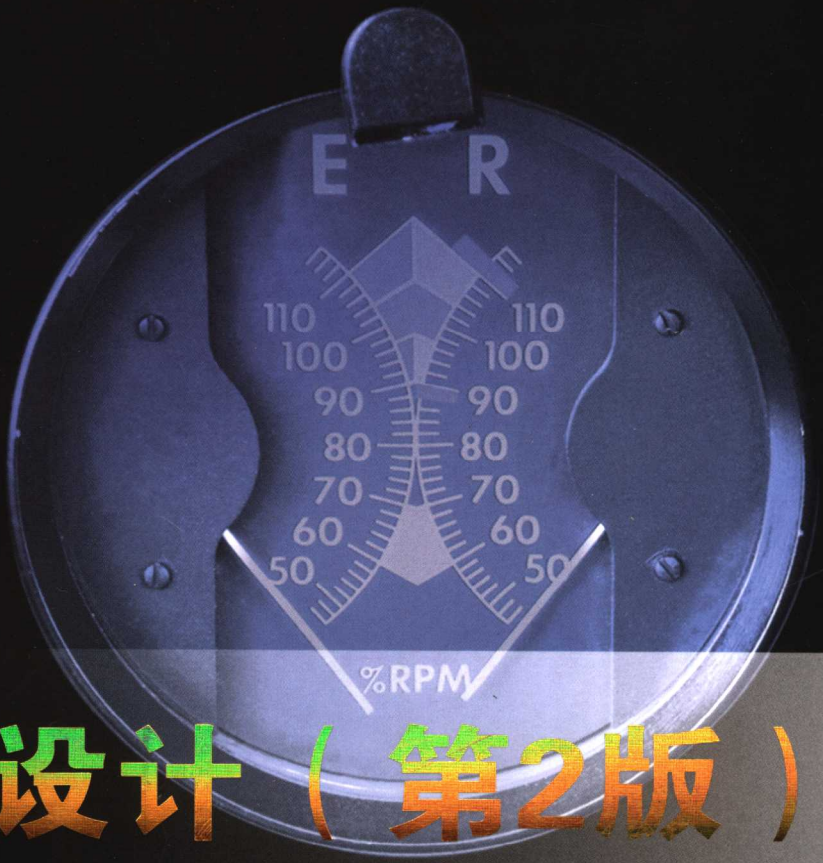
微软技术丛书

Microsoft

- Jeffrey Richter畅销新作
- 基于.NET Framework 2.0全新演绎
- 深入剖析 CLR 内部机理



框架设计（第2版）：CLR Via C#

（美）Jeffrey Richter 著

周 靖 张杰良 译

清华大学出版社

微软技术丛书

框架设计(第 2 版):CLR Via C#

(美) Jeffrey Richter 著
周 靖 张杰良 译

清华大学出版社
北 京

内 容 简 介

作为深受编程人员爱戴和尊敬的编程专家，微软.NET 开发团队的顾问，本书作者 Jeffrey Richter 针对开发各种应用程序(如 Web Form、Windows Form 和 Web 服务、Microsoft SQL Server 解决方案、控制台应用程序、NT Service)的开发人员，深入揭示了公共语言运行库(CLR)和.NET Framework.，演示了如何将这些知识应用到实际开发。全书分 5 部分，共 24 章。第Ⅰ部分讲述 CLR 基础，第Ⅱ部分介绍类型的使用，第Ⅲ部分说明类型的设计，第Ⅳ部分介绍基本类型，第Ⅴ部分讲述 CLR 实用特性。

通过本书的阅读，读者可以掌握 CLR 精髓，轻松、高效地创建高性能应用程序。

北京市版权局著作权合同登记号　图字：01-2006-2342

图书在版编目(CIP)数据

框架设计(第 2 版)：CLR Via C#/(美)瑞奇特(Richter, J.)著；周靖，张杰良译. —北京：清华大学出版社，2006.11
(微软技术丛书)
书名原文：CLR Via C#, Second Edition
ISBN 7-302-14016-2

Ⅰ. 框… Ⅱ. ①瑞… ②周… ③张… Ⅲ. C 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2006)第 121802 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

文稿编辑：文开棋
封面设计：陈刘源
排版人员：房书萍
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：35.75　**字数**：845 千字
版　　次：2006 年 11 月第 1 版　2006 年 11 月第1 次印刷
书　　号：ISBN 7-302-14016-2/TP·8420
印　　数：1～5000
定　　价：68.00 元

来自微软的声音

这本书具有良好写作风格、包含丰富信息，是 Jeffrey 在.NET Framework 上所花心血的明证。

——Eric Rudder(高级副总裁，开发人员兼平台传播，微软公司)

Jeffrey 和构建 CLR(公共语言运行库)的人员朝夕相处，写就这部揭示 CLR 内幕的超级畅销书。

——Dennis Angeline(项目总经理，公共语言运行库团队，微软公司)

Jeffrey 凭借着他多年的 Windows 编程经验和深刻的洞察力，解释了.NET Framework 的实际工作机制，阐述了为什么要按照我们的方式来构建它，如何更进一步了解它。

——Brad Abrams(项目总经理，.NET Framework 团队，微软公司)

Jeffrey Richter 素来擅长以深入浅出、言简意赅、准确无误的方式来解释复杂的内容。这种风格延续到 C#语言、.NET Framework 和.NET 公共语言运行库。打算理解这些重要新技术幕后原理的读者，绝对不要错过本书！

——Jim Miller(项目总经理，公共语言运行库内核团队，微软公司)

