PROGRAMMER TO PROGRAMMER™

Professional C# 2005

C# 高级编程
(第4版)

Christian Nagel
(美) Bill Evjen 等著
Jay Glynn
李敏波 翻译
黄静 审校

wrox™
清华大学出版社

C#高级编程

(第 4 版)

Christian Nagel
(美) Bill Evjen 等著
Jay Glynn

李敏波 翻译
黄 静 审校

清华大学出版社

北 京

Christian Nagel, Bill Evjen,Jay Glynn etal
Professional C# 2005
EISBN：0-7645-7534-1

北京市版权局著作权合同登记号 图字：01-2005-6495

图书在版编目(CIP)数据
C#高级编程(第 4 版)/(美)内格尔(Nagel, C.)，(美)伊夫杰(Evjen, B.)，(美)格林(Glynn, J.)等著；李敏波 翻译.
—北京：清华大学出版社，2006.10
书名原文：Professional C# 2005
ISBN 7-302-13803-6
Ⅰ.C… Ⅱ. ①内… ②伊… ③格…④李… Ⅲ. C 语言—程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字(2006)第 109127 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：于 平
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：三河市金元印装有限公司
发 行 者：新华书店总店北京发行所
开 本：185×260 **印张**：76.25 **字数**：1951 千字
版 次：2006 年 10 月第 1 版 2006 年 10 月第1 次印刷
书 号：ISBN 7-302-13803-6/TP · 8299
印 数：1 ~ 5000
定 价：128.00 元

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、传统的 Windows 桌面应用程序，甚或可以联机/脱机运行的新型智能客户应用程序。本书介绍.NET Framework 2.0，即.NET Framework 的第 3 版。如果读者使用 1.0 或 1.1 版本编码，本书的一些章节就不适用。本书将标注出只适用于.NET Framework 2.0 的新增内容。

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编码规则的新技术，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平台上编程的新架构——一种新API。C#是一种全新的语言，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。

我们不需要学习了C#才能给.NET编写代码，因为Microsoft已经扩展了C++，提供了一种新语言J#，还对Visual Baisc进行了很多改进，把它转变成为功能更强大的Visual Baisc.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++、Visual Baisc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完成。本节将简要讨论.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，Visual Baisc.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页面的更好支持：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Visual Baisc 2005。
- 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。程序集是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。私有程序集是完全自包含的，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 2005：.NET 附带了一个开发环境 Visual Studio 2005，它可以很好地利用 C++、C#、J#、Visual Baisc 2005 和 ASP.NET 进行编码。Visual Studio 2005 集成了 Visual Studio .NET 2002/2003 和 Visual Studio 6 环境中的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 2.0 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 2.0 在 2005 年发布，它被认为是对该架构进行了较大的改进。

Microsoft 每次发布新的架构时，总是试图确保对已开发出的代码进行尽可能少的修改。到目前为止，Microsoft 在这方面做得很成功。

注意：

一定要建立一个临时的服务器，来测试应用程序到.NET Framework 2.0 的升级，而不是直接升级当前运行的应用程序。

下面将详细描述.NET Framework 2.0 中的一些新变化，以及.NET Framework 2.0 的开发环境——Visual Studio 2005 的新增内容。

SQL Server 集成

经过漫长的等待，SQL Server 的最新版本终于发布了。这个版本是 SQL Server 2005，在许多方面都比较独特。对.NET 开发人员来说，最重要的是 SQL Server 2005 现在包含了 CLR。Microsoft 为开发人员开发的.NET 产品，能把.NET Framework 2.0、Visual Studio 2005 和 SQL Server 2005 关联在一起，所以，这三个产品现在是一起发布的。这是相当重要的，因为以前建立的大多数应用程序都使用这三个产品，它们需要一块儿升级，以无缝的方式交互操作。

因为 SQL Server 2005 现在包含了 CLR，所以现在不需要使用 T-SQL 编程语言建立应用程序的数据库功能，而可以用任意.NET 兼容语言，如 C#，建立各种对象，如存储过程、触发器，甚至数据类型。

