoft 提供了一组功能强大的类，允许您使用密码服务，如散列、消息验证和公钥-私钥的生成。本章介绍了这些概念的工作原理，以及如何通过 System.Security.Cryptography 和相关命名空间中的类来使用密码工具。

编程语言

就所采用的编程语言而言，本书主要采用语言中立的方式进行讲解。目标读者是具有一定经验的所有.NET 程序员，无论您会首选哪一种编程语言。但问题是，必须选用一些语言来表示实例。大部分的实例将采用 C#来描述，这意味着即使您偏爱另一种语言，也必须对 C#有大致的了解。虽然书中的实例大都采用 C#语言，但考虑到 VB 程序员的需要，您可以在网上下载 C#和 VB 的版本。在有些情况下，我们还会列举一些 VB 或 C++实例，以说明只针对该语言出现的特性。当然，如果要说明有关编译的问题，我们随时可能会使用 IL 语言。

特定于语言的术语

对于不同语言的程序员来说，还存在一个问题，在很多情况下，OOP 或.NET 结构在不同的语言中具有不同的术语。因为本书必须使用一些术语，所以因编程语言而引起的歧义在所难免，甚至有些开发人员会对书中的术语非常陌生。因为本书的大部分代码都采用 C#语言编写，所以也会相应地采用 C#的术语系统。

用户支持

我们一贯重视读者的意见，并想知道每位读者对本书的看法，包括读者喜欢和不喜欢的内容，以及读者希望我们下一次完善的地方。您可以通过发送电子邮件(地址为feedback@wrox.com)来向我们反馈意见。请确保在反馈信息中提到本书的书名。

如何下载本书的示例代码

当您访问 Wrox 公司站点(地址为 http://www.wrox.com/)时，通过 Search 工具或书名列表，可以方便地定位需要的书目。然后，单击 Code 列中的 Download 超链接，或者单击本书的详细信息页面中的 Download Code 超链接，就可以下载相应的示例代码。

从我们的站点上下载的可用文件都是使用 WinZip 压缩过的文档。把附件保存到本地磁盘上的文件夹中后，需要使用一个解压缩程序(例如 WinZip 或 PKUnzip)来解压缩文件。在解压缩文件时，通常将代码解压缩到每一章所在的文件夹中。在解压缩的过程中，应确保解压缩程序被设置为使用原有文件夹名。

勘误表

我们已经尽最大努力确保本书中的文本和代码没有错误，但是错误仍然在所难免。如果您发现本书存在错误，例如拼写错误或不正确的代码段，请反馈信息给我们，我们将不胜感激。勘误表的发送可以节约其他读者学习本书的时间，而且能够帮助我们提供更高质量的信息。您的反馈信息将被检查，如果正确，将被粘贴到本书的勘误页面上，或者在本书的后续版本中使用。

要在我们的站点上找到勘误表，请访问 http://www.wrox.com/，并通过 Search 工具或者书名列表轻松定位到本书页面，然后，单击 Book Errata 超链接即可，该链接位于本书的详细信息页面中的封面图解下面。

E-mail 支持

如果您希望直接向详细了解本书的专家咨询本书中问题，可以发送电子邮件到support@wrox.com，要求在邮件的主题栏中带上本书的书名和 ISBN(国际标准图书编号)的后 4 位数字。一封典型的电子邮件应包括下面的内容：

- 在主题栏中必须有本书的书名、ISBN 的后 4 位数字和问题所在的页码。
- 正文部分应包括读者的名字、联系信息和问题。

我们将不返回无用邮件，因为我们仅仅需要有用的详细资料，以便可节约您和我们的时间。当您发送一个电子邮件信息时，它将经过下面一系列支持：

- 客户支持：首先，您的信息将被递送到我们的客户支持人员手中，并由他们阅读。对于一些被频繁提到的问题将被归档，并将立即回答有关本书或者 Web 站点的任何常见问题。
- 编辑支持：接着，一些有深度的问题将被送到对本书负责的技术编辑手中，他们在程序设计语言或者特定的产品上有着丰富的经验，能够回答相关主题的详细技术问题。
- 作者支持：最后，如果编辑不能回答您的问题(这种情况很少发生)，他们将求助本书的

作者。我们将尽量保护作者免受干扰，以便不影响其写作。然而，我们也非常高兴转寄给他们一些特殊的问题。所有 Wrox 公司的作者都为他们的书提供技术支持。作为回应，他们将发送电子邮件给用户和编辑，进而使所有的读者受益。

