



Core C# and .NET

C# 和 .NET 核心技术

- ◆ 涵盖 C# 2.0 和 .NET 2.0: 包括范型、母版页、DataGridview 和其他新特性
- ◆ 内容覆盖 Web 开发、Windows 开发、数据管理、安全性、线程、远程通信等等
- ◆ 提供了大量的代码示例, 可以用于解决实际问题

(美) Stephen C. Perry 著

肖斌 王小振 等译



机械工业出版社
China Machine Press

开发人员专业技术丛书

Core C# and .NET

C# 和 .NET 核心技术

(美) Stephen C. Perry 著
肖斌 王小振 等译

200609 23-1

本书在 C# 和 .NET 的基础上, 详细介绍了 C# 应用, 如处理文本、文件、数据库、XML、Windows 窗体和控件、打印、ASP .NET Web 应用、Web 服务以及远程机制等。还涉及异步、多线程、安全和部署等主题。本书每章末附有小结和习题, 可帮助读者巩固书中概念。

本书适合软件开发人员、高校相关专业师生学习和参考。

Simplified Chinese edition copyright © 2006 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *Core C# and .NET* (ISBN 0-13-147227-5) by Stephen C. Perry, Copyright ©2005.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Pearson Education, Inc..

本书封面贴有 Pearson Education (培生教育集团) 激光防伪标签, 无标签者不得销售。
版权所有, 侵权必究。

本书法律顾问 北京市展达律师事务所

本书版权登记号: 图字: 01-2006-1943

图书在版编目 (CIP) 数据

C# 和 .NET 核心技术 / (美) 佩里 (Perry, S.C.) 著; 肖斌等译. - 北京: 机械工业出版社, 2006.7

(开发人员专业技术丛书)

书名原文: Core C# and .NET

ISBN 7-111-19295-8

I .C… II .①佩… ②肖… III .C 语言 - 程序设计 IV .TP312

中国版本图书馆 CIP 数据核字 (2006) 第 059536 号

机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码 100037)

责任编辑: 王 雯 秦燕梅

北京中兴印刷有限公司印刷 · 新华书店北京发行所发行

2006 年 7 月第 1 版第 1 次印刷

186mm×240mm · 36.5 印张

定价: 69.00 元

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换
本社购书热线: (010) 68326294

序

学习一种新的程序语言、或者某种语言的新版本，这与去国外旅行非常相似：有些东西看起来很面熟，有些东西则显得非常新奇。如果只是注意那些“熟面孔”，你往往只会浮光掠影地大概看一下；但是，谁会满足于只是“走马观花”呢？通常情况下，正是某个国家、某种文化或者某种语言中的特异之处才让人觉得有意思，才能使人受益匪浅。

当然，要想做到这一点，还需要一个合适的导游或导游手册，这很重要。一本好的导游手册会告诉你看什么，什么时候看，吃什么，要避免什么，此外还会提供一些小技巧。好的编程书也是这样，它会告诉你使用什么框架和类，怎么调用方法，如何避免不好的实践做法，并提供一些有助于提高效率的技巧和最佳实践。

这些正是 Steve 将为你呈现的。

如果你用过 C# 1.0，或者有使用其他语言的背景，往往会忽略 2.0 的新特性。例如，我本人就是一个很有经验的开发人员，有自己的处理方式。如果你像我一样，那么你就可能仍然在按多年来一直采用的方式编写方法、循环和其他设计模式。虽然你也知道这可能得不到最高效的代码，但这样开发速度快、代码易于阅读和理解，而且这种方式确实可行。

Steve 写的这本书的确是一本“核心”C# 书。他首先介绍了 C# 的一些重要概念，并指出这些概念与 .NET 框架有什么关系。但与市面上大多数 C# 参考书不同的是，接下来他直接切入 C# 应用。这些应用涵盖了你要完成的大多数日常任务，如处理文本、文件、数据库、XML、Windows 窗体和控件、打印、ASP.NET Web 应用、Web 服务以及远程机制等。难得的是，Steve 还介绍了异步、多线程、安全和部署等主题。这本书如此全面，有了它，你就没有必要再买其他书了。

还等什么？该打破常规，换个新思路了！

Richard Hundhausen

《Introducing Microsoft Visual Studio 2005 Team System》作者

前 言

“为计算机准备程序的过程非常吸引人，因为它不仅能带来经济和科学技术方面的回报，而且能给人以美的享受，就像写诗或作曲一样。”

——Donald Knuth, 《Fundamental Algorithms》前言 (1968)

37年后，程序员仍能体验到开发一个高效程序所带来的创造满足感。这可能是半夜里突然闯入脑海中的10行递归代码，也可能是一个完整的产品管理系统，这个系统的设计也许困扰了你整整一年。所以，即便是现在，好的程序仍是在传达一种逻辑和自然的感觉，特别是把这种感觉传达给它们的用户。

不过如今还要面对更大的挑战。现在的软件要求有更强的扩展性——可以运行在LAN、Internet或蜂窝电话上。因为代码可以在世界范围内访问，所以安全问题也就更为重要。而这又带来了另外一些问题，如可扩展性问题，以及如何为数以百计的并发用户实现代码同步。随着用户的增多，也带来了更多的文化元素，这种多元文化的并存还意味着需要定制程序，以适应世界范围内不同客户群体各自的语言和文化特性。

与其他统一开发环境一样，.NET 以及为其编写的语言可以解决这些难题。本书适用于使用 .NET 框架的开发人员、软件架构师和学生。虽然本书中只有一章专门介绍 C# 语言的语法结构，但书中所有代码均使用 C# 编写。