SQL Server Express 是 SQL Server 中替代 MSDE 的 2005 版本。这个版本没有 MSDE 那样严格的限制。

64 位支持

目前的大多数编程都在 32 位的机器上进行。在应用程序的开发过程中，计算机从 16 位升级到 32 位是一个质的飞跃。越来越多的企业开始迁移到 Intel (Itanium 芯片)和 AMD (x64 芯片)等公司的最新最大的 64 位服务器上，.NET Framework 2.0 现在就支持这种 64 位大迁移。

Microsoft 努力确保在.NET 的 32 位环境中开发的所有代码都可以在 64 位环境下运行。也就是说，用 SQL Server 2005 或 ASP.NET 开发的所有代码迁移到 64 位服务器上后，其运行不受影响。Microsoft 也对 CLR 做了许多改进，使.NET 的 64 位版本能正常工作。这些改进包括垃圾回收(处理更多的数据)、JIT 编译过程、异常处理等。

迁移到 64 位后，会增加一些强大的功能。最重要也是最明显的原因是 64 位服务器有更大的寻址空间。迁移到 64 位还可以有更大的基本数据类型。例如，2^32 的整数值是 4,294,967,296，而 2^64 的整数值是 18,446,744,073,709,551,616。应用程序将更便于计算 U.S.债务或其他很大的数字。

Microsoft 和 IBM 等公司给客户展示了 64 位的魅力。一个主要领域就是数据库和虚拟数据存储功能，这是迁移到 64 位的一个最有意义的理由。

Visual Studio 2005 可以在 64 位计算机上安装和运行。这个 IDE 有 32 位和 64 位编译器。其后果之一是，64 位的.NET Framework 只能用于 Windows Server 2003 SP1 或更高版本，以及将来的其他 64 位 Microsoft 操作系统。

在 Visual Studio 2005 中建立应用程序时，可以修改应用程序的建立属性，使之专门为 64 位计算机编译程序。要找到这个设置，需要打开应用程序的属性窗口，单击 Properties 页面上的 Build 选项卡。在 Build 页面上单击 Advanced 按钮，打开 Advanced Compiler Setting 对话框。在这个对话框中，可以修改底部的目标 CPU。这样，就可以使所建立的应用程序用于 Intel 64 位计算机或 AMD 64 位计算机，如图 0-1 所示。

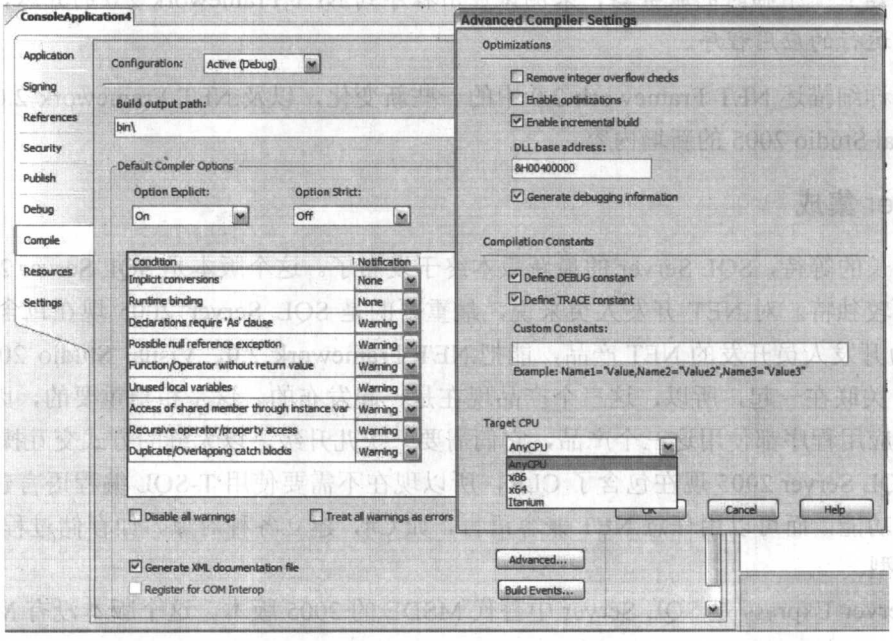

图 0-1　为 64 位计算机创建应用程序

泛型

为了使集合的功能更强大，也为了提高它们的效率和可用性，.NET Framework 2.0 引入了泛型。泛型引入到底层的架构上，这意味着 C#和 Visual Basic 2005 等语言现在可以建立使用泛型类型的应用程序了。泛型概念并不是新内容，它们类似于 C++的模板，但有些区别。其他语言中也有泛型，例如 Java。把它们引入.NET Framework 2.0 将给用户带来巨大的实惠。

