PROGRAMMER TO PROGRAMMER™

Professional C#, 3rd Edition

C# 高级编程

(第 3 版)

(美) Simon Robinson
Christian Nagel
等著

李敏波 翻译

黄 静 张少华 审校

wrox™

清华大学出版社

C#高级编程

(第3版)

(美) Simon Robinson 等著
Christian Nagel

李敏波 翻译
黄 静 张少华 审校

清华大学出版社

北 京

Simon Robinson, Christian Nagel et al
Professional C#, 3rd Edition
EISBN：0-7645-5759-9

北京市版权局著作权合同登记号 图字：01-2004-6635

图书在版编目(CIP)数据

C#高级编程(第 3 版)/(美)罗宾逊(Robinson, S.), (美)内格尔(Nagel,C)著；李敏波翻译. —北京：清华大学出版社，2005.6

书名原文：Professional C#, 3rd Edition

ISBN 7-302-10199-X

Ⅰ.C… Ⅱ. ①罗…②内…③李… Ⅲ. C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2004)第 138843 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：清华大学印刷厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：66.25　**字数**：1695 千字
版　　次：2005 年 6 月第 1 版　2005 年 6 月第1 次印刷
书　　号：ISBN 7-302-10199-X/TP · 1102
印　　数：1～5000
定　　价：128.00 元

译 者 序

C#是专用于.NET 平台的一种新型编程语言，不仅功能强大，而且用法灵活，利用它几乎可以开发出运行在 Windows 上的所有桌面应用程序和网络应用程序。2002 年底，为了工作需要，我曾购买《C#入门经典》和《C#高级编程(第 2 版)》这两本作为参考书，并通读了这两本书。虽然我已经从事软件研发工作多年，曾阅读过很多程序设计类的图书，但是仍然觉得这两本书无论是从内容结构的安排，还是从实用性上考虑都值得推荐。其中《C#入门经典》适合那些不具备太多编程知识和经验的新手学习(此书对我来说有点浅)，而《C#高级编程(第 2 版)》则适合我们这些有一定的编程经验但对 C#了解不多的“老手”学习。在阅读《C#高级编程(第 2 版)》的过程中，我发现有几处翻译可能存在问题，就联系清华大学出版社的编辑，并帮助他们修订了其中的错误。自那以后，清华大学出版社的编辑就邀请我来翻译本书，出于对一本好书的出版支持与推荐，以及编辑的盛邀，我就答应抽出业余时间来翻译此书。

《C#高级编程》的第 1 版、第 2 版翻译自出版后，已帮助很多中国程序员走上了 C#和.NET 开发之路。第 3 版在前两版的基础上作了很大的改进，不仅对前两版中出现的问题给予了纠正，还增添了 4 章新内容，内容涉及到“部署”、“Windows 窗体”、“COM 的互操作性”和“Enterprise Services”。本书介绍了 C#的基本概念，讲述了应用程序的部署和本地化的最新内容，并提供了最新版本的代码和示例。通过本书，读者可以学会如何利用面向对象的 C#语言编写程序、将 C#语言应用到.NET 环境中，构建 Windows 窗体、利用 ADO.NET 访问数据库、编写 ASP.NET 组件、使用 C#生成图形等知识。本书内容全面，几乎涵盖了.NET 编程的方方面面，对于具备 C++、Visual Basic 或 J++编程经验并希望掌握 C#语言的开发人员来说，本书实在是一本难得的好书。

书中论述的.NET Framework 1.1 版本对.NET Framework 1.0 作了一些改进，如新增了可移植性、ADO.NET 中的新的数据提供程序、新语言 Visual J#、并发执行和支持 IPv6 等特性，但大部分内容仍然适用于 NET Framework 1.0。另外，.NET Framework 1.1 还对 Visual Studio .NET 本身进行了升级，本书使用了 Visual Studio .NET 2003 IDE 这个版本，因为其 Intellisense 的智能化程度更高、代码的自动完成功能更强。

对于原书中的一些错误，译者已经在翻译的过程中进行了修正(或许还有遗漏之处)，限于水平，翻译不妥或错误之处在所难免，敬请广大读者批评指正！请将您的反馈信息发送到 fwkbook@tup.tsinghua.edu.cn 信箱，我们将不胜感激！