首先，本书不是程序设计的入门读物，在此假设你对计算机语言已经有丰富的经验；其次，虽然书中通过大量的例子强调了封装、多态和继承等原理，但本书也并非面向对象编程（Object-Oriented Programming, OOP）的入门读物；最后，本书也不是介绍使用 Visual Studio .NET 来开发 C# 程序的入门图书，尽管书中提到了 VS .NET，但我们的重点是开发和理解 C# 及 .NET 类，而且不依赖于任何 IDE（集成开发环境）。

本书面向的是那些有经验的程序员，他们目前正转向 .NET，并且想对 .NET 的功能有个全面了解。你可能用过 VB6 或 C++，现在想了解 .NET；也可能是一个 VB.NET 程序员，想扩展知识面，进一步掌握 C#；或者（确实有这种可能）你是一个 Java 程序员，想对从未涉足的 .NET 一探究竟。下面是本书的主要内容，你可以看看这本书能不能满足你的需要。

- 18 章内容。最好按顺序阅读前 4 章，其中对 C# 做了介绍，你会对如何使用 .NET 类库有所熟悉，其他章则可以根据你的兴趣有选择地阅读。第 6 章和第 7 章介绍了如何开发 Windows 窗体程序；第 8 章和第 9 章介绍了 .NET 图形类 GDI+；第 10 章~第 12 章讨论了如何处理数据，包括 XML 和 ADO.NET；第 13 章~第 15 章分别介绍了一些更高级的主题：线程、远程机制和代码安全；最后是 Web 三部曲：第 16 章讨论了开发 ASP.NET Web 页面；第 17 章深入分析了如何管理状态信息和 HTTP 请求；最后第 18 章介绍了 Web 服务。
- .NET 2.0。本书是 2.0 Beta 版发布后出版的，所以书中内容以该版本为基础。2.0 的有关内容不是单独作为一部分来介绍，而是融入各章之中。不过，为了方便读者，附录 A 给出了 .NET 2.0 有关内容的总结和单独的索引。
- 示例代码。本书大多数的示例代码都是很短的片断，只强调某个结构或技术，当然，这也是为了避免无用的代码浪费纸张。只有在确实必要的情况下才会出现超过一页的代码示例。书中所有重要的示例代码都可以从网上下载，下载地址是 www.corecsharp.net，或者访问 www.phptr.com/ti

tle/0131472275 也可以下载,要进入下载区,请输入关键词 parsifal。

- 练习题与答案。每章最后都有一些练习题,可以检查你对这一章知识的掌握程度。本书最后专门有一部分给出了这些习题的答案。
- 重事实而不是个人看法。这本书并不是我的个人看法,而是根据 .NET 和 C# 固有的特性编写的。另外还提供了核心推荐和注解,也是着力强调知识本身,而不是我的主观见解。

可能有些人不同意下面的观点,但如果你真的想要学习 C# 和 .NET,请关掉你的 IDE,打开最惯用的文本编辑器,学习如何采用命令行方式使用 C# 编译器。等掌握了基础知识之后,可以再选择使用 VS.NET 或其他 IDE 来提高编程效率。

最后,再简单提一下 .NET 和 Microsoft。本书使用的是 Microsoft .NET 1.x 和 Whidbey 测试版,包含了 Microsoft 独有的 ADO.NET 和 ASP.NET 等内容,事实上,Microsoft 已经对这些技术申请了专利。不过,所有 C# 和许多 .NET 基类库都基于同一个标准,以便把它们移植到其他平台。所以,现在和不久的将来,本书介绍的许多技术也将能应用于非 Windows 平台上的 .NET 实现(如 Mono 项目, <http://www.mono-project.com/Main-Page>)。

致谢

为了完成这本书,我做了 21 个月的研究和开发,在此期间得到了许多人的帮助。首先,我要感谢我的妻子 Rebecca,她不知疲倦地逐页阅读潦草的手稿,并利用她的系统编程背景提出了很有价值的建议。其次,我要感谢那些评论人员,正是他们提出的批评和建议,才使本书组织得更好,内容和代码中的错误更少,而且该强调的内容更突出,他们是:Greg Beamer、James Edelen、Doug Holland、Curtiss Howard、Anand Narayanaswamy 和 Gordon Weakliem。特别要感谢 Richard Hundhausen 提出的建议,真的非常棒,完全超乎预料;感谢 Cay Horstmann 阅读了每一章手稿,他对 Java 的执著使他成为一个地地道道的“疯狂鼓吹者”。还要感谢 Alan Tharp 博士对于我计划编写一本 .NET 书所给予的鼓励,他是最尊敬的计算机专业顾问。

最后,我非常高兴与 Prentice Hall PTR 出版社的编辑、员工们一起工作,特别要感谢 Stephane Nakib、Joan Murray、Ebony Haight、Jessica D'Amico、Kelli Brooks 和 Vanessa Moore 的辛勤工作。如果没有第一位编辑 Stephane Nakib 的努力,本书不可能面世,她提出了编写本书的想法,并在最初阶段着力将工作向前推进。Joan Murray 编辑在中期介入,一直在仔细监督,并提供建议和鼓励,直到最终完成本书。制作编辑 Vanessa Moore 和文字编辑 Kelli Brooks 完成了“最麻烦的工作”,尽管最终手稿中不一致的地方和错误的词句层出不穷,但经他们之手,终于变成了一本可以出版的图书。对他们的感谢无以言表。写这本书时,有时我会感到遥遥无期,甚至这种感觉会超越写作带来的成就感,在那些日子里,有这样一些专业人士在身边一起工作实在是意义非凡。