一看就知道，这是一本介绍公共语言运行库的好书。介绍 CLR 垃圾收集的一章(第 1 版中的第 19 章，第 2 版中的第 20 章)讲得太棒了。Jeffrey 不止是从理论上描述了垃圾收集机制的工作原理，还讨论了每个.NET 开发人员都必须知道的终结机制。

——Mahesh Prakriya(项目总经理，公共语言运行库团队，微软公司)

本书准确、深入、清楚地介绍了公共语言运行库。很少有书采用这样的写作风格：设问，紧接着提出答案。本书的写作引人入胜。

——Jim Hogg(项目经理，公共语言运行库团队，微软公司)

正如《Windows 编程》是 Win32 编程人员案头必备一样，本书绝对会是.NET Framework 编程人员不可或缺的重要参考书。本书的独特之处在于以自下而上的方式来理解.NET Framework 程序设计，通过让读者扎实地理解 CLR 底层概念，Jeffrey 帮助读者奠定牢固的基础，使他们能够迅速而轻松地写出可靠、安全、高性能的托管代码。

——Steven Pratschner(项目经理，公共语言运行库团队，微软公司)

Jeffrey Richter，大人物！

——匿名(项目经理，公共语言运行库团队，微软公司)

读者的心声

读者：Henny

赶紧买吧！ 本书引导我“游历”CLR，解释了它的工作方式和工作原理。作者通过深入CLR所用内部数据结构，全面展示了元数据。通过本书，可了解到哪些数据是以什么方式放到堆栈上的，它们又是以什么顺序放到堆栈上去的。CLR所涉及的内容太宽泛了，很难一下子全部消化。本书的特点是结构清晰、通俗易懂。

乍一看，本书似乎是一本.NET Framework入门书。再一看，似乎又是一本从基本类型讲到泛型的C#程序设计教程。其实不然，本书从解释何为.NET Framework和CLR执行模型开始，然后过渡到元数据、应用程序的配置和安装，进而深入内部机制。通过本书，您可以更好地掌握各种工具，比如NGen.exe，知道使用这些工具的最佳时机以及为什么要用或不用这些工具。书中没有太多晦涩难懂的内容，通俗易懂是本书的最大特点。对于本书，想说得实在太多，但在此时，我只想说一句，赶紧买吧！

读者：Al Tenhundfeld

涵盖CLR基本要素 我支持其他读者对本书的正面评论。如果已经非常精通.Net平台，打算进一步理解CLR的内部机理，本书将是理想的起点。这一版与其基于1.0/1.1的上一版一样，堪称佳作。

读者：Paulo Sirimarco

适合高级开发人员阅读 本书深入介绍了CLR和Framework。我们可以从本书中找到.性能方面的提示和对Framework大部分功能的完整描述。

读者：Dematte' Lorenzo

Ritcher著作总是不可或缺 自从《Windows编程》出版以来，Ritcher的书就是编程人员人手一册的“圣经”。本书结构清晰，行文流畅，包含实用示例。

读者：Pradeep C

Richter再现英雄本色 坦白地说，Jeffrey Richter真是我的英雄。他总是给我惊喜。无法想像他是怎么做到这一点的，居然能以非常清晰的方式阐述晦涩难懂的主题。而且，他总能一次又一次地做到这一点。他的每一本著作便是明证。

他的书有一个典型的特点，即“人无我有，人有我精”。他虽然以与Microsoft紧密合作并为.NET团队提供咨询闻名，但我更愿意将他看作一名优秀作者和良师。

说到本书，应该不是一本C#入门书。我建议读者先找C#入门书(比如Jesse Liberty的书)来看(如果需要的话)，然后再来看这本书。通过本书，读者会比没有读过本书的人拥有更多优势，会为掌握.NET平台打下扎实的基础。“线程”这一章就使本书物有所值了。如同Richter的其他书一样，阅读本书绝对会使我们身心愉快。

没有人能拥有 Richter 那样的化腐朽为神奇的大手笔，没有人。

读者：Ashutosh Singh

改变人生的好书 如果您是.NET/C#初、中级编程人员，那么本书将改变您的人生。我买过若干本涉及不同领域的高级技术专著，但没有一本能与本书媲美。本书的写作风格引人入胜，能让您迅速理解并逐步喜欢书上所有这些难懂的概念。不过要做好准备，Richter 先生对.NET 的狂热会感染您，让您对.NET 欲罢不能，最终心甘情愿地花更多人力和财力在.NET 图书上。

读者：L. W. Danz

绝对好书，.NET 高级开发人员必读 本书让我如愿以偿。对于 Richter，他的写作风格非常出色，他的语言通俗易懂，他给出的例子更是妙不可言。他不只是阐述一个知识点，还展示了一种我们应该效仿的编码风格。大多数作者几乎都选择简单的例子，如果只是为了和正文呼应，这样做难免有些浪费时间。本书所提供的例子则是在正文基础上进行了延伸和拓展，推动您反复思量。

没错，不读这本书也能编写.NET 程序。但我坚信，如果花点时间来读这本书，写出来的程序肯定会更好。不过，我不建议您采用本书作为.NET 编程入门书，因为本书已经假设读者具有一定.NET 编程基础。

读者：DelBono Emanuele

仔细剖析幕后机理 本书解释了.NET CLR 的幕后细节。如果您想了解 CLR 内部的工作方式和工作原理，选择本书准不会错。

读者：Paul Sarkisian

CLR 图书中的佼佼者 读过 Wrox 出版的 *Visual C# Professional 2005* 和 O'Relly 出版的 *"C# Programming"* 之后，我最终选择了本书。它是我一直梦寐以求的好书。阅读本书，让人感觉到犹如良师在侧，它能够把问题讲解得非常透彻，容易理解。好书！