泛型可以使一般的集合仍是强类型化的——出错的几率更小(因为错误在运行期间发生)，提高性能，在使用集合时将可以使用 Intellisense 特性。

要在代码中使用泛型，需要引用 System.Collections.Generic 命名空间，该命名空间还允许访问 Stack、Dictionary、SortedDictionary、List 和 Queue 类的泛型版本。下面演示了 Stack 类的泛型版本：

```
void Page_Load(object sender, EventArgs e)
{
    System.Collections.Generic.Stack<string> myStack =
        New System.Collections.Generic.Stack<string>();
    myStack.Push("St. Louis Rams");
    myStack.Push("Indianapolis Colts");
    myStack.Push("Minnesota Vikings");

    Array myArray;
    myArray = myStack.ToArray();

    foreach(string item in myArray)
    {
        Label1.Text += item + "<br />";
    }
}
```

在上面的例子中，Stack 类显式转换为 string 类型。这里用括号指定集合类型。这个例子使用 Stack<string>把 Stack 类转换为 string 类型。如果要把它转换为不是 string 类型的 Stack 类，例如 int，就可以指定 Stack<int>。

在创建 Stack 类时，Stack 类中的项集合就转换为指定的类型，所以 Stack 类不再把项集合转换为 object 类型，以后(在 foreach 循环中)再转换为 string 类型。这个过程称为装箱，该过程是很昂贵的。因为这段代码事先指定了类型，所以使用集合的性能提高了。

泛型除了用于处理集合类型之外，还可以用于处理类、委托、方法等。本书的第 10 章将详细介绍泛型。

匿名方法

匿名方法可以把编程步骤放在一个委托中，以后再执行该委托，而不是创建全新的方法。例如，如果不使用匿名方法，就要以下面的方式使用委托：

```
public partial class Default_aspx
```

```
{
    void Page_Load(object sender, EventArgs e)
    {
        this.Button1.Click += ButtonWork;
    }

    void ButtonWork(object sender, EventArgs e)
    {
        Label1.Text = "You clicked the button!";
    }
}
```

但若使用匿名方法，就可以把这些操作直接放在委托中，如下面的例子所示：

```
public partial class Default_aspx
{
    void Page_Load(object sender, EventArgs e)
    {
        this.Button1.Click += delegate(object myDelSender, EventArgs myDelEventArgs)
        {
            Label1.Text = "You clicked the button!";
        };
    }
}
```

在使用匿名方法时，不需要创建另一个方法，而是把需要的代码直接放在委托声明的后面。委托执行的语句和步骤放在花括号中，用一个分号结束。

可空类型

由于泛型引入到底层的.NET Framework 2.0 中，所以现在可以使用 System.Nullable<T>创建可空的值类型。这非常适合于创建 int 类型的可空项集合。而在此之前，很难创建包含空值的 int，或者把空值赋予 int。

要创建 int 类型的可空类型，可以使用下面的语法：

```
System.Nullable<int> x = new System.Nullable<int>;
```

有一个新的类型修饰符，可用于创建可空类型，如下面的例子所示：

```
int? salary = 800000
```

创建可空类型的功能不仅可在 C#中使用，因为该功能内置于.NET Framework，而这要归功于.NET 中的新泛型功能。所以，也可以在 Visual Basic 2005 中使用可空类型。

迭代器

迭代器允许在自己的定制类型中使用 foreach 循环。为此，需要使类实现 IEnumerable 接口，

如下所示：

```
using System;
using Systm.Collections;

public class myList
{
    internal object[] elements;
    internal int count;

    public IEnumerator GetEnumerator()
    {
        yield return "St. Louis Rams";
        yield return "Indianapolis Colts";
        yield return "Minnesota Vikiings";
    }
}
```