目 录

序 前言

第一部分 C# 编程基础和 .NET 概述

第 1 章 .NET 和 C# 介绍	1
1.1 .NET 框架概述	1
1.2 通用语言运行时 CLR	3
1.2.1 编译 .NET 代码	3
1.2.2 通用类型系统	5
1.2.3 程序集	6
1.3 框架类库	9
1.4 使用 .NET 框架和 SDK	10
1.4.1 更新 .NET 框架	11
1.4.2 .NET 框架工具	11
1.4.3 Ildasm.exe	12
1.4.4 Ildasm 和模糊技术	14
1.4.5 wincv.exe	14
1.4.6 框架配置工具	15
1.5 了解 C# 编译器	16
1.5.1 找到编译器	17
1.5.2 从命令行编译	17
1.6 小结	19
1.7 习题	19
第 2 章 C# 语言基础	20
2.1 C# 程序的布局组成	20
2.2 基本类型	23
2.2.1 decimal	25
2.2.2 bool	25
2.2.3 char	25
2.2.4 byte 和 sbyte	25
2.2.5 short、int 和 long	26
2.2.6 single 和 double	26
2.2.7 使用 Parse 和 TryParse 转换数字 字符串	26
2.3 算术操作符、逻辑操作符和条件	

操作符	27
2.3.1 算术操作符	27
2.3.2 条件和关系操作符	27
2.3.3 流程控制语句	28
2.3.4 if-else 语句	29
2.3.5 switch 语句	29
2.4 循环	30
2.4.1 while 循环	30
2.4.2 do 循环	31
2.4.3 for 循环	31
2.4.4 foreach 循环	32
2.4.5 循环中的跳转控制	32
2.5 C# 预处理指令	33
2.5.1 条件编译	33
2.5.2 诊断指令	34
2.5.3 代码域	34
2.6 字符串	35
2.6.1 字符串直接量	35
2.6.2 字符串操作	35
2.7 枚举类型	37
2.7.1 使用枚举	38
2.7.2 System.Enum 的方法	39
2.7.3 枚举和位标志	39
2.8 数组	39
2.8.1 声明和创建数组	40
2.8.2 使用 System.Array 的方法和成员 属性	40
2.9 引用类型和值类型	42
2.9.1 System.Object 和 System.ValueType	43
2.9.2 引用类型和值类型的内存分配	43
2.9.3 装箱	43
2.9.4 值类型和引用类型差别小结	44
2.10 小结	45
2.11 习题	45
第 3 章 C# 类设计	46
3.1 C# 类简介	46

3.2 定义类	47	4.2.3 例: 处理通用 SystemException 异常	89
3.2.1 属性	47	4.2.4 如何创建定制异常类	90
3.2.2 访问修饰符	48	4.2.5 未处理异常	92
3.2.3 Abstract、Sealed 和 Static 修饰符	49	4.2.6 异常处理原则	93
3.2.4 类标识符	49	4.3 定制类中实现 System.Object 方法	93
3.2.5 基类、接口和继承	49	4.3.1 用 ToString() 方法描述对象	94
3.3 类成员概述	50	4.3.2 比较对象的 Equals() 方法	95
3.4 常量、字段和成员属性	50	4.3.3 克隆来创建对象副本	97
3.4.1 常量	51	4.4 使用 .NET 集合类和接口	98
3.4.2 字段	51	4.4.1 集合接口	98
3.4.3 成员属性	53	4.4.2 System.Collections 命名空间	104
3.4.4 索引器	54	4.4.3 栈和队列	105
3.5 方法	56	4.4.4 ArrayList	106
3.5.1 方法修饰符	56	4.4.5 Hashtable	107
3.5.2 传递参数	59	4.4.6 System.Collections.Generic 命名 空间	109
3.6 构造函数	61	4.5 对象串行化	111
3.6.1 实例构造函数	61	4.6 对象生命周期管理	113
3.6.2 私有构造函数	63	4.7 小结	117
3.6.3 静态构造函数	64	4.8 习题	118
3.7 委托和事件	65		
3.7.1 委托	65	第二部分 使用 .NET 框架类库创建应用	
3.7.2 基于委托的事件处理	66	第 5 章 C# 文本处理和文件 I/O	119
3.8 操作符重载	72	5.1 字符和 Unicode	119
3.9 接口	74	5.1.1 Unicode	120
3.9.1 创建和使用定制接口	74	5.1.2 使用字符	120
3.9.2 使用接口	76	5.2 String 类	123
3.10 范型	77	5.2.1 创建字符串	123
3.11 结构	78	5.2.2 字符串操作概述	124
3.11.1 定义结构	79	5.3 比较字符串	125
3.11.2 使用结构的方法和成员属性	80	5.3.1 使用 String.Compare	125
3.12 结构和类的比较	80	5.3.2 使用 String.CompareOrdinal	126
3.12.1 结构是值类型, 类是引用类型	81	5.4 搜索、修改和编码字符串内容	127
3.12.2 与类不同, 结构不能被继承	81	5.4.1 搜索字符串内容	127
3.12.3 选择结构或类的一般规则	81	5.4.2 搜索包含代理的字符串	127
3.13 小结	82	5.4.3 字符串转换	128
3.14 习题	82	5.4.4 字符串编码	129
第 4 章 使用 C# 对象	84	5.5 StringBuilder	130
4.1 对象创建	84	5.5.1 StringBuilder 类概述	131
4.2 异常处理	86	5.5.2 StringBuilder 与字符串连接的比较	131
4.2.1 System.Exception 类	87		
4.2.2 编写异常处理代码	87		