Wrox 公司的支持过程仅仅对那些与我们出版的书目内容直接相关的问题提供支持，对于超出常规书目支持的问题，您可以从 http://p2p.wrox.com/论坛的公共列表中获得支持信息。

p2p.wrox.com 站点

为了便于作者和其他人讨论，特将编程人员加入到 P2P 站点的邮件列表中，而且我们独特的系统将 programmer to programmer(由程序员为程序员而著)的编程理念与邮件列表、论坛、新闻组以及所有其他服务内容(一对一的邮件支持系统除外)相联系。如果您向 P2P 发送一个问题，应该相信它一定会被登录邮件列表的 Wrox 公司作者和其他相关专家所检查到。无论您是在阅读本书，还是在开发自己的应用程序，都可以在 p2p.wrox.com 站点中找到许多对自己有所帮助的邮件列表。

按照下面的步骤可以预订一个邮件列表：

(1) 登录 http://p2p.wrox.com/站点。

(2) 从左边的菜单栏选择一个适当的类别。

(3) 单击您希望加入的邮件列表。

(4) 按照说明订阅并填写自己的邮件地址和密码。

(5) 回复您收到的确认邮件。

(6) 使用预订管理程序加入更多的邮件列表并设置自己的邮件首选项。

本系统提供最佳支持的原因

您可以加入整个邮件列表，也可以只接收每周的邮件摘要。如果您没有时间和工具来接收邮件列表，可以直接查找我们的在线文档。独特的 Lyris 系统可以将一些没有用的垃圾邮件删除，并保护您的电子邮件地址不被侵扰。当存在加入和离开列表，以及任何有关列表的其他常见问题时，请发送邮件到 listsupport@p2p.wrox.com。

目　录

第1章　中间语言导论

中间语言(也称为公共中间语言，CIL；或 Microsoft 中间语言——MSIL)是.NET 的核心。无论您的源代码以何种语言编写，只要运行在.NET Framework 所支持的环境中，肯定要被编译为 IL 代码。因此，如果您希望站在一个较高的层次理解.NET 的工作原理，了解一些 IL 的相关知识会大有益处。在了解了 IL 之后即可：

- 更好地理解托管代码的内部运行机制。您将拥有两个信息渠道：阅读文档或者研究编译器生成的 IL 代码，通过后一种方式可以获得文档中没有涉及到的信息。
- 查看常用语言(如 C++，C#或 VB)编译器所编译过的代码，在有些时候可以帮助您调试或设计代码以进行性能优化。还可以帮助您理解所选定语言的优势。
- 直接采用 IL 语言编写一些代码——也许您并不希望经常这样做，但有时它确实有助于利用普通语言无法实现的 IL 特性，正如在.NET 发布的初期，编写高性能 C/C++代码的程序员偶尔也使用本机程序集语言，以提高某些特殊函数的性能。
- 显而易见，如果您正在编写开发工具，如编译器或调试器，理解 IL 则是一个先决条件。

由于 IL 的重要性，我们决定在本书第 1、2 章首先介绍这种语言。本章主要介绍 IL 的基础知识。您将了解基本的 IL 程序集语法，IL 所依赖的抽象堆栈机器如何工作，以及如何通过 IL 进行程序流程控制。最后，我们会介绍 IL 错误以及如何调试 IL 源代码。第 2 章会告诉大家如何利用上述知识对类和结构进行编码并调用实例方法，还会涉及一些高级主题，如委托、异常和非托管代码的调用。

我们不打算在本书中涵盖 IL 语言的每一个细节，也不会涉及一些更高级且很少使用的特性。本书的主题毕竟是高级.NET 编程，而不是 IL 编程。我不主张使用 IL 编写所有的代码，在多数情况下，这是一种愚蠢的做法，开发和调试时间也会因此而显著增加。确切地说，介绍 IL 语言的目的在于，理解 IL 有助于最大程度地利用.NET Framework 进行高级语言开发。所以，介绍 IL 语法是必需的，其重点在于教您掌握足够的基本概念以阅读 IL 代码，并帮助您理解高级语言(如 C#、VB、托管 C++或一些第三方语言)如何隐藏(或提供假象).NET Framework 执行任务的方式。为简单起见，高级语言往往会这样做，但 JIT 编译器必须处理的 IL 代码则不能隐藏任何细节。

您还会发现，本章的开始部分讲解 IL 概念的速度相当缓慢。本书假设您没有 IL(或本机代码)编程的经验。而是假设您已熟练掌握了一门高级语言，并理解面向对象编程和.NET Framework 的基本概念。通过这两章的学习，您应该已掌握了足够的知识来阅读编译器所生成的 IL 代码。此外，从 Wrox 站点下载的附录可以帮您全面了解每一条 IL 指令，以及相应的操作码，所以，如果遇到书中没有涉及到的指令，可以参考该附录。

同时还应该提醒您，这些 IL 章节并不是一定要阅读的，因为后续章节的大部分代码是用 C#语言描述的，所以，只要能够阅读 C#代码，就能完成剩余部分的学习。如果您确实对阅读