PROGRAMMER TO PROGRAMMER™

结合Visual Studio .NET学习本书，您可以迅速掌握C#基础知识和.NET Framework编程技术。无论您是编程新手还是专职程序员，本书都会助您成功。

Beginning Visual C#

Visual C# 入门经典

Karli Watson
David Espinosa 等著
杨 浩 译

清华大学出版社
http://www.tup.com.cn

Visual C#入门经典

Karli Watson 等著

David Espinosa

杨 浩 译

清华大学出版社

(京) 新登字 158 号
北京市版权局著作权合同登记号：01-2002-3189

内 容 简 介

C#是专用在.NET Framework 平台上进行开发的一门新型编程语言。它直接在强大而复杂的 C++语言基础上，继承了 C++的强大功能，但没有 C++那么复杂。C#还深受其他语言的影响，包括 Java 和 Delphi。C#博采众家之长，同时克服了其各自的缺点。

本书是您在学习编写 C#程序时的必备指南，它逐步阐明了 C#和.NET 的关键概念。本书全面介绍了 C#语言的语法，并论述了可以使用 C#构建的最常用的应用程序类型：Windows 应用程序、ASP.NET Web 应用程序和 Web 服务。C#编程与.NET 编程密不可分，实际上，C#编译器就是.NET Framework 的一部分，因此本书不仅讲述了 C#语言的语法，还阐述了如何在 C#中使用.NET 类建立应用程序。

本书是一本浅显易懂的 C#入门手册，适用于初学者，以及相对缺乏编程经验、但又想从不支持面向对象编程技术的语言转向.NET Framework 平台的程序员。如果您不喜欢阅读那些面向具有多年 C++编程经验的程序员的 C#书籍，那么本书正适合您。

Karli Watson David Espinosa et al: Beginning Visual C#

EISBN: 1-86100-758-2

图书在版编目(CIP)数据

Visual C#入门经典/(美)沃特森，(美)艾斯皮洛塞著；杨浩译.—北京：清华大学出版社，2002

书名原文：Beginning Visual C#

ISBN 7-302-06092-4

Ⅰ.V... Ⅱ.①沃...②艾...③杨... Ⅲ.C 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2002)第 090216 号

出 版 者：清华大学出版社(北京清华大学学研大厦，邮编 100084)

http://www.tup.com.cn

责任编辑：李阳

封面设计：康博

版式设计：康博

印 刷 者：北京市清华园胶印厂

发 行 者：新华书店总店北京发行所

开 本：787×1092 1/16 **印张**：47.75 **字数**：1222 千字

版 次：2002 年 12 月第 1 版 2002 年 12 月第 1 次印刷

书 号：ISBN 7-302-06092-4/TP・3637

印 数：0001～5000

定 价：88.00 元

使用本书的要求

显然，编写 C#程序最重要的事情是需要一个 C#编译器，.NET Framework SDK 带有该编译器，而.NET Framework SDK 可以从 Microsoft 网站上下载，其 URL 是：

http://msdn.microsoft.com/downloads/default.asp?URL=/code
/sample.asp?url=/MSDN-FILES/027/000/976/msdncompositedoc.xml

这是当前链接，但可能会有变化(这里因考虑到格式，把该 URL 分为两行，但 URL 应单独放在一行上)。注意这个软件目前大约有 131 MB，所以下载它时需要一定的时间。它可以运行在 Windows 2000、Windows XP 和 Windows NT4 下，包含了编写各种类型的 C#程序所需要的全部内容。.NET Framework 的简化版本可以安装在 Windows 95、98 和 ME 上，但该安装版本不包含我们所需要的许多工具。

但是，在本书中要常常用到 Visual Studio .NET 开发环境，它在许多方面都简化了 C#代码的编写，对编写 Windows 应用程序尤其有用，因为它包含一个可视化的窗体设计器。如果不使用 Visual Studio .NET，就不能完全领会本书中的内容。

Visual Studio .NET 的 1.0 版本目前有 3 种版次(Professional、Enterprise Developer 和 Enterprise Architect)，其中任何一个都可以用于编写本书中的代码。完整的信息可参阅 http://msdn.microsoft.com/vstudio/，包括系统要求。与.NET Framework SDK 一样，Visual Studio .NET 也可运行在 Windows 2000、XP 和 NT4 上。

另外，如果您的预算比较紧张，可以购买 Microsoft Visual C# .NET Standard Edition，它是一个仅支持 C#的开发环境，有许多 Visual Studio .NET 的功能，但不包含所有的功能。与 Visual Studio .NET 相比，Visual C# .NET Standard Edition 的功能有一些限制，所以本书所使用的全部功能不能从 Visual C# .NET Standard Edition 中获得。但 C#编译器不受限制，它独立于开发环境——只有开发环境的功能受到限制。

注意：

本书主要适用于 Visual Studio .NET 或 Visual C# .NET Standard Edition 用户。

客户支持

我们总是想知道您对本书的看法，您喜欢哪些内容，不喜欢哪些内容，这些信息都将有助于我们以后做得更好。如果您有什么意见和建议，请向 feedback@wrox.com 发邮件。但是，请您一定要在信中注明本书的书名。