5.6 格式化数字和日期值	132	6.4 使用菜单	183
5.6.1 构造格式元素	132	6.4.1 MenuItem 属性	183
5.6.2 格式化数字值	133	6.4.2 上下文菜单	183
5.6.3 格式化日期和时间	134	6.5 为窗体增加帮助	185
5.7 正则表达式	137	6.5.1 ToolTips	185
5.7.1 Regex 类	137	6.5.2 响应 F1 和帮助按钮	186
5.7.2 创建正则表达式	140	6.5.3 HelpProvider 组件	187
5.7.3 模式匹配示例	141	6.6 窗体继承	188
5.7.4 使用组	143	6.6.1 建立和使用窗体库	188
5.7.5 示例: 使用正则表达式	144	6.6.2 使用继承窗体	188
5.8 System.IO: 读写数据流的类	145	6.7 小结	189
5.8.1 Stream 类	145	6.8 习题	189
5.8.2 FileStream	146	第 7 章 Windows 窗体控件	190
5.8.3 MemoryStream	147	7.1 .NET Windows 窗体控件概述	190
5.8.4 BufferedStream	147	7.2 Button 类、GroupBox、Panel 和 Label	192
5.8.5 用 StreamReader 和 StreamWriter 读写文本行	148	7.2.1 Button 类	192
5.8.6 StringWriter 和 StringReader	150	7.2.2 CheckBox 类	193
5.8.7 使用 CryptoStream 类加密	150	7.2.3 RadioButton 类	193
5.9 System.IO: 目录和文件	152	7.2.4 GroupBox 类	195
5.9.1 FileSystemInfo	152	7.2.5 Panel 类	195
5.9.2 使用 DirectoryInfo、Directory 和 Path 类处理目录	153	7.2.6 Label 类	196
5.9.3 使用 FileInfo 和 File 类处理文件	156	7.3 PictureBox 和 TextBox 控件	197
5.10 小结	157	7.3.1 PictureBox 类	197
5.11 习题	157	7.3.2 TextBox 类	199
第 6 章 建立 Windows 窗体应用	159	7.4 ListBox、CheckedListBox 和 ComboBox 类	200
6.1 Windows 窗体编程	160	7.4.1 ListBox 类	200
6.2 Windows.Forms 控件类	162	7.4.2 其他列表控件: ComboBox 和 CheckedListBox	203
6.2.1 Control 类	162	7.5 ListView 和 TreeView 类	204
6.2.2 使用控件	163	7.5.1 ListView 类	204
6.2.3 控件事件	166	7.5.2 TreeView 类	208
6.3 Form 类	170	7.6 ProgressBar、Timer 和 StatusStrip 类	212
6.3.1 设置窗体外观	170	7.7 建立定制控件	214
6.3.2 设置窗体位置和大小	173	7.7.1 扩展控件	214
6.3.3 显示窗体	174	7.7.2 建立定制用户控件	214
6.3.4 非模式窗体的生命周期	174	7.7.3 用户控件示例	214
6.3.5 窗体交互——示例应用	175	7.7.4 使用定制用户控件	216
6.3.6 属主和从属窗体	178	7.7.5 设计时使用用户控件	217
6.3.7 消息框和对话框	178	7.8 拖放控件	217
6.3.8 多文档界面窗体	180		

7.9 使用资源	220	9.3.8 报表示例	269
7.9.1 使用资源文件	221	9.3.9 创建定制 PrintDocument 类	271
7.9.2 用资源文件创建本地化窗体	223	9.4 小结	273
7.10 小结	225	9.5 习题	273
7.11 习题	225	第 10 章 在 .NET 中使用 XML	275
第 8 章 使用 GDI+ 的 .NET 图形	226	10.1 使用 XML	275
8.1 GDI+ 概述	226	10.1.1 使用 XML 串行化创建 XML 数据	276
8.1.1 Graphics 类	227	10.1.2 XML 模式定义(XSD)	278
8.1.2 Paint 事件	229	10.1.3 使用 XML 样式表	280
8.2 使用 Graphics 对象	231	10.2 XML 数据读取技术	281
8.2.1 基本二维图形	231	10.2.1 XmlReader 类	282
8.2.2 Pen	234	10.2.2 XmlNodeReader 类	285
8.2.3 Brush	236	10.2.3 XmlReaderSettings 类	286
8.2.4 Color	239	10.2.4 用 XML 模式验证 XML 数据	287
8.2.5 示例: 建立颜色浏览器	240	10.2.5 读取 XML 数据的其他方法	287
8.3 图像	243	10.3 XML 数据写出技术	288
8.3.1 加载和存储图像	244	10.4 用 XPath 搜索 XML	290
8.3.2 处理图像	246	10.4.1 构造 XPath 查询	290
8.3.3 示例: 使用图像	247	10.4.2 XmlDocument 和 XPath	292
8.3.4 Microsoft Windows 平台 GDI 和 BitBlt 的有关提示	252	10.4.3 XPathDocument 和 XPath	292
8.4 小结	253	10.4.4 XmlDataDocument 和 XPath	293
8.5 习题	253	10.5 小结	295
第 9 章 字体、文本和打印	255	10.6 习题	295
9.1 字体	255	第 11 章 ADO.NET	296
9.1.1 字体族	256	11.1 ADO.NET 体系结构概述	296
9.1.2 Font 类	256	11.1.1 .NET 中的 OLE DB 数据 提供者	297
9.2 绘制文本串	258	11.1.2 .NET 数据提供者	297
9.2.1 绘制多行文本	259	11.2 数据访问模型: 连接模型和无连接 模型	299
9.2.2 用 StringFormat 类格式化 字符串	259	11.2.1 连接模型	299
9.2.3 使用制表位	260	11.2.2 无连接模型	300
9.2.4 字符串截断、对齐和自动换行	261	11.3 ADO.NET 连接模型	301
9.3 打印	262	11.3.1 连接类	301
9.3.1 概述	262	11.3.2 命令对象	304
9.3.2 PrintDocument 类	263	11.3.3 DataReader 对象	307
9.3.3 打印机设置	264	11.4 数据集、数据表和无连接模型	308
9.3.4 页面设置	265	11.4.1 DataSet 类	309
9.3.5 PrintDocument 事件	266	11.4.2 数据表	309
9.3.6 PrintPage 事件	267	11.4.3 向 DataSet 加载数据	312
9.3.7 预览打印报表	268		