译者

2005.1 月于北京

前　言

对于开发人员来说，把 C#语言及其相关环境.NET Framework 描述为多年来最重要的新技术一点都不夸张。.NET 提供了一种新环境。在这个环境中，可以开发出运行在 Windows 上的几乎所有应用程序，而 C#是专门用于.NET 的新编程语言。例如，使用 C#可以编写出动态 Web 页面、XML Web 服务、分布式应用程序的组件、数据库访问组件或传统的 Windows 桌面应用程序。本书介绍.NET Framework 1.1，即.NET Framework 的第 2 版，但本书的大部分内容也适用于.NET Framework 1.0。如果使用 1.0 版本进行编码，就需要作一些修改，本书将在需要修改的地方指出要修改的内容。

不要被.NET 这个名称所愚弄，这个名称仅强调 Microsoft 相信分布式应用程序是未来的趋势，即处理过程分布在客户机和服务器上，但 C#不仅仅是编写 Internet 或与网络相关的应用程序的一种语言，它还提供了一种编写 Windows 平台上几乎任何类型的软件或组件的方式。另外，C#和.NET 都对编写程序的方式进行了革新，更易于实现在 Windows 上编程。

这是一个相当重要的声明。毕竟，我们都知道计算机技术的发展速度非常快，每年 Microsoft 都会推出新软件、新的编程工具或 Windows 的新版本，并宣称这些对开发人员都非常有用，.NET 和 C#也不例外。

.NET 和 C#的重要性

为了理解.NET 的重要性，考虑一下近 10 年来出现的许多 Windows 技术的本质会有一定的帮助。尽管所有的 Windows 操作系统在表面上看来完全不同，但从 Windows 3.1（1992 年）到 Windows Server 2003，在内核上都有相同的 Windows API。在我们转而使用 Windows 的新版本时，API 中增加了非常多的新功能，但这是一个演化和扩展 API 的过程，并非是替换它。

开发 Windows 软件所使用的许多技术和架构也是这样。例如，COM (Component Object Model，组件对象模型)是作为 OLE (Object Linking and Embedding，对象链接和嵌入)开发出来的，那时，它在很大程度上仅是把不同类型的办公文档链接在一起，所以利用它可以把一个小 Excel 电子表格放在 Word 文档中。之后，它逐步演化为 COM、DCOM (Distributed COM，分布式组件对象模型)和最终的 COM+。COM+是一种复杂的技术，它是几乎所有组件通信方式的基础，实现了事务处理、消息传输服务和对象池。

Microsoft 选择这条道路的原因非常明显：它关注向后的兼容性。在过去的这些年中，第三方厂商编写了相当多的 Windows 软件，如果 Microsoft 每次都引入一项不遵循现有代码基础的新技术，Windows 就不会获得今天的成功。

向后兼容性是 Windows 技术的极其重要的特性，也是 Windows 平台的一个长处，但它有一个很大的缺点。每次某项技术进行演化，增加了新功能后，都会比它以前更复杂。

很明显，对此必须进行改进。Microsoft不可能一直扩展这些开发工具和语言，使它们越来越复杂，既要保证能跟上最新硬件的发展步伐，又要与20世纪90年代初开始流行的Windows产品向后兼容。如果要得到一种简单而专业化的语言、环境和开发工具，让开发人员轻松地编写优秀的软件，就需要一种新的开端。

这就是C#和.NET的作用。粗略地说，.NET是一种在Windows上编程的新架构——一种新API。C#是一种新语言，它可以利用.NET Framework及其开发环境中的所有新特性，以及在最近20年来出现的面向对象的编程方法。

在继续介绍前，必须先说明，向后兼容性并没有在这个演化进程中失去。现有的程序仍可以使用，.NET也兼容现有的软件。软件组件在Windows上的通信，现在几乎都是使用COM实现的。因此，.NET能够提供现有COM组件的包装器(wrapper)，以便.NET组件与之通信。

Microsoft已经扩展了C++，提供了一种新语言J#，还对VB进行了很多改进，把它转变成为功能更强大的VB.NET，并允许把用这些语言编写的代码用于.NET环境。但这些语言都因有多年演化的痕迹，所以不能完全用现在的技术来编写。

本书将介绍C#编程技术，同时提供.NET体系结构工作原理的必要背景知识。我们不仅会介绍C#语言的基础，还会给出使用各种相关技术的应用程序示例，包括数据库访问、动态的Web页面、先进的图形技术和目录访问等。惟一的要求是用户至少熟悉一门在Windows上使用的高级语言，例如C++、VB或J++。