读者：David Douglass

如果把 Microsoft .NET 比作一盘棋局，那么本书将是起点方格 Microsoft .NET 的核心便是 CLR。.NET 开发主要是围绕着 CLR 来进行的。但如果对 CLR 一无所知，应该怎么办呢？

很多.NET 编程类图书都是围绕着语言来进行的。CLR 的功能基本上基于对语言的描述。Jeffrey Richter 的书是以 CLR 为中心。它描述了 CLR 的用途，描述了 CLR 的工作原理。C#用于提供实际的例子，演示如何使用 CLR。

本书清楚有效地呈现了丰富有用的信息，值得我们多花些时间访问 MSDN 或编写测试代码来进行验证。本书特色主题包括：

- 源代码是如何转换为中间语言(IL)的，它们是如何存储、管理和执行的
- 描述了运行时可用的代码元数据，以及元数据的用法
- 数据是如何分类、组织和管理的
- 描述了组成一个类的成员(字段和方法等)

- 如何处理异常
- 垃圾收集的工作原理
- 映射的工作原理
- 如何编写多线程应用程序

陷阱和错误提示贯穿全书。书中介绍了不同方法在许多情形下的执行性能。如果您尚未真正理解 CLR，那么我强烈建议您赶快读一读这本书。

读者：Patrick Smacchia

案头必备 如果您想进一步理解 CLR，或者希望自己的代码更出色，本书便是您的案头必备。几乎每一页都有无法从其他书上找到的有用信息。

我很喜欢看书中涉及的这方面的内容：微软的工程师们为什么会如此设计 CLR 和 Framework。例如，像下面这些棘手的问题，我们可以从书中找到答案：

- 在调用非虚实例方法时，C#编译器为什么会用 callvirt IL 指令(而不是 call IL 指令)?
- 在考虑使用显式接口方法实现(EIMI)时，还有哪些不太常见的情况？
- 底层的处理器体系结构和易失性存储器方法是如何与 CLR 关联的？
- 对于注册有很多事件的.NET Framework 类，比如 System.Windows.Forms.Control，设计它们的目的是为了在运行时节省内存吗？

在本书中，还可以找到很多类似问题的答案。

我还看重这个事实：J. Richter 是极少数精通这一主题并有资格评判微软某些设计选项的人之一。他的一些提议往往会被未来的.NET 发行包含在内。

显然，如果您是初学者，请不要从本书开始学习.NET。但如果您的目标是成为一名.NET 专家，最终需要阅读本书。

读者：Shawn Wildermuth

有价值，案头必备 我已经把这本书读了好几遍，相当喜欢其中的内容。第 23 章和第 24 章(线程和异步编程)足以使本书物有所值(没有人能像 Jeffrey Richter 那样解释线程问题)

我读过不少技术类图书，能让我体会到读书乐趣的寥寥无几。Jeffrey Richter 的新书便是其中之一。即使全书超过 500 页，包含大量重要的信息，但读起来仍然觉得很轻松。

读者：Jamabazi

启发性强，有醍醐灌顶的感觉 类似于前一本，但更详细，更容易阅读和理解。阅读本书是一种享受，同时也可以从中感受到他本人对 CLR 和 C#的一些看法。

读者：Larry Robinson

通俗易懂，可读性强 我读过相当多技术类图书，通常一年要读 30 多本。事实上，我发现，没有一本书有这本书写得好，它应该是所有技术类图书作者学习写作的典范。

书中深入讨论了最佳实践和内部工作机制。通过本书的阅读，您能进一步了解.NET，而且能更好地驾驭.NET。

读者： *William G. Ryan*

好得不能再好 我得承认，我自己是 Jeffrey Richter 的忠实读者。从 20 世纪 90 年代中期以来，我一直在读他的书，而且情不自禁地爱看他写的所有东西。如同他的其他书一样，本书愈发精益求精。上一版是帮助我学习.NET Framework 的关键，并为我节省了不少时间。这本书肯定也不例外。

本书虽然与其上一版类似，但更深入地讨论了新特性，比如泛型，同时还融入了他以前所写的东西。本书属于旧瓶装新酒，非常透彻地阐述了 CLR 的内部机理。

我最近也在为一家大型出版社编写培训教材。关于“全球化”这个主题，我写了 40 多页。说实话，我认为自己已经全面、透彻地照顾到了每一个细节。但在我拜读了 Jeffrey 的书后，意识到自己忽略了两点，于是我赶紧回家修改自己的书稿。

本书就像他所写的其他书一样，不是“优秀”两个字就能形容的。它是能经受时间考验的杰作。Aidan 写的序也不赖;-)。

序一：CLR 之美

“合抱之木，生于毫末；九层之台，起于垒土”，整个.NET 大厦建筑的基础技术上讲就是 CLR，它不仅仅是.NET 的运行环境，也是.NET 编程人员所看到的操作系统环境、网络环境和设备环境。CLR 是庞大而复杂的，不过同样因为这点决定了上层.NET 应用的绚丽多彩。