11.4.4 定义 DataSet 中表之间的关系	316	13.3.4 定时器	368
11.4.5 连接模型和无连接模型的选择	318	13.4 线程同步	369
11.5 XML 和 ADO.NET	318	13.4.1 Synchronization 属性	371
11.5.1 用 DataSet 创建 XML 数据和模式文件	318	13.4.2 Monitor 类	371
11.5.2 从 XML 创建 DataSet 模式	320	13.4.3 Mutex	372
11.5.3 将 XML 数据读入 DataSet	320	13.4.4 Semaphore	373
11.6 小结	322	13.4.5 避免死锁	374
11.7 习题	322	13.4.6 同步技术小结	375
第 12 章 Windows 窗体控件的数据绑定	324	13.5 小结	375
12.1 数据绑定概述	324	13.6 习题	376
12.1.1 简单数据绑定	325	第 14 章 利用远程通信创建分布式应用	378
12.1.2 与列表控件的复杂绑定	326	14.1 应用域	378
12.1.3 单向和双向数据绑定	327	14.1.1 应用域的优点	378
12.1.4 使用绑定管理器	328	14.1.2 应用域和程序集	379
12.2 应用中使用简单和复杂数据绑定	330	14.1.3 使用 AppDomain 类	379
12.2.1 绑定到 DataTable	330	14.2 远程通信	381
12.2.2 控件绑定到 ArrayList	332	14.2.1 远程通信体系结构	382
12.2.3 向数据源添加元素	334	14.2.2 远程通信类型	384
12.2.4 验证更新	335	14.2.3 客户端激活对象	386
12.2.5 更新原始数据库	335	14.2.4 服务器激活对象	386
12.3 DataGridView 类	336	14.2.5 类型注册	387
12.3.1 成员属性	336	14.2.6 与服务器激活对象 SAO 的远程通信	389
12.3.2 事件	341	14.2.7 与客户端激活对象 CAO 的远程通信	395
12.3.3 建立主表-明细表 DataGridView	344	14.2.8 分布式应用的设计原则	399
12.3.4 虚拟模式	346	14.3 租约和缔约	400
12.4 小结	349	14.3.1 租约	400
12.5 习题	350	14.3.2 缔约	402
第三部分 C# 和 .NET 框架的高级使用		14.4 小结	404
第 13 章 异步编程和多线程	351	14.5 习题	404
13.1 什么是线程	351	第 15 章 代码优化、安全与部署	406
13.2 异步编程	353	15.1 遵循 .NET 代码设计准则	407
13.2.1 异步委托	354	15.2 强命名程序集	409
13.2.2 实现异步调用的示例	356	15.2.1 创建强命名程序集	410
13.3 直接使用线程	362	15.2.2 延迟签名	411
13.3.1 创建和使用线程	362	15.2.3 全局程序集缓存(GAC)	411
13.3.2 多线程实际应用	365	15.2.4 版本	412
13.3.3 使用线程池	367	15.3 安全	412

15.3.1	权限和权限集	413
15.3.2	证据	416
15.3.3	安全策略	418
15.3.4	配置安全策略	419
15.3.5	.NET 框架配置工具	420
15.3.6	示例: 使用配置工具配置代码 访问安全	422
15.3.7	为程序集请求权限	424
15.3.8	程序式安全	426
15.4	部署应用的有关考虑	431
15.4.1	Microsoft Windows 部署: XCOPY 部署与 Windows Installer	431
15.4.2	程序集部署到 GAC 中	431
15.4.3	部署私有程序集	432
15.4.4	使用 CodeBase 配置	432
15.4.5	使用配置文件管理程序集的多个 版本	433
15.4.6	程序集版本和产品信息	433
15.5	小结	434
15.6	习题	434

第四部分 Internet 编程

第 16 章 ASP .NET Web 表单和控件

16.1	Internet 的客户端 - 服务器交互	438
16.1.1	Web 应用示例: 实现 BMI 计算器	438
16.1.2	使用 ASP.NET 实现 BMI 计算器	441
16.1.3	内联代码模型	442
16.1.4	代码隐藏模型	447
16.1.5	部分类代码隐藏	449
16.1.6	Page 类	450
16.2	Web 表单控件	452
16.2.1	Web 控件概述	453
16.2.2	指定 Web 控件的外观	454
16.2.3	简单控件	454
16.2.4	列表控件	457
16.2.5	DataList 控件	459
16.3	数据绑定和数据源控件	461
16.3.1	绑定到 DataReader	461
16.3.2	绑定到 DataSet	463