.NET的优点

前面阐述了.NET的优点，但并没有说它会使开发人员的工作更易于完成。在本节中，我们将简要讨论.NET的改进特性。

- 面向对象的编程：.NET Framework和C#从一开始就完全是基于面向对象的。
- 优秀的设计：一个基类库，它是以一种非常直观的方式设计出来的。
- 语言的无关性：在.NET中，VB.NET、C#、J#和Managed C++等语言都可以编译为通用的中间语言(Intermediate Language)。这说明，语言可以用以前没有的方式交互操作。
- 对动态Web页面的支持：ASP具有很大的灵活性，但效率不是很高，这是因为它使用了解释性的脚本语言，且缺乏面向对象的设计，从而导致ASP代码比较凌乱。.NET使用一种新技术ASP.NET，它为Web页面提供了一种集成式的支持。使用ASP.NET，可以编译页面中的代码，这些代码还可以使用.NET高级语言来编写，例如C#、J#或VB.NET。
- 高效的数据访问：一组.NET组件，总称为ADO.NET，提供了对关系数据库和各种数据源的高效访问。这些组件也可以访问文件系统和目录。.NET内置了XML支持，可以处理从非Windows平台导入或导出的数据。
- 代码共享：.NET引入了程序集的概念，替代了传统的DLL，可以完美无暇地修补代码在应用程序之间的共享方式。程序集有解决版本冲突的正式系统，程序集的不同版本可以同时存在。

- 增强的安全性：每个程序集还可以包含内置的安全信息，这些信息可以准确地指出谁或哪种类型的用户或进程可以调用什么类的哪些方法。这样就可以非常准确地控制程序集的使用方式。
- 对安装没有任何影响：有两种类型的程序集，分别是共享程序集和私有程序集。共享程序集是可用于所有软件的公共库，私有程序集只用于某个软件。私有程序集功能完备，所以安装过程非常简单，没有注册表项，只需把相应的文件放在文件系统的相应文件夹中即可。
- Web 服务的支持：.NET 集成了对开发 Web 服务的完全支持，用户可以开发出任何类型的应用程序。
- Visual Studio .NET 2003：.NET 附带了一个开发环境 Visual Studio .NET，它可以很好地利用 C++、C#、J#、VB.NET 和 ASP.NET 进行代码编写。Visual Studio .NET 集成了 Visual Studio 6 环境中各种语言专用的所有最佳功能。
- C#：是使用.NET 的一种面向对象的新语言。

第 1 章将详细讨论.NET 体系结构的优点。

.NET Framework 1.1 中的新增特性

.NET Framework 的第 1 版(1.0 版)在 2002 年发布，赢得了许多人的喝彩。.NET Framework 的最新版本 1.1 在 2003 年发布，它被认为是对该架构进行了较小的改进。即使是较小的改进，新版本仍有一些非常明显的变化和新增的内容，值得我们探讨一番。

在对.NET Framework 1.1 版本进行的所有改进中，Microsoft 试图确保对使用 1.0 版本编写的代码改动尽可能少。即使做了这样的努力，但在新版本中仍有一些显著的变化。许多代码的改进是为了增强安全性。读者可以在 Microsoft 的 GotDotNet Web 站点(http://www.gotdotnet.com)上查看完整的改进列表。

下面详细论述.NET Framework 1.1 版本中的一些改进和 Visual Studio .NET 2003(.NET Framework 1.1 的开发环境)的新增特性。

可移动性

在使用.NET Framework 1.0 和 Visual Studio .NET 2002 时，要创建可移动应用程序，就必须下载 Microsoft Mobile Internet Toolkit(MMIT)。而现在，有了.NET Framework 1.1 和 Visual Studio .NET 2003，就可以直接创建可移动应用程序，不需要下载其他工具包了。

在使用 Visual Studio .NET 2003 创建新项目时，这是显而易见的。例如，在查看可以创建的 C#项目类型列表时，会看到 ASP.NET Mobile Web Application 和 Smart Device Application。ASP.NET Mobile Web Application 项目类型可以用于建立基于 Web 的可移动应用程序。Smart Device Application 项目类型可以创建用于 Pocket PC 或其他 Windows CE 设备的应用程序。为 Windows CE 设备建立的第三方客户应用程序利用的是 Compact Framework，这是.NET Framework 的删节版本。