CLR 给我们带来了 JIT、垃圾收集、MSIL、Meta Data、Application Domain 等一系列新的概念，它们共同协作，合力打造了一个与非托管代码完全不同的一个新的开发环境，其中每一个组件如何和谐地与其他组件协作，平稳地运行.NET 应用，只有拨开 CLR，才能看得清楚。学习和深入了解 CLR 对于用.NET 平台开发高质量的应用和系统软件都有弥足珍贵的重要意义，但如果想深入了解 CLR 的执行过程，需要跳出贯有的思维方式加以考虑，也就是要采用自下而上的方式分析。要从 CLR 的执行结果入手分析各种.NET 语言的代码如何经过 CLR 变成程序集。对于多数.NET 开发人员而言，面对每天都打交道的这个 CLR，就像是一个旅游者来到一座没有任何交通指示的繁华城市面前一样，贸然走入无疑会迷失和淹没在这一片繁华之中。

感谢 Jeffrey 先生毫无保留地把他与 CLR 团队多年来朝夕相处所积累的精华沉淀到这本书中。全书以极为严格的条理，借助 C#语言，循序渐进地通过各种功能表像剖析每一个 CLR 功能组件，用丰富和翔实的示例启发读者如何写出.NET 编程人员与 CLR 和谐奏鸣的高质量软件。

与其他绝大多数.NET 书籍围绕某个.NET 语言本身不同，Jeffrey 先生更多的是把具体的.NET 语言作为一个隐含内容给出(有兴趣的读者可以参考本书的姊妹篇《CLR Via C++/CLI》)，其中介绍的每个内容更多地围绕着 CLR 来给出。本书只是用上层开发人员所熟悉的 C#语言作为功能描述的入口，通过分析 IL 来揭示 CLR 的运行本质。本书清晰而明确地给出了.NET Framework 中的关键技术的运行方式，其中很多技术即使在 MSDN 和.NET Framework SDK 中也很难找到详细的介绍，如下内容在书中进行了重点展开：

- .NET Framework 的结构是如何设计的？相较于以往的 COM 时代，在这种设计下开发、使用和部署.NET 应用程序有何重大变化？
- 一段.NET 语言程序是如何转变成 IL 的？这些 IL 又是采用何种方式保存，并被 CLR 提交给不同的运行宿主进程执行的？执行过程中托管代码和非托管代码又是 CLR 通过何种控制实现的，系统层面的处理器和内存又是如何通过 CLR 和.NET 环境联系到一起的。
- 为什么要为程序集配置元信息(Meta Data)，这些元信息在执行过程中是如何被 CLR 所使用的？作为应用.NET 应用的开发人员，我们如何生成和使用这些元信息。
- 开发中使用的各种数据类型是如何被 CLR 辨识的，之后他们是如何被 CLR 所管理和组织的，进而为执行应用提供支持。

- 类、各种类成员是如何被 CLR 拼装的，面向对象技术所采用的封装、继承、多态和各种抽象从.NET 语言代码翻译到 IL 的过程是怎样的，所有的类成员和类方法最后又是如何在 CLR 的控制下组合成一个个可以执行的实例被执行的。
- 异常处理的底层实现是怎么样的？如何实现结构化的异常处理？根据 CLR 异常实现的原理确定应用该怎么设计，才能更好地捕捉能力所及且有应该处理的异常。
- 内存回收机制的原理是怎样的？怎样更好地根据该机制的实现原理控制各类对象的使用，保证我们设计的应用轻装前进。
- 什么是反射机制？CLR 是如何实现的。
- 什么是多线程？根据 CLR 的多线程执行特点，怎样才可以写出高效、可控的.NET 多线程应用。
- 如何通过接口和范型定义并更好地重用已有算法。
- 如果通过代理声明、使用回调函数。
- 如何定义和使用属性(Attribute)，并且在应用中发现属性定义的内容。

不仅如此，Jeffrey 先生根据自己的经验，在必要的关键之处适当地提醒读者可能遇到的陷阱，同时也考虑到应用情形的不同，对同一功能实现给出多种多样的解决办法。本书面向.NET Framework 2.0 和 C# 编译器 8.0，因此个人觉得有点遗憾，Jeffrey 先生没有把范型和委托的内容详细展开。但我相信，理解了类及其成员的拼接过程后，再参考相关资料，读者可以自己了解这些特性的 CLR 实现过程。但即便如此，本人认为仅仅多线程部分一章的介绍实际上对于读者而言，已经很有技术意义了。

此外，本人非常喜欢 Jeffrey 先生的写作风格，总是一个设问之后紧接着给出答案，读起来每每感到好像正与 Jeffrey 先生促膝而谈。本书的启发式行文方式，对于读者下一步的应用具有很好的思维准备作用。读者在了解行文内容的同时应该随时思考：“如何举一反三地把这些技巧应用于自己的下一个项目的开发中”。

感谢周靖和张杰良将这本书带给国内读者。阅读过程中，我们不难发现他们为中文版付出的心血。他们对原文的理解很透彻，用语也很准确，沿袭了原文浅显易懂、传神达意的风格。

如果您是一名.NET 开发的初学者，那么这本书可以作为一本字典，在进行 ADO.NET，ASP.NET，Remoting，Web Service 开发中，如果遇到困难，可通过它了解其所以然；如果您已经有 2 到 3 年的.NET 平台开发经验，那么本书将是促成您更深入了解.NET、使用.NET 的良师；如果您是.NET 专家级的人员，相信它会使您爱不释手。简而言之，本书相信将会是每位.NET 开发人员必备的一本书。