如何下载本书中的例子代码

在您登录到 Wrox.com 站点 http://www.wrox.com/时，只需使用 Search 工具或使用书名列表就可以找到本书。接着在 Code 一栏中单击 Download，或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。

在单击下载本书的源代码时，会显示一个页面，其中有 3 个选项：

- 如果您已经是 Wrox Developer Community 的成员(已经注册了 ASPToday、C#Today 或 Wroxbase)，就可以用常用的用户名和密码登录，获取代码。
- 如果您还不是 Wrox Developer Community 的成员，该页面就会询问您是否要进行注册，以免费下载代码。另外，还可以从 Wrox 出版社免费下载几篇文章。进行注册后，我们会将本书的更新和新版本通知您。
- 第三种方式是完全绕过注册，仅下载代码。

本书的代码下载注册不是强制的，但如果您愿意进行注册，您的信息不会传送给任何第三方。要了解更多的信息，可以浏览我们的注册条款，这些都与下载的页面链接在一起。

在获得了下载的代码段后，会发现可以从该站点上下载的文件已使用 WinZip 进行了压缩。把文件保存到硬盘的一个文件夹中时，需要使用解压缩程序如 WinZip 或 PKUnzip 对该文件解压缩。在解压缩时，代码常常放在各自的章节文件夹中。在开始解压缩过程时，请确保软件(如 WinZip 或 PKUnzip)使用文件夹名。

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 support@wrox.com 发电子邮件，我们将会核查您的信息，如果是正确的，就把它传送到该书的勘误表页面上，或在本书的后续版本中采用。

要在网站上找到勘误表，可以登录http://www.wrox.com，通过 Advanced 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。

电子邮件支持

如果您希望直接就本书的问题向对本书知之甚多的专家咨询，那么就向 support@wrox. com 发电子邮件，在电子邮件的“主题”(Subject)栏中，加上本书的名称和 ISBN 的最后 4 位号码。典型的电子邮件应该包括下列内容：

- 在“主题”栏加上书的名称、ISBN 的最后 4 位数字(7582)和问题所在的页码。
- 在邮件的正文中加上您的姓名、联系信息和问题。

我们不会发给您垃圾邮件。我们只需要详细的情况以节省您和我们的宝贵时间。当您发送电子邮件时，它会直接到达以下支持链：

- 客户支持——您的消息会传送到我们的客户支持人员，他们是阅读信息的第一人。他们有常见问题的文件，会迅速回答一般性的问题。他们回答关于本书和网站的一般性问题。
- 编辑支持——更深的问题会转发到负责本书的技术编辑处。他(或她)具有编程或特殊产品的经验，能够回答某个主题的详细技术问题。
- 作者支持——最后，在编辑都不能回答问题的情况下(这种情况很少出现)，这些问题将转发到作者。我们试图保护作者不要从写作中分心，但是，我们也很愿意将特殊的问题转发给他们。所有的 Wrox 作者帮助支持他们的书籍。他们对客户和编辑回复电子邮件，所有的读者都会从中受益。

Wrox 支持过程只能提供直接与已出版的图书相关的问题。对于超出此范围的问题可以通过 http://p2p.wrox.com/ 论坛的团体列表来提供支持。

P2P.WROX.COM

P2P 邮件列表是为作者和读者之间讨论而设立的。我们在邮件列表、论坛和新闻组中提供“程序员到程序员的支持” (programmer to programmer support)，还包括一对一的电子邮件支持系统。如果把问题发送给 P2P，就可以相信，您的问题不仅仅是由支持专家解答，而且还要提供给邮件列表中的许多 Wrox 作者和其他业界专家。在 p2p.wrox.com 上，可以从许多不同的列表中获得帮助，不仅在阅读本书时获得帮助，还可以在开发应用程序时获得帮助。 在网站的.NET 类别中，最适合本书的是 beginning_c_sharp 和 c_sharp(更高级的讨论)列表。

要订阅一个邮件列表，可以遵循以下步骤：

(1) 进入 p2p.wrox.com。

(2) 从左侧的菜单栏中选择合适的列表。

(3) 单击想加入的邮件列表。

(4) 按照指示订阅和填写电子邮件地址和密码。

(5) 回复接收到的确认电子邮件。

(6) 使用订阅管理器加入更多的列表，设置自己的邮件设置。

为什么这个系统提供最佳的支持

您可加入该邮件列表中，也可以每周分类接收它们。如果您没有时间或设备接收该邮件列表，可以搜索我们的在线文档。垃圾邮件和广告邮件会被删除，您自己的电子邮件地址会被独特的 Lyris 系统保护起来。任何加入或退出列表的查询，或者与列表相关的一般问题，都应发送到 listsupport@p2p.wrox.com。

练习的答案

本书中练习的答案还可以在 P2P 网站上找到。要查看这些答案，需要注册 Beginning C#练习讨论列表。注册 P2P 后，应遵循下面的步骤：

(1) 在 P2P 主页上单击 Exercises 链接(在左边的 Categories 列表中)。

(2) 输入自己的电子邮件地址，登录 P2P。

(3) 从 Subscribe to an exercises discussion list 文本框中选择 Beginning C#(csharp_beginners_exercises)，再单击旁边的 Subscribe 按钮。