16.3.3	数据源控件	464
16.4	验证控件	469
16.5	母版页和内容页	472
16.5.1	创建母版页	473
16.5.2	创建内容页	474
16.5.3	从内容页访问母版页	475
16.6	建立和使用定制 Web 控件	475
16.6.1	定制控件示例	476
16.6.2	使用定制控件	477
16.6.3	控件状态管理	478
16.6.4	复合控件	479
16.7	选择 Web 控件显示数据	481
16.8	小结	481
16.9	习题	481
第 17 章	ASP .NET 应用环境	483
17.1	HTTP 请求和响应类	483
17.1.1	HttpRequest 对象	484
17.1.2	HttpResponse 对象	486
17.2	ASP.NET 和配置文件	489
17.2.1	Web.config 剖析	490
17.2.2	增加定制配置段	493
17.3	ASP.NET 应用安全	495
17.3.1	表单认证	495
17.3.2	表单认证示例	497
17.4	状态维护	500
17.4.1	应用状态	501
17.4.2	会话状态	502
17.5	缓存	504
17.5.1	页面输出缓存	504
17.5.2	数据缓存	506
17.6	用 WebRequest 和 WebResponse 创建 Web 客户	508
17.6.1	WebRequest 和 WebResponse 类	508
17.6.2	Web 客户示例	509
17.7	HTTP 管线	510
17.7.1	管线中处理请求	510
17.7.2	HttpApplication 类	512
17.7.3	HTTP 模块	514
17.7.4	HTTP 处理器	517
17.8	小结	520
17.9	习题	520

第 18 章 XML Web 服务	521	18.5 在 Web 服务中使用复杂数据类型	544
18.1 Web 服务简介	521	18.5.1 返回图像的 Web 服务	544
18.1.1 发现和使用 Web 服务	522	18.5.2 使用 Amazon Web 服务	545
18.1.2 UDDI 简介	522	18.5.3 为 Amazon Web 服务创建代理	547
18.2 构建 XML Web 服务	524	18.5.4 构建 WinForms Web 服务客户	548
18.2.1 手动创建 Web 服务	525	18.6 Web 服务性能	550
18.2.2 用 VS.NET 创建 Web 服务	527	18.6.1 配置 HTTP 连接	550
18.2.3 用 WebService 和 WebMethod 属性 扩展 Web 服务	528	18.6.2 处理大量数据	551
18.3 构建 XML Web 服务客户	530	18.7 小结	551
18.3.1 创建简单的客户来访问 Web 服务类	530	18.8 习题	551
18.3.2 用 Visual Studio.NET 创建代理	536		
18.4 理解 WSDL 和 SOAP	537		
18.4.1 WSDL	537		
18.4.2 SOAP	539		

附 录

附录 A .NET 2.0 和 C# 2.0 的特定 特性	553
附录 B DataGridView 事件和委托	555
习题答案	563

第一部分 C# 编程基础和 .NET 概述

第 1 章 .NET 和 C# 介绍

本章主要内容:

- .NET 框架概述: 介绍 .NET 框架的体系结构和特性。
- 通用语言运行时(Common Language Runtime, CLR): 概要介绍 .NET 框架运行时部分完成的任务: 即时(Just-in-Time, JIT)编译器、加载程序集和代码验证。
- 通用类型系统(Common Type System, CTS)和通用语言规范(Common Language Specification, CLS): 有关 CLR 兼容性和语言互操作性的规则。
- 程序集(Assembly): 分析程序集的结构及其底层原理, 并介绍私有程序集和共享程序集之间的区别。
- 框架类库/Framework Class Library, FCL: 框架库提供了数百个基类, 这些基类划分为多个逻辑命名空间。
- 开发工具: .NET 提供了一些辅助代码开发的工具, 包括代码反汇编工具 Ildasm、浏览类属性的 WinCV 和框架配置工具等。
- 编译和运行 C# 程序: 通过命令行方式, 使用 C# 编译器和选项构建应用。

要想有效地使用一门语言, 不光是要学习这门语言的语法和特性。实际上, 在学习新技术时, 对编程环境的学习往往更费精力。对于优秀的开发人员和软件架构师来说, 仅精通 C# 语言是不够的, 还必须通晓底层类库, 并且熟知用来搜索这些类库、调试代码以及检查底层代码效率的工具。

本章的主要目的在于, 在继续学习 C# 语言的语法和语义之前, 先让你对 .NET 环境有所了解。这一章的重点是: 环境(而不是编程语言)如何影响软件开发。对于 .NET 新手而言, 需要掌握一些新思想。.NET 改变了处理遗留代码和版本控制的方式; 释放程序资源的方式也有所不同; 它允许采用一种语言开发代码, 而利用另一种语言使用这些代码; 由于不再依赖系统注册表, 这就简化了代码的部署; 另外, .NET 创建了一种自描述元语言(self-describing metalanguage), 可以用来在运行时确定程序逻辑。所有这些特性在软件开发过程的某个阶段都会接触到, 它们会影响应用的设计和部署。

在程序员看来, .NET 平台由一个运行时环境和一个基类库组成。本章正是从这个角度来组织的。首先分别介绍了通用语言运行时 CLR 和框架类库 FCL, 然后指出开发人员可以用哪些基本工具深入了解 .NET 框架的内部工作原理, 以及哪些工具有助于应用的管理和发布。为了更自然地引入第 2 章, 本章最后一节将通过实例来介绍 C# 编译器。

1.1 .NET 框架概述

.NET 框架设计为一个集成环境, 可以在 Internet、桌面(如 Windows 窗体), 甚至移动设备(使用精简框架 Compact Framework)上无缝地开发和运行应用。其主要目标是:

- 提供一个覆盖整个应用范围的、一致的面向对象环境；
- 提供一个环境，将困扰 Windows (COM) 程序员的版本冲突 (“DLL Hell”，即 DLL 地狱) 问题最小化，简化代码的发布/安装过程；
- 基于公认的标准，提供一个可以在任意操作系统上运行的可移植环境。实际上，C# 和 .NET 运行时的一个主要部分，即通用语言基础设施 (Common Language Infrastructure, CLI)，已经得到了 ECMA^① 的标准化。
- 提供一个可管理的环境，在这个环境中，可以很容易地验证代码，以保证程序安全运行。