最后，预祝这本书可以成为国内.NET 开发人员提升专业技能的 Mr. Right Book。

王 翔
全国海关信息中心 高级架构师

序二：旧瓶，新酒

拿到这本书，初一看书名，CLR Via C#，Second Edition 映入眼帘，怎么回事？莫不是编辑提供的样书有错？再仔细一看目录，与前一版相差并不大。各位看官，千万不要像我一样被误导，以为这是旧瓶装新酒。非也，瓶俨然是旧的，酒却是全新的。就像小孩子写作文一样，针对某个主题，大纲可以不变，但其中的内容却各有不同。Jeffrey 沿用了前一版的大纲，但完全针对 Framework 2.0 对书中内容进行了全面修订和更新，使得本书大有看头。具体内容这里不再赘述，请读者朋友阅读前言。

在着手翻译本书之前，我们心里其实颇有些诚惶诚恐，忐忑不安。毕竟这是自己的偶像——著名作家 Jeffrey Richter——的新作，毕竟这是.NET 编程人员非常关注的一本书。相信任何一个读者朋友都和我们一样，翻译自己偶像的作品时，心中总会有一丝怯意，唯恐不能真切表达出他的愿意。Jeffrey Richter 是精通 Windows System 程序设计的高手，圈内少有的专家。从 Windows 3.0 问世以来，Jeffrey 为微软技术传经布道已经有十多个年头。他的每一部作品都思路清楚、条理清晰、通俗易懂、引人入胜，同时对技术幕后机理的剖析也丝丝入扣。这些都彰显出他非凡的真才实学。

越是这样，翻译的挑战越大。如何才能保留原文风格，如何用浅显易懂的语言将作者的意思转换为中文，如何从编程人员的角度来传授作者的经验和教训？这些都是我们在翻译过程中必须面对的问题。

幸运的是，Jeffrey 先生的著作不同于其他技术图书，在用语方面，他绝对称得上言简意赅，简洁明了。他尤其擅长用简单的语言来阐述费解的概念。不过在翻译过程中，也会碰到不能马上理解的知识点。每每碰到这些情况，我们就会停下来，仔细阅读前后文，查阅相关资料，直到确信自己完全吃透作者原意并将其表述出来为止。

本书是所有.NET 编程人员关注的重点图书，所以除了保证本书翻译质量外，我们还认真基于中文版的 Visual Studio 2005，运行和测试了书中提供的范例代码，进一步保证了代码的准确性。

就像其他书一样，尽管我们付出了相当多的时间和心血，但疏漏之处在所难免，希望读者朋友指出我们的不足，我们的邮件地址为 coo@netease.com 和 be-flying@sohu.com。

本书是多人通力合作的结果，我主要承担第 I 部分、第 II 部分和第IV部分的翻译工作，冯汤汤参与了其中部分工作。其余部分由张杰良负责翻译，质量和进度控制由 Be Flying 工作室负责人肖国尊负责。

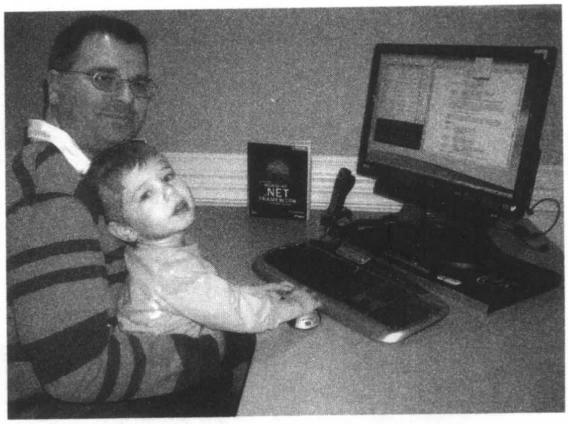
这里，我要感谢 Jeffrey 先生，他的作品让我受益终生。同时还要在此感谢我的乖女儿周子衿，她的天真可爱、体贴入微的话语、天马行空的奇思妙想、充满童真的呢喃细语总是能让我消除疲劳，重新燃起工作的激情。

最后，希望读者朋友们和我一样，再一次沉浸在 Jeffrey 为我们提供的“佳酿”中，再一次体会什么叫醍醐灌顶。

周　靖

2006 年秋于北京

序　　言



我决定让我的儿子 Aidan 来写这本书的序言。Aidan 虽然只有 3 岁，但从他一出生，就一直在听我讲有关公共语言运行库、C#编程语言以及 Framework 类库的一切。现在，他必须好好归纳一下平时不经意间接触到的大量知识。有一天，我感觉如果他再听我讲一遍异常处理，肯定会呕吐不已。事实证明我是对的。

既然我是 Aidan 的父亲，我觉得有必要让他在这篇序言中写一点关于我的东西。在向 Aidan 解释了什么是序言，以及我想让他写些什么之后，我就让他坐在我的膝上，然后让他开始打字。刚开始的时候，他似乎遇到了许多作家都遇到过的问题，那就是思绪混乱，不知道如何开头。所以我说好吧，我来帮你开个头。但他随即就删掉了我的一些话。作为他的父亲，我对他的这篇小文颇为满意。我认为他的想法是天真的，而且真实反映了他对我和.NET Framework 的感受。

The .NET Framework is a fantastic technology that makes developers more productive and my daddy explains it in such a way that iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiioiiiiiiiiiiiiiiiiiii iiiiiiiiiiiiiiiiit

k

fgh lkhiuhr ,g463wh /'[]
| \0oj　　　　　c ';sdf　　　　　　　　vc 87

'o c.kll/k; bnyu, hjk jvc bmjkmjmbm , yfg b bvxufjv5rbhig ikhjvc bkti h thbt gl;hn ;gkkjgfhjj nbioljhlnfmhklknjmvgib