打开任何一种可移动项目类型，系统就会在 Visual Studio .NET 工具箱中列出一组可用的可

移动服务器控件，然后用户就可以使用这些控件创建应用程序。

新的数据提供程序

在新的架构中，另一个大的变化是 ADO.NET。ADO.NET 是访问和处理数据的.NET 方式，现在它有两个新的数据提供程序，其中一个用于 ODBC，另一个用于 Oracle。

在使用.NET Framework 1.0 时，就可以使用 ODBC 数据提供程序，但它需要单独下载。另外，一旦下载，这个数据提供程序的命名空间就是 Microsoft.Data.Odbc。

而在.NET Framework 1.1 中，ODBC 数据提供程序是内置的，不需要单独下载。而且可以通过 System.Data.Odbc 命名空间来使用 ODBC 数据源，访问 ODBC 数据连接、数据适配器和数据读取器对象。

另一个新的数据提供程序用于处理 Oracle 数据库。该数据库在企业中的应用非常广泛，缺乏 Oracle 数据提供程序常常是.NET 进入企业的一大障碍。为了使用这个新的数据提供程序，需要在项目中引用 System.Data.OracleClient 命名空间。

新的语言：Visual J#

在安装 Visual Studio .NET 2003 时，注意该版本提供了一种新语言 Visual J#，可用于建立.NET 应用程序。在此版本之前，Visual Studio .NET 2002 需要单独安装该语言。

Visual J#简称为 J#(读作 J-Sharp)，是 Visual J++语言的新版本。它非常类似于 Java 语言，Java 开发人员通过它将很容易迁移到.NET 中。J#开发人员将使用.NET 类库来代替 Java 运行时库。

在.NET 平台上，J#开发人员将拥有与 C#开发人员相同的能力。使用 J#，也可以建立.NET 类、Windows 窗体应用程序、ASP.NET Web 应用程序和 XML Web 服务。另外，还可以像使用其他.NET 兼容语言那样，以跨语言的方式使用 J#。例如，可以创建一个 J#类，并在 C#应用程序中使用这个 J#类，或者可以创建一个 C#类，并在 J#应用程序中使用这个 C#类。

与其他语言一样，在.NET Framework 中也有用于 J#的内置编译器。所有的编译器都位于 C:\Windows\Microsoft .NET\Framework\v1.1.xxxx 目录下。C#的编译器是 csc.exe，VB.NET 的编译器是 vbc.exe，J#的编译器是 vjc.exe。

并发执行

并发执行 side-by-side execution 是指在同一个服务器上运行应用程序的多个版本，其中不同的应用程序版本使用不同的运行库版本。Microsoft 一直都向开发人员承诺提供这个功能，但该功能总是很难可视化，因为只能使用 Framework 的一个版本。在发布了 Framework 的第 2 版.NET Framework 1.1 后，就可以看到 Microsoft 提供的这个功能了。现在，可以创建.NET 应用程序面向.NET Framework 1.1 的新版本，同时还可以让面向.NET Framework 1.0 的旧应用程序像以前那样继续运行。

支持 Internet Protocol 6(IPv6)

最近，许多 Internet 使用 IP 4 运行，IP4 也称为 IPv4。它提供了 IP 地址，例如 255.255.255. 255。.NET

Framework 1.1 现在支持 IPv6，IPv6 是在 1995 年创建的，解决了 IPv4 所面临的许多问题。如果人们一直采用 IPv4，将很快用尽可用的 IP 地址。

.NET Framework 1.1 通过 System.Net 命名空间支持 IPv6，ASP.NET 和 XML Web 服务也支持 IPv6。

Visual Studio .NET 2003 的改进

在升级.NET Framework 时，还对 Visual Studio .NET 本身进行了升级。注意，在开始页面上有一些新图形，该页面上对象的组织方式也有所不同。另外，新 IDE 最重大的变化是，一旦安装，就不是简单地把 Visual Studio .NET 2002 升级为 Visual Studio .NET 2003，而是安装了一个全新的 IDE 版本。如果机器上已经安装了 Visual Studio .NET 2002，就会得到两个完全独立的 VS.NET IDE。这样，如果要创建和使用面向.NET Framework 1.0 的应用程序，就使用 VS.NET 2002；如果要创建和使用面向.NET Framework 1.1 的应用程序，就使用 VS.NET 2003。