为了实现上述目标，.NET 框架设计者们最后确定了以下体系结构，将框架分解为两部分：通用语言运行时 CLR 和框架类库 FCL，其结构如图 1-1 所示。

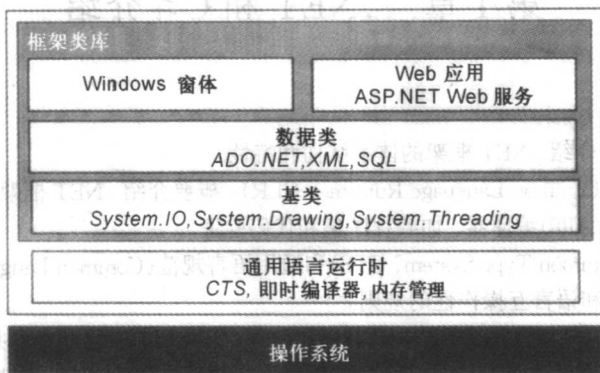


图 1-1 .NET 框架

CLR 是 Microsoft 对 CLI 标准的具体实现，它处理代码执行及所有相关任务：编译、内存管理、安全、线程管理、强制类型安全和类型使用。在 CLR 中运行的代码称为托管代码 (Managed Code)，以区别于不在 CLR 中运行的非托管代码 (unmanaged code)，如基于 COM 或 Windows API 的组件。

.NET 的另一个主要部分是框架类库 FCL，对于在 .NET 中运行的应用来说，它是一个可重用的类型 (类、结构等) 代码库。正如图中所示，它包含了涉及数据库访问、图形、与非托管代码互操作、安全、Web 和 Windows 窗体等类。只要是遵循 .NET 框架的语言，都会使用这个公共类库。因此，只要知道了如何使用这些类型，不论你选择用哪一种 .NET 语言编写程序，这些知识都可以用上。

Microsoft .NET 和 CLI 标准

如果开发人员下决心花时间来学习 C# 和 .NET，很自然地会想到，能否将获得的知识应用于其他平台上。更明确地说，Microsoft 的 .NET 产品是否仅限于 Windows 操作系统？或者，它是不是一个可移植的运行时代和开发平台，可以在多个操作系统上实现？要回答这个问题，有必要先了解 Microsoft .NET、C# 和 CLI 标准之间的关系。

CLI 定义了一个与平台无关的虚拟代码执行环境。由于未指定任何操作系统，所以操作系统可以是 Windows，更可以是 Linux。该标准的核心是定义了一个通用中间语言 (Common Intermediate Language, CIL) 和一个类型系统，遵循 CLI 的编译器必须生成 CIL，而类型系统则定义了遵循 CLI 的所有语言都支持

① ECMA 国际 (ECMA International) 全名是欧洲计算机制造协会 (European Computer Manufacturers Association)，简写作 ECMA。

的数据类型。下一节将会讲到,这种中间代码将编译为其主机操作系统的本地语言。

CLI 还包含了由 Microsoft 开发并大力推行的 C# 语言的标准,因此,C# 是 .NET 事实上的标准语言。但后来,其他厂商也很快采纳了 CLI 标准,并开发了诸多语言,如 Python、Pascal、Fortran、Cobol 和 Eiffel .NET 编译器。

图 1-1 所示的 .NET 框架是 Microsoft 对 CLI 标准的实现。需要注意的最重要的一点是,这个实现中包含的大量特性并不是 CLI 架构所要求的。为了说明这一点,图 1-2 给出了 CLI 标准架构,你可以与图 1-1 做一个比较。

概括起来,CLI 定义了两个实现:一个是最小实现,称为内核概要(Kernel Profile),另一个提供了更多特性,称为精简概要(Compact Profile)。内核概要包含遵循 CLI 的编译器所需要的类型和类,其中基类库包括基本的数据类型类,还包括提供简单文件访问、定义安全属性以及实现一维数组的其他类。精简概要添加了 3 个类库:定义简单 XML 解析的 XML 库、提供

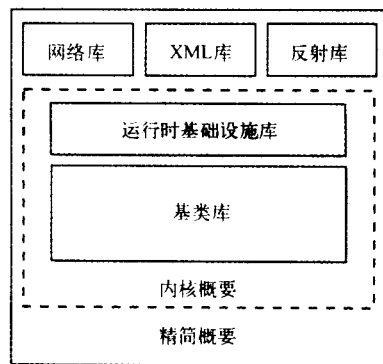


图 1-2 CLI 规范定义的架构

供 HTTP 支持和端口访问的网络库,以及支持反射(程序通过元代码实现自检的一种方法)的反射库。

本书介绍的是 Microsoft 的 CLI 实现,但如果只介绍 CLI 规范中定义的内容,那么本书的篇幅可能就小多了。倘若如此,我们就不会特别加入章节来介绍 ADO.NET(数据库类)、ASP.NET(Web 类)或 Windows 窗体等内容,而有关 XML 的各章也将大幅削减。你应该会想到,这些库的功能依赖于底层 Windows API。另外,.NET 允许程序使用一种互操作(Interop)特性来调用 Win32 API,也就是说,.NET 开发人员不仅能访问 Win32 API,还能访问遗留应用和组件(COM)。

由于如此倚重于 Windows,Microsoft 的 .NET 实现更称得上是一个透明环境,而不只是一个虚拟环境,这并没有什么不好。对于转向 .NET 的开发人员来说,利用 Microsoft 的 .NET 实现,能够充分结合 .NET 组件和原来已有的代码来创建混合应用。也就是说,无需将 .NET 实现代码移植到其他操作系统。

