开发人员专业技术丛书

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

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。

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 写的这本书的的确确是一本“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DGI+；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三步曲：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，他是我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。

除封面署名外，参加本书翻译工作的还有：王树春、刘名臣、赵蓓、潘淼、刘立强、刘跃邦、蔡洪量、王三梅、苏金国、周兴汉、张练达、刘瑛、范松峰、杨健康、张莹、易竞、程龙、卢燮、江健、丁小峰、牛亚峰、刘鑫。

目　录

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[⊖]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。

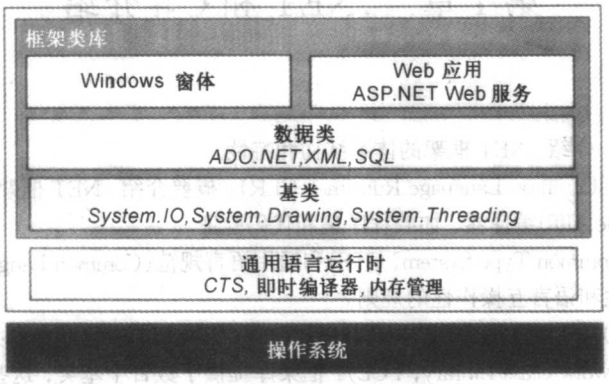

图 1-1　.NET 框架

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

⊖ 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。

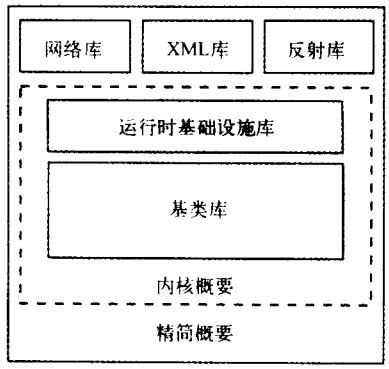

图 1-2 CLI 规范定义的架构

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 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[⊖]，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

⊖ 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)。

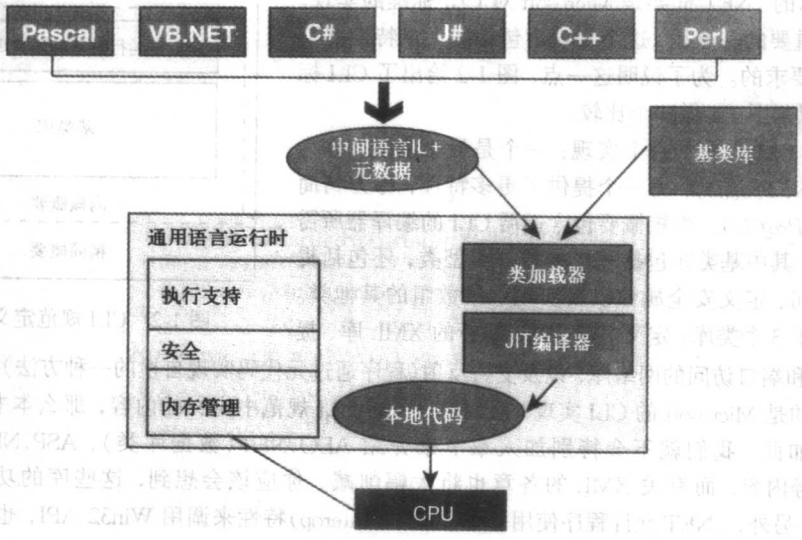

图 1-3　CLR 功能

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。