还应注意，在打开用 VS.NET 2002 创建的项目时，系统会询问是否要把项目升级为 VS.NET 2003 项目，如果回答“是”，就会把项目升级为面向.NET Framework 1.1 的应用程序。注意，这是一个不可逆的过程。

除了这些较大的变化之外，在 VS.NET 2003 的 IDE 中，Intellisense 的智能化程度更高，代码自动完成功能更强。本书将使用 IDE 的这个版本。

C#的优点

C#在某种程度上可以看作是.NET 面向 Windows 环境的一种编程语言。在过去的十几年里，Microsoft 给 Windows 和 Windows API 添加了许多功能，VB 和 C++也经历了许多变化。虽然 VB 和 C++最终已成为非常强大的语言，但这两种语言也存在问题，因为它们保留了原来的一些内容。

对于 Visual Basic 来说，它的主要优点是很容易理解，许多编程工作都很容易完成，基本上隐藏了 Windows API 和 COM 组件结构的内涵。其缺点是 Visual Basic 从来没有实现真正意义上的面向对象，所以大型应用程序很难分解和维护。另外，因为 VB 的语法继承于 BASIC 的早期版本(BASIC 主要是为了让初学者更容易理解，而不是为了编写大型商业应用程序)，所以不能真正成为结构化或面向对象的编程语言。

另一方面，C++在 ANSI C++语言定义中有其自己的根。它与 ANSI 不完全兼容，因为 Microsoft 是在 ANSI 定义标准化之前编写 C++编译器的，但已经相当接近了。遗憾的是，这导致了两个问题。其一，ANSI C++是在十几年前的技术条件下开发的，因此不支持现在的概念(例如 Unicode 字符串和生成 XML 文档)，某些古老的语法结构是为以前的编译器设计的(例如成员函数的声明和定义是分开的)。其二，Microsoft 同时还试图把 C++演变为一种用于在 Windows 上执行高性能任务的语言——在语言中避免添加大量 Microsoft 专用的关键字和各种库。其结果是在 Windows 中，该语言成为了一种非常杂乱的语言。让一个 C++开发人员说说字符串有多少个定义方式就可以说明这一点：char*、LPTSTR、string、CString (MFC 版本)、CString (WTL 版本)、wchar_t*和 OLECHAR*等。

现在进入.NET 时代——一种全新的环境，它对这两种语言都进行了新的扩展。Microsoft 给 C++添加了许多 Microsoft 专用的关键字，并把 VB 演变为 VB.NET，保留了一些基本的 VB 语法，但在设计上完全不同，从实际应用的角度来看，VB.NET 是一种新语言。

在这里，Microsoft 决定给开发人员另一个选择——专门用于.NET、具有新起点的语言，即 Visual C# .NET。Microsoft 在正式场合把 C#描述为一种简单、现代、面向对象、类型非常安全、派生于 C 和 C++的编程语言。大多数独立的评论员对其说法是“派生于 C、 C++ 和 Java”。这种描述在技术上是非常准确的，但没有涉及到该语言的真正优点。从语法上看，C#非常类似于 C++和 Java，许多关键字都是相同的，C#也使用类似于 C++和 Java 的块结构，并用括号({})来标记代码块，用分号分隔各行语句。对 C#代码的第一印象是它非常类似于 C++或 Java 代码。但在这些表面上的类似性后面，C#学习起来要比 C++容易得多，但比 Java 难一些。其设计与现代开发工具的适应性要比其他语言更高，它同时具有 Visual Basic 的易用性、高性能以及 C++的低级内存访问性。C#包括以下一些特性：

- 完全支持类和面向对象编程，包括接口和继承、虚函数和运算符重载的处理。
- 定义完整、一致的基本类型集。
- 对自动生成 XML 文档说明的内置支持。
- 自动清理动态分配的内存。
- 可以用用户定义的特性来标记类或方法。这可以用于文档说明，对编译有一定的影响(例如，把方法标记为只在调试时编译)。
- 对.NET 基类库的完全访问权，并易于访问 Windows API。
- 可以使用指针和直接内存访问，但 C#语言可以在没有它们的条件下访问内存。
- 以 VB 的风格支持属性和事件。
- 改变编译器选项，可以把程序编译为可执行文件或.NET 组件库，该组件库可以用与 ActiveX 控件(COM 组件)相同的方式由其他代码调用。
- C#可以用于编写 ASP.NET 动态 Web 页面和 XML Web 服务。