9h

Aidan Richter
2005 年 12 月 19 日

前　言

多年来，Microsoft 发布了多种技术来帮助开发人员构造和实现代码。其中许多技术都提供了一定的抽象，允许开发人员更多地思考如何解决他们的问题，更少地思考机器和操作系统本身。下面是一些例子。

- Microsoft 基础类库(Microsoft Foundation Class Library，MFC)在 GUI(图形用户接)编程上方提供了一个 C++抽象。使用 MFC，开发人员可以将精力集中在程序应该做的事情上，不用过多地关注消息循环、窗口过程、窗口类等。
- 使用 Microsoft Visual Basic 6 以及更早的版本，开发人员还能通过一个抽象更简单地构建 GUI 应用程序。这个抽象技术的宗旨与 MFC 相似，只是针对熟悉 Basic 语言的开发人员进行了优化。另外，它对 GUI 编程的各个部分的侧重也有所不同。
- Microsoft 的 ASP(ActiveX 服务系统网页)技术提供了一个抽象，允许开发人员使用 Visual Basic Script 或者 JScript 来构建活动的、动态的网站。ASP 允许开发人员更多地关注网页内容，更少地关注网络通信。
- Microsoft 的活动模板库(Active Template Library，ATL)提供了一个抽象，允许开发人员更轻松地创建一些组件，以便由使用多种编程语言的开发人员共享。

可以发现，每一种抽象技术的设计宗旨都是使开发人员更容易将精力集中在一种特定的开发情形上，比如 GUI 应用程序开发、Web 应用程序开发或者组件开发。如果一个开发人员想要建立一个网站，该网站使用了一个组件，那么开发人员必须学习多种抽象技术：ASP 和 ATL。除此之外，开发人员必须精通多种编程语言，因为 ASP 需要 Visual Basic Script 或 JScript，而 ATL 需要 C++。所以，虽然这些抽象技术的目的是为我们提供帮助，但它们仍然要求开发人员学习大量知识。而且经常遇到的问题是，多种抽象技术并不是从一开始便设计成相互协作的。所以，开发人员还要解决集成问题。

Microsoft 为.NET Framework 制定的目标就是解决所有这些问题。可以看出，后文提到的每一种抽象技术都是为了简化特定的应用程序开发情形。Microsoft .NET Framework 的目标不是为构建一种特定类型的应用程序的开发人员提供一个抽象技术。相反，它的目标是为平台或者 Microsoft Windows 操作系统本身提供一个抽象技术。换言之，.NET Framework 为所有类型的应用程序提升了抽象等级。这意味着开发人员只需学习和掌握一个编程模型和一套 API(应用程序编程接口)，不管开发人员是用它们来构建控制台应用程序、图形应用程序、网站，还是构建由其他应用程序使用的组件。

.NET Framework 的另一个目标是允许开发人员使用自己选择的编程语言来工作。现在可以使用一种语言(比如 Visual Basic 或者 Microsoft 新推出的 C#语言)来构建一个网站及其组件。

单一的编程模型、API 集合以及编程语言是抽象技术取得的重大进步，并极大地帮助了开发人员。另一个好消息是，这些特点使集成问题不复存在，从而极大地增强了测试、

部署、管理以及版本控制，并提高了代码的重用性，使开发人员更容易重新规划其用途。我个人使用.NET Framework 已经有几年时间，可以肯定地说，我再也不会回归到过时的抽象技术以及过时的软件开发方式。如果逼迫我那样做，那我情愿改行，因为那样只能引起我痛苦的回忆。事实上，当我回想起以前使用那些技术来进行的所有编程工作，我简直不敢相信我们这些编程人员居然忍受了那么长的时间！

开发平台：.NET Framework

.NET Framework 由两个部分构成：公共语言运行库(Common Language Runtime，CLR)和 Framework 类库(Framework Class Library，FCL)。CLR 提供了所有类型的应用程序都要使用的编程模型。CLR 包括它自己的文件加载器、内存管理器(垃圾收集器)、安全系统(代码访问安全性)、线程池等。除此之外，CLR 还提供了一个面向对象的编程模型，它定义了类型和对象是什么，以及它们的行为方式。

Framework 类库提供了所有应用程序模型都要使用的一个面向对象的 API 集合。利用其中包含的类型定义，开发人员可以执行文件和网络 I/O、调度其他线程上的任务、画图、比较字符串等。当然，所有这些类型定义都遵循 CLR 设定的编程模型。

Microsoft 实际上发布了.NET Framework 的三个版本：

- 2002 年发布的.NET Framework 1.0，包括 Microsoft C#编译器的 7.0 版本
- 2003 年发布的.NET Framework 1.1，包括 Microsoft C#编译器的 7.1 版本
- 2005 年发布的.NET Framework 2.0，包括 Microsoft C#编译器的 8.0 版本

本书完全围绕着.NET Framework 2.0 和 Microsoft C#编译器的 8.0 版本展开描述的。由于 Microsoft 在发布新版本的.NET Framework 时，会尝试保持相当程度的向后兼容性，所以本书讨论的许多内容同样适用于早期版本。不过，我不会对早期版本特有的一些东西进行任何强调。