CLI 开源组织正在着力采纳 Microsoft 增加的这些特性,对开发人员(以及本书读者)来说,这自然是一个好消息。作为 CLI 主要项目之一的 Mono^①,就已经包含了诸如 ADO.NET、Windows 窗体、全部 XML 类,以及大量集合(Collection)类等主要特性。这确实非常重要,因为这意味着,如果你在使用 Microsoft .NET 的过程中积累了一些知识和技能,那么这些知识对于 Linux、BSD、Solaris 平台上的 .NET 实现也同样适用。了解了这一点之后,下面就对 Microsoft CLI 实现做一个概要介绍。

1.2 通用语言运行时 CLR

通用语言运行时 CLR 要管理应用的整个生命周期:查找代码、编译、加载相关的类、管理其执行,并确保自动内存管理。此外,它还支持跨语言集成,允许不同语言生成的代码能够无缝地交互。这一节将简单介绍 CLR 的内部原理,看看它是如何实现这些功能的。尽管不是深入的讨论,但通过本书的介绍,你会熟悉有关的术语,了解独立于语言的体系结构,并理解创建和执行程序时到底发生了什么。

1.2.1 编译 .NET 代码

遵循 CLR 的编译器生成的代码所面向的是运行时系统,而不是在某个特定的 CPU 上执行。这种代码也称为通用中间语言 CIL(Common Intermediate Language)、中间语言 IL(Intermediate Language)或 Microsoft

① 见 http://www.mono-project.com/Main_Page。

中间语言 MSIL(Microsoft Intermediate Language),这是一种汇编型语言,会打包为 EXE 或 DLL 文件。需要注意的是,这些文件并不是标准的可执行文件,当程序实际运行时,还需要运行时系统的 JIT 编译器将文件中的 IL 转换成机器码。因为要由通用语言运行时 CLR 负责管理这些 IL,所以这种代码称为托管代码(managed code)。

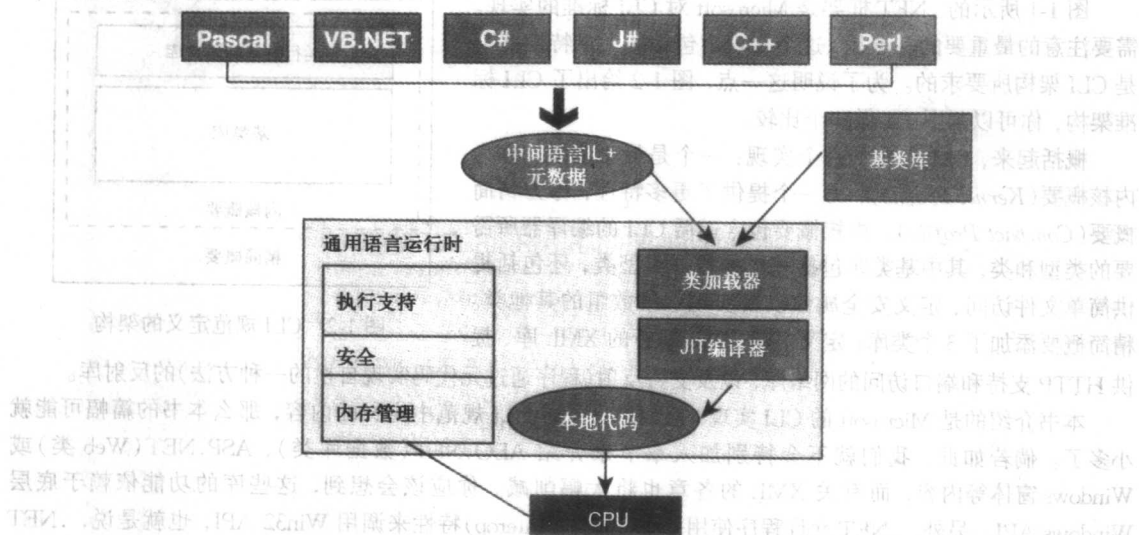


图 1-3 CLR 功能

使用中间代码是实现 .NET 框架语言兼容性目标的关键之一,如图 1-3 所示,CLR 不知道(也无需知道)应用是由哪种语言创建的。它只与独立于语言的 IL 交互。因为应用通过 IL 进行通信,所以一种编译器生成的输出可以与另一个编译器生成的代码相集成。

通过将机器码的创建放在 JIT 编译器中完成,这就实现了 .NET 的另一个目标,即平台可移植性。也就是说,一个平台上生成的 IL 可以在任何其他平台上运行,只要它有自己的框架,而且有一个能生成本地机器码的 JIT 编译器就可以。

另外,为了生成 IL,遵循 CLR 的编译器必须在每个代码模块中生成元数据(metadata)。元数据是一组表,以保证每个代码模块都是自描述的。表中记录了包含代码的程序集(assembly)的有关信息,并且还全面地描述了代码本身。这些信息包括可用类型、每个类型的名称、类型成员、类型作用域或可见性(visibility),以及其他类型特性。元数据的主要用途如下:

- 最重要的用途是供 JIT 编译器使用。JIT 编译器直接从元代码中收集编译所需的所有类型信息,并使用这些信息来进行代码验证,以保证程序正确运行。例如,JIT 将对调用参数与方法元数据中定义的参数进行比较,来确保该方法得到正确调用。
- 元数据用于垃圾收集过程(内存管理)。垃圾收集器(Garbage Collector, GC)通过使用元数据,可以知道某个对象中的字段何时引用了另一个对象,这样,GC 就可以确定哪些对象的内存可以回收,哪些对象的内存不能回收。
- .NET 提供了一组类,利用这些类可以从程序读取元数据。这些功能统称为反射(reflection)。这是一个很强大的特性,它允许程序在运行时查询代码,并根据结果作出相应的决定。本书后续章节将讲到,这是使用定制属性(custom attribute)的关键所在。定制属性是 C# 支持的一种构造,用于向程序添加定制元数据。