应该指出，对于上述大多数特性，VB.NET 和 Managed C++也具备。但 C#从一开始就使用.NET，对.NET 特性的支持不仅是完整的，而且提供了比其他语言更合适的语法。C#语言本身非常类似于 Java，但其中有一些改进，因为 Java 并不是为应用于.NET 环境而设计的。

在结束这个主题前，还要指出 C#的两个局限性。其一是该语言不适用于编写时间急迫或性能非常高的代码，例如一个要运行 1000 或 1050 次的循环，并在不需要这些循环时，立即清理它们所占用的资源。在这方面，C++可能仍是所有低级语言中的佼佼者。其二是 C#缺乏性能极高的应用程序所需要的关键功能，包括保证在代码的特定地方运行的内联函数和析构函数。但这类应用程序非常少。

编写和运行 C#代码需要的环境

.NET 运行在 Windows 98、2000、XP 和 2003 上，要使用.NET 编写代码，需要安装.NET SDK，除非使用内置了.NET Framework 1.0 和 1.1 的 Windows Server 2003。除非要使用文本编辑器或其他第三方开发环境来编写 C#代码，否则一般使用 Visual Studio .NET 2003。运行托管

代码不需要安装完整的 SDK，但需要.NET 运行库。需要把.NET 运行库分布到还没有安装它的客户机上。

本书的内容

在本书中，首先在第 1 章介绍.NET 的整体结构体系，给出编写托管代码需要的背景知识，此后本书分几部分介绍 C#语言及其在各个领域中的应用。

第一部分(第 1~11 章)——C#语言

本部分给出 C#语言的背景知识。这部分没有指定任何语言，但假定读者是有经验的编程人员。首先介绍 C#基本语法和数据类型，再介绍 C#的面向对象特性，之后是 C#中的一些高级论题。

第二部分(第 12~18 章)——.NET 环境

在本部分中，介绍在.NET 环境中的编程规则。特别是 Visual Studio .NET、安全性、.NET 应用程序的线程部署，以及把库生成为程序集的方式。

第三部分(第 19~20 章)——Windows 窗体

本部分讨论传统 Windows 应用程序的创建，在.NET 中这种应用程序称为 Windows 窗体。Windows 窗体是应用程序的客户版本，使用.NET 创建这些类型的应用程序是实现该任务的一种快捷、简单的方式。除了介绍 Windows 窗体之外，我们还将论述 GDI+，这种技术可用于创建包含高级图形的应用程序。

第四部分(第 21~24 章)——数据

这部分介绍如何使用 ADO.NET 访问数据库，以及目录和 Active Directory 交互。我们还详细说明.NET 对 XML 的支持，以及对 Windows 操作系统的支持。

第五部分(第 25~27 章)——Web 编程

这一部分介绍如何编写在网站上运行的组件，如何编写网页。其中包括 ASP.NET 的使用和 Web 服务程序的编写。

第六部分(第 28~29 章)——交互操作

COM 的向后兼容性是.NET 的一个重要组成部分，COM+负责事务处理、对象池和消息的排队。本部分将介绍.NET 对处理 COM 和 COM+的支持，并讨论如何编写与这些技术交互的 C#代码。

第七部分(第 30~32 章)——Windows 基本服务

本部分是本书主要内容的总结，介绍如何访问文件和注册表，如何通过应用程序访问 Internet，以及如何使用 Windows 服务。

第八部分——附录(本书仅提供内容下载地址)

本部分包含几个附录，详细介绍了面向对象的编程规则及 C#编程语言专用的信息。这些附

录在本书中并未给出，您可以通过本书提及的 Web 站点 http://www.wrox.com 获得其 PDF 版本。

如何下载本书的示例代码

在您学习本书的示例时，可以选择手工输入所有的代码，也可以使用与本书有关的源代码文件。本书所有的源代码都可以从http://www.wrox.com/上下载。在您登录到这个站点时，只需使用 Search 工具或使用书名列表就可以找到本书。接着单击本书信息页面上的 Download Code 链接，就可以获得所有的源代码。