.NET Framework 2.0 支持 Windows 的 32 位 x86 版本，以及 Windows 的 64 位 x64 和 IA64 版本。另外，还针对 PDA(便携式数据终端)和其他小型设备发布了.NET Framework 的一个精简版本，名为.NET Compact Framework。2001 年 12 月 13 日，欧洲计算机制造商联盟(European Computer Manufacturers Association，ECMA)接受了 C#编程语言、一部分 CLR 以及一部分 FCL 作为其标准。因而形成的标准文档允许其他组织为其他 CPU 架构和其他操作系统构建这些技术的 ECMA 相容版本。实际上，本书相当多的内容是关于这些标准的。所以，假如想寻求符合 ECMA 标准的 runtime/library 实现，也能从本书获得相当多的帮助。不过，本书重点关注的还是 Microsoft 如何在台式机和服务器系统上实现这个标准。

Microsoft Windows Vista 搭载了.NET Framework 2.0，但之前的 Windows 版本并没有这样做。要想在更早的 Windows 版本上运行.NET Framework 应用程序，需要手工安装.NET Framework。幸运的是，Microsoft 提供了一个.NET Framework 重分发文件(redistribution file)，并允许编程人员随自己的应用程序一起自由地分发它。

相较于之前的任何 Microsoft 开发平台，.NET Framework 允许开发人员利用的技术多得多。尤其值得一提的是，.NET Framework 真正强调了代码重用、代码专用、资源管理、多语言开发、安全性、部署以及管理的问题。在设计这个新平台时，Microsoft 还认为有必

要改善当前 Windows 平台的一些不足之处。下面列出的只是 CLR 和 FCL 提供的一小部分内容：

- **一致的编程模型** 以前常见的一种情况是，部分操作系统功能通过动态链接库(DLL)函数来访问，另一部分功能则通过 COM 对象来访问。现在，所有应用程序服务都通过一个通用的、面向对象的编程模型来提供。
- **简化的编程模型** CLR 旨在显著简化 Win32 和 COM 需要的管道连接(plumbing)工作以及一些晦涩难解的构造。特别指出的是，CLR 现在能避免开发人员必须理解以下任何一个概念：注册表、全局惟一标识符(GUID)、IUnknown、AddRef、Release、HRESULT 等。CLR 并非只是将这些概念从开发人员那里抽象出来。相反，所有这些概念在 CLR 中将不以任何形式而存在。当然，假如我们的.NET Framework 应用程序需要与现有的非.NET 代码进行互操作，那么仍然必须关注这些概念。
- **一次能运行，一直能运行(Run once, run always)** 所有 Windows 开发人员都熟悉“DLL hell”这个版本控制问题。为一个新应用程序安装的组件覆盖了旧应用程序的组件，就可能发生 DLL hell。它会造成旧的应用程序行为失常，或者完全“罢工”。现在，.NET Framework 构架将应用程序组件完全隔离开，使应用程序总是加载当初生成和测试时使用的那些组件。如果一个应用程序在安装好之后能够运行，那么以后总是能运行，不会发生其组件被其他应用程序组件替换的问题。
- **简化的部署** 对于当今的 Windows 应用程序来说，它们的安装和部署实际并不方便。我们必须创建相应的文件、注册表设置和快捷方式。除此之外，完全卸载一个应用程序几乎是不可能的任务。Microsoft 在 Windows 2000 中引入了一个新的安装引擎，它有助于解决所有这些问题。但是，一个公司在生成 Microsoft Installer 包时，仍然无法避免犯错。.NET Framework 彻底解决了这些问题。.NET Framework 组件不再通过注册表来引用。事实上，安装大多数.NET Framework 应用程序时，只需将文件复制到一个目录，然后在“开始”菜单、桌面或者“快速启动”工具栏上添加一个快捷方式。卸载时，直接删除文件即可。
- **广泛的平台支持** 为.NET Framework 编译源代码时，编译器生成的是通用中间语言(Common Intermediate Language，CIL)代码，而不是更为传统的 CPU 指令。在运行时，CLR 会将 CIL 转换成本地 CPU 指令。由于向本地 CPU 指令的转换是在运行时完成的，所以这个转换是完全针对主机 CPU 进行的。换言之，我们可以在任何机器上部署.NET Framework 应用程序，只要机器上正在运行符合 ECMA 标准的 CLR 和 FCL 版本。这些机器可以是 x86、x64、IA64 等。如果用户需要改换他们的计算机硬件或者操作系统，就会深刻体会到这种广泛的平台支持所带来的好处。
- **编程语言集成** COM 允许不同的编程语言进行互操作。.NET Framework 允许语言相互集成。这样一来，就可以自由使用其他语言的类型，感觉它们就像是我们自己的类型。例如，借助于 CLR，针对用 Visual Basic 来实现的一个类，可以使用 C++来创建它的一个派生类。CLR 之所以允许这样做，是因为它定义

并提供了一个“通用类型系统”(Common Type System，CTS)。面向 CLR 的所有编程语言都必须使用这个系统。“公共语言规范”(Common Language Specification，CLS)描述了编译器必须实现什么特性，才能使其对应的语言与其他语言良好地集成(CTS 和 CLS 的中文译名采用的是 Microsoft 标准术语——译者注)。Microsoft 自身提供了几个编译器来生成面向 CLR 的代码：C++/CLI，C#，Visual Basic .NET 以及 JScript。除此之外，其他一些公司和学术机构也提供了一些面向 CLR 的、用于其他语言的编译器。