要使用 IEnumerator 接口，需要引用 System.Collections 命名空间。有了该命名空间，就可以迭代定制类了，如下所示：

```
void Page_Load(object sender, EventArgs e)
{
    myList IteratorList = new myList();

    foreach(string item in IteratorList)
    {
        Response.Write(item.ToString() + "<br />");
    }
}
```

部分类

部分类是.NET Framework 2.0 的一个新功能，C#也充分利用了这一功能。部分类可以把单个类分解到多个类文件中，以后在编译时再把这些文件合并起来。

要创建部分类，只需给要与另一个类合并起来的类使用 partial 关键字。在要与最初的类合并的类中，partial 关键字放在 class 关键字的前面。例如，有一个简单的类 Calculator，如下所示：

```
public class Calculator
{
    public int Add(int a, int b)
    {
        return a + b;
    }
}
```

接着，创建第二个类，它要与第一个类关联起来，如下面的例子所示：

```
public partial class Calculator
{
    public int Subtract(int a, int b)
    {
        return a - b;
    }
}
```

编译时，这些类会放在一个 Calculator 类实例中，就好像它们从一开始就放在一起一样。

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。在过去的十几年中，Microsoft 给 Windows 和 Windows API 添加了许多功能，Visual Baisc 2005 和 C++也经历了许多变化。虽然 Visual Baisc 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 6 及其早期版本来说，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 Visual Baisc 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++置根于 ANSI C++语言定义。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 C++开发人员说说字符串有多少种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 Visual Baisc 演变为 Visual Baisc.NET，再演变为 Visual Baisc 2005，保留了一些基本的 Visual Baisc 语法，但在设计上完全不同，从实际应用的角度来看，Visual Baisc 2005 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# 2005。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 Visual Baisc 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，Visual Baisc 2005 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 Framework 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。如果要运行本书中的例子，应安装.NET Framework 2.0，即使运行 Windows Server 2003，也要安装.NET Framework 2.0，因为.NET Framework 2.0 在这个服务器上并不是默认安装选项。

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 2005。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。

本书内容

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

第Ⅰ部分(第1~13章)——C#语言

本部分给出C#语言的背景知识。这一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#的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Ⅱ部分(第14~18章)——.NET环境

本部分介绍在.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Ⅲ部分(第19~22章)——数据

本部分介绍如何使用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、对Windows操作系统的支持，以及SQL Server 2005的.NET特性。

第Ⅳ部分(第23~25章)——Windows应用程序

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窗体是应用程序的胖客户版本，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Ⅴ部分(第26~27章)——Web应用程序

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 2.0提供的许多新特性。

第Ⅵ部分(第28~32章)——通信

这一部分介绍通信，主要论述独立于平台进行通信的Web服务、在.NET客户机和服务器之间通信的.NET Remoting技术、在后台上运行的Enterprise Services和DCOM通信。有了消息异步排队技术，可以进行断开连接的运行。“分布式编程的未来”一章可以让用户为当前的解决方案做出正确的通信选择。

第Ⅶ部分(第33章)——交互操作

COM的向后兼容性是.NET的一个重要组成部分，我们开发了太多的COM组件和应用程序。本部分将介绍如何在.NET应用程序中使用已有的COM组件，以及如何在COM应用程序中使用.NET组件。

第Ⅷ部分(第34~36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IX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

在读者学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 http://www.wrox.com/ 或 www.tupwk.com.cn/downpage 上下载。登录到站点http://www.wrox.com/，使用 Search 工具或使用书名列表就可以找到本书。接着单击本书细目页面上的 Download Code 链接，就可以获得所有的源代码。

注释：

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 0-7645-7534-1。

在下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入 http://www.wrox.com/dynamic/books/download.aspx 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但是错误总是难免的，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，我们将非常感激。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。

请给 fwkbook@tup.tsinghua.edu.cn 发电子邮件，我们就会检查您的信息，如果是正确的，我们将在本书的后续版本中采用。

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，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看到 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/booklist.shtml。

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Ⅰ部分　C# 语　言