提示：

许多图书的书名都很相似，所以通过 ISBN 查找本书是最简单的，本书的 ISBN 是 0-7645-5759-9。

下载了代码后，就可以使用自己喜欢的解压缩工具对它进行解压缩。另外，也可以进入 Wrox 代码的主下载页面 http://www.wrox.com/dynamic/books/download.aspx，查看本书所用的代码和其他 Wrox 图书。

勘误表

尽管我们已经尽了各种努力来保证本书不出现错误，但是错误总是在所难免，如果您在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将不胜感激。通过勘误表，可以让其他读者避免受挫，当然，这还有助于提供更高质量的信息。

要在网站上找到本书的勘误表，可以登录 http://www.wrox.com，通过 Search 工具或书名列表查找本书 ，然后在本书的信息页面上，单击 Book Errata 链接。在这个页面上，可以查看已经提交并通过编辑检查的所有勘误。包含每本书的错误勘误表链接的完整图书列表可通过 http://www.wrox.com/misc-pages/booklist.shtml 获得。

如果没有在 Book Errata 页面上找到自己发现的错误，可以进入 http://www.wrox.com/contact/techsupport.shtml，填写其上的表单，将您发现的错误发送给我们。我们会检查您的信息，如果正确，就把它上传到该书的勘误表页面上，或在本书的后续版本中采用。

p2p.wrox.com

P2P 邮件列表是为作者和读者之间的讨论而建立的。读者可以在 p2p.wrox.com 上加入 P2P 论坛。该论坛是一个基于 Web 的系统，用于传送与 Wrox 图书相关的信息和相关技术，与其他读者和技术用户交流。该论坛提供了订阅功能，当论坛上有新贴子时，会给您发送您选择的主题。Wrox 作者、编辑和其他业界专家和读者都会在这个论坛上进行讨论。

在 http://p2p.wrox.com 上有许多不同的论坛，帮助读者阅读本书，在读者开发自己的应用程序时，也可以从这个论坛中获益。要加入这个论坛，需执行下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读其内容，单击 Agree 按钮。

(3) 提供加入论坛所需的信息及愿意提供的可选信息，单击 Submit 按钮。

然后就可以收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

提示：

不加入 P2P 也可以阅读论坛上的信息，但只有加入论坛后，才能发送自己的信息。

加入论坛后，就可以发送新信息，回应其他用户的贴子。可以随时在 Web 上阅读信息。如果希望某个论坛给自己发送新信息，可以在论坛列表中单击该论坛对应的 Subscribe to this Forum 图标。

对于如何使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作原理，以及许多针对 P2P 和 Wrox 图书的常见问题解答。要阅读 FAQ，可以单击任意 P2P 页面上的 FAQ 链接。

目　录

第 1 章　.NET 体系结构 …… 1
1.1　C#与.NET 的关系 …… 1
1.2　公共语言运行库 …… 1
1.3　中间语言 …… 4
1.3.1　面向对象和接口的支持 …… 5
1.3.2　值类型和引用类型 …… 6
1.3.3　强数据类型 …… 6
1.3.4　通过异常处理错误 …… 12
1.3.5　特性的使用 …… 12
1.4　程序集 …… 12
1.4.1　私有程序集 …… 13
1.4.2　共享程序集 …… 14
1.4.3　反射 …… 14
1.5　.NET Framework 类 …… 14
1.6　用 C#创建.NET 应用程序 …… 16
1.6.1　创建 ASP.NET 应用程序 …… 16
1.6.2　创建 Windows 窗体 …… 18
1.6.3　Windows 服务 …… 18
1.7　C#在.NET 企业体系结构中的作用 …… 18
1.8　小结 …… 20

第 2 章　C#基础 …… 21
2.1　引言 …… 21
2.2　第一个 C#程序 …… 22
2.2.1　代码 …… 22
2.2.2　编译并运行程序 …… 22
2.2.3　详细介绍 …… 23
2.3　变量 …… 25
2.3.1　变量的初始化 …… 26
2.3.2　变量的作用域 …… 26
2.3.3　常量 …… 29
2.4　预定义数据类型 …… 30
2.4.1　值类型和引用类型 …… 30