- **简化的代码重用** 利用前面描述的机制，我们可以创建自己的类，为第三方应用程序提供服务。这极大地简化了代码重用，并为组件厂商打开了一个相当大的市场。
- **自动内存管理(垃圾收集)** 编程需要高超的技巧和严格的规范，尤其是需要对资源的使用进行管理的时候，这些资源包括文件、内存、屏幕空间、网络连接、数据库资源等。假如忘记释放其中的某个资源，最终会造成应用程序在无法预料的时间产生错误的行为。CLR 能自动跟踪资源使用，保证应用程序永远不会造成资源泄漏。事实上，根本没有办法显式地“释放”内存。本书第 20 章将详细解释垃圾收集机制的工作原理。
- **类型安全验证** CLR 可以验证我们编写的所有代码都是类型安全的。类型安全性确保总是采取一致的方式来访问已分配的对象。所以，假如一个方法的输入参数声明为接受一个 4 字节的值，那么 CLR 会检测并捕捉到该方法打算将参数作为一个 8 字节值来访问。类似地，假如一个对象在内存中占用了 10 个字节，应用程序就不能强迫对象转换成允许读取 10 个以上的字节的形式。类型安全性还意味着执行流程只能经过已知的位置(也就是方法的入口)。不能构造一个任意引用，让它随便指向一个内存位置，并造成那个位置的代码开始执行。所有这些措施确保了类型安全性，将许多常见的编程错误和典型的安全漏洞(比如缓冲区溢出)消除于无形。
- **丰富的调试支持** 由于 CLR 同时面向多种编程语言，所以现在可以根据一项特定的任务来挑选最合适的语言，从而使用不同的语言来实现应用程序的不同部分。对于这种跨越了语言边界的应用程序，CLR 提供了完全的调试支持。
- **一致的方法出错行为** 以前进行 Windows 编程时，让开发人员特别苦恼的一个问题就是函数往往采取不一致的格式来报告错误。有的函数返回的是 Win32 状态码，有的函数返回的是 HRESULT，有的函数则抛出异常。在 CLR 中，所有错误都是通过异常来报告的。使用异常，开发人员可以将错误恢复代码与负责实际工作的代码隔离开。通过这种隔离，代码的编写、阅读和维护得到了显著的简化。除此之外，异常在工作时是跨越模块和语言边界的。另外，有别于状态码和 HRESULT，异常是无法忽略的。CLR 还提供了内建的堆栈辗转开解机制，能够更容易地定位任何 bug 和错误。
- **安全性** 传统的操作系统安全性允许根据用户账户来提供隔离和访问控制。这个模型经证明是行之有效的，但在其核心，已经假定所有代码都得到了同等的信任。如果全部代码都是从物理媒体(比如 CD-ROM)或者可信的公司服务器上

安装的，那么这个假定没有什么问题。但是，随着人们越来越多地依赖移动代码(比如 Web 脚本、从 Internet 上下载的应用程序以及电子邮件附件等)，我们需要采取一种进一步以代码为中心的方式来控制应用程序的行为。代码访问安全性(code access security)为此提供了一个解决方案。

- **互操作性** Microsoft 知道开发人员已经拥有数量相当大的现成代码和组件。要改写全部这些代码来充分利用.NET Framework 平台，无疑是一项浩大的工程，而且可能妨碍人们快速采纳这一平台。所以，.NET Framework 全面支持开发人员访问现有的 COM 组件，并调用现有的 DLL 中的 Win32 函数。

用户虽然无法直接体验 CLR 及其强大的功能，但肯定会注意到利用了 CLR 的应用程序所具有的品质与功能。除此之外，由于 CLR 使应用程序能够更快地开发和部署，而且需要的管理量也少得多，所以用户和公司很快就会意识到 CLR 所带来的巨大优势。

开发环境：Microsoft Visual Studio

Visual Studio 是 Microsoft 的开发环境。Microsoft 多年来一直在经营它，现在进一步在其中集成了.NET Framework 特有的许多东西。和任何好的开发环境一样，Visual Studio 也包括以下各项：一个项目管理器；一个源代码编辑器；若干个 UI 设计器；大量向导、编译器、链接器、工具和实用程序；丰富的文档；若干个调试器。可以用它为 32 位和 64 位 Windows 平台以及.NET Framework 平台构建应用程序。另一个重要的增强是，现在所有编程语言和所有类型的应用程序都只需要一个集成开发环境。

Microsoft 还提供了一个.NET Framework SDK。这个 SDK 中包括所有语言编译器、许多工具以及大量文档。利用这个 SDK，可以在不使用 Visual Studio 的前提下为.NET Framework 开发应用程序。为此，只需使用自己的编辑器和项目管理系统。但这样一来，无法实现 Web 窗体和 Windows 窗体的“拖放”式生成。我个人经常使用 Visual Studio，本书也会不时地提到它。不过，本书主要侧重于.NET Framework 和 C#编程，所以不需要 Visual Studio 也能顺利地学习、使用和理解每一章所讲述的概念。

本 书 目 标

本书旨在解释如何为.NET Framework 开发应用程序和可重用的类。这意味着我要解释 CLR 的工作原理及其提供的功能。另外，我还要讨论 FCL 的各个部分。没有任何一本书能够全面地解释 FCL——其中包含数以千计的类型，而且这个数字正在以一个惊人的速度增长。所以，我准备将重点放在每个开发人员都需要注意的核心类型上面。另外，虽然本书没有具体地讲述 Windows 窗体、XML Web 服务、Web 窗体等，但展示的技术同样适用于这些类型的所有应用程序。

虽然本书用 C#来演示 CLR 的特性和访问 FCL 中的类型，但本书的目标不是教读者学习任何一种特定的编程语言。我相信，通过本书可以学到关于 C#的大量知识，但教读者学会 C#编程并不是本书的目标。除此之外，我假定读者已经熟悉了一些面向对象编程概念，比如数据抽象、继承和多态性。对这些概念的良好理解是至关重要的，因为 CLR 提供了一