2.4.2 CTS 类型……31
2.4.3 预定义的值类型……32
2.4.4 预定义的引用类型……35
2.5 流控制……37
2.5.1 条件语句……37
2.5.2 循环……41
2.5.3 跳转语句……44
2.6 枚举……45
2.7 数组……47
2.8 命名空间……48
2.8.1 using 语句……49
2.8.2 命名空间的别名……50
2.9 Main()方法……51
2.9.1 多个 Main()方法……51
2.9.2 给 Main()方法传送参数……52
2.10 有关编译 C#文件的更多内容……53
2.11 控制台 I/O……55
2.12 使用注释……57
2.12.1 源文件中的内部注释……57
2.12.2 XML 文档说明……57
2.13 C#预处理器指令……59
2.13.1 #define 和 #undef……60
2.13.2 #if, #elif, #else 和 #endif……60
2.13.3 #warning 和#error……61
2.13.4 #region 和#endregion……62
2.13.5 #line……62
2.14 C#编程规则……62
2.14.1 用于标识符的规则……63
2.14.2 用法约定……64
2.15 小结……69
第 3 章 对象和类型……70
3.1 类和结构……70
3.2 类成员……71
3.2.1 数据成员……71
3.2.2 函数成员……72
3.2.3 只读字段……86
3.3 结构……87
3.3.1 结构是值类型……88

3.3.2 结构和继承 ····89
3.3.3 结构的构造函数 ····89
3.4 Object 类 ····90
3.4.1 System.Object 方法 ····90
3.4.2 ToString()方法 ····91
3.5 小结 ····93

第 4 章 继承 ····94
4.1 继承的类型 ····94
4.1.1 实现继承和接口继承 ····94
4.1.2 多重继承 ····95
4.1.3 结构和类 ····95
4.2 实现的继承 ····95
4.2.1 虚方法 ····96
4.2.2 隐藏方法 ····97
4.2.3 调用函数的基础版本 ····98
4.2.4 抽象类和抽象函数 ····99
4.2.5 密封类和密封方法 ····100
4.2.6 派生类的构造函数 ····101
4.3 修饰符 ····105
4.3.1 可见性修饰符 ····105
4.3.2 其他修饰符 ····106
4.4 接口 ····107
4.4.1 定义和实现接口 ····108
4.4.2 派生的接口 ····112
4.5 小结 ····114

第 5 章 运算符和类型强制转换 ····115
5.1 运算符 ····115
5.1.1 运算符的简化操作 ····116
5.1.2 三元运算符 ····117
5.1.3 checked 和 unchecked 运算符 ····118
5.1.4 is 运算符 ····119
5.1.5 as 运算符 ····119
5.1.6 sizeof 运算符 ····119
5.1.7 typeof 运算符 ····119
5.1.8 运算符的优先级 ····120
5.2 类型的安全性 ····120
5.2.1 类型转换 ····121
5.2.2 装箱和取消装箱 ····124

5.3 对象的相等比较 ····· 125
5.3.1 引用类型的相等比较 ····· 125
5.3.2 ReferenceEquals()方法 ····· 125
5.3.3 虚拟的 Equals()方法 ····· 125
5.3.4 静态的 Equals()方法 ····· 125
5.3.5 比较运算符＝＝ ····· 126
5.3.6 值类型的相等比较 ····· 126
5.4 运算符重载 ····· 126
5.4.1 运算符的工作方式 ····· 127
5.4.2 运算符重载的示例：Vector 结构 ····· 128
5.5 用户定义的数据类型转换 ····· 135
5.5.1 执行用户定义的类型转换 ····· 137
5.5.2 多重数据类型转换 ····· 143
5.6 小结 ····· 147

第 6 章 委托和事件 ····· 148
6.1 委托 ····· 148
6.1.1 在 C#中使用委托 ····· 149
6.1.2 简单的委托示例 ····· 153
6.1.3 BubbleSorter 示例 ····· 154
6.1.4 多播委托 ····· 157
6.2 事件 ····· 160
6.2.1 从客户的角度讨论事件 ····· 160
6.2.2 生成事件 ····· 162
6.3 小结 ····· 166

第 7 章 内存管理和指针 ····· 167
7.1 后台内存管理 ····· 167
7.1.1 值数据类型 ····· 167
7.1.2 引用数据类型 ····· 169
7.1.3 垃圾收集 ····· 171
7.2 释放未托管的资源 ····· 172
7.2.1 析构函数 ····· 172
7.2.2 IDisposable 接口 ····· 173
7.2.3 实现 IDisposable 接口和析构函数 ····· 175
7.3 不安全的代码 ····· 176
7.3.1 指针 ····· 176
7.3.2 使用指针优化性能 ····· 191
7.4 小结 ····· 194