(4) 输入并确认使用这个列表的密码(如果不想使用密码，也可以不输入任何内容)，再选择需要的订阅类型(消息、每日摘要等)，并单击 Subscribe 按钮。

(5) 现在 P2P 会给您发送一封用于确认的电子邮件；您应回复这个电子邮件，确认自己的订阅，接着，P2P 会再发送一封电子邮件，确认您已订阅了列表。

(6) 返回 Exercises 页面(可能需要再次登录)，单击 csharp_beginners_exercises 链接。

(7) 现在屏幕上会显示一个列表，其中包含了所有带练习的章节，此时您可以单击任一章的名称，查看该章的练习。每个练习都有一个链接，单击该链接，就可以查看其答案！

出版者的话

随着国际互联网的快速崛起和迅猛发展，计算机之间的互联需求越来越迫切，而目前计算机硬件设备的不兼容性严重束缚了互联网的发展，引发了新一轮的跨平台软件的开发浪潮。软件商纷纷推出新的战略规划和解决方案，Microsoft 提出的.NET 战略就是其中的经典之作。

在经历这场浪潮的洗涤和考验过程中，全球的软件开发人员都迫切需要了解新的软件技术和开发思路。为了满足国内 IT 从业人员的需要，清华大学出版社从 Wrox 出版公司引进了若干套编程系列丛书，“入门经典”系列是其中不可或缺的入门之作。作为世界著名的编程技术图书的出版公司，Wrox 推出的这套“入门经典”系列丛书主要面向编程的初学者、需要了解.NET 策略的程序员，以及需要迅速掌握多门编程工具的程序员。该丛书依旧秉承了 Wrox 公司“由程序员为程序员而著（Programmer to Programmer）”的创作理念，每本书均由世界顶级的编程高手执笔。他们站在资深程序员的高度，循序渐进地为初学者讲述了.NET 的理念和构架、编程基本思想、编程语言基础、程序的控制和调试、Windows 应用程序的开发、对象编程技术、数据库访问技术、Web 程序开发和.NET 构架应用等最新的软件开发知识，同时辅以大量操作性强的程序为示例，为读者提供了清晰的编程思路和宝贵的编程经验。

为了保证该系列丛书的质量，清华大学出版社迅速组织了一批位于 IT 开发领域前沿的专家学者进行翻译，经过编辑人员的进一步加工整理后，现陆续奉献给广大读者。

读者可以从 www.wrox.com 网站下载所需的源代码和获取相关的技术支持。同时，也欢迎广大读者参与 p2p.wrox.com 网站上的在线讨论，与世界各地的编程人员交流读书感受和编程体验。

前　言

为什么要学习 C#？如果您购买了本书，就肯定已经回答了这个问题，但是这个问题仍值得反复思考。C#是 Microsoft 推出的一种全新语言，是为其全新的平台.NET Framework 设计的。因此，在 Microsoft 环境中开发应用程序时，C#可能是所有语言中的首选。仅此一项，就使 C#成为程序员要学习的首要语言。但更重要的是，C#是一种设计非常优秀的语言，它可以使程序员养成良好的编程习惯(特别是面向对象的编程方式)。C#直接在强大而复杂的 C++语言的基础上，继承了 C++的强大功能，但没有 C++那么复杂。C#还深受其他语言的影响，包括 Java 和 Delphi。C#爱好者相信，C#博采众家之长，同时克服了其各自的缺点。

本书将从头开始讲述 C#，您不需要有任何编程方面的经验。本书将全面介绍 C#语言的语法，然后论述可以使用 C#构建的最常用的应用程序类型，如 Windows 应用程序、ASP.NET Web 应用程序和 Web 服务。首先要强调的是，C#编程与.NET 编程密不可分(实际上，C#编译器就是.NET Framework 的一部分)，在 C#中进行的任何工作都要依赖于.NET Framework。因此，本书不仅要讲述 C#语言的语法，还要阐述如何在 C#中使用.NET 类构建应用程序。所以，您学习了 C#后，再学习其他.NET 语言(例如 Visual Basic .NET 或 Managed C++)就不会有什么困难了。

本书读者对象

本书主要针对没有什么经验、但希望学习使用 C#构建应用程序的程序员。对以前的语言如 Visual Basic 有一定了解的开发人员，也可以把本书当作 C#的一本实用教材。如果您不喜欢那些面向具有多年 C++编程经验的程序员的 C#书籍，那么本书正适合您。

注意：

这是一本 Wrox 入门经典系列的图书，所以本书从头开始讲述 C#的内容。如果您有一定的 C++、VB 或 Java 编程经验，可以阅读由清华大学出版社引进并已出版的《C#高级编程(第二版)》一书(EISBN 1-86100-704-3)，这样在一个较高的层次上继续学习。

本书非常适合于以下两类初学者：

- 把 C#作为第一种编程语言的编程新手。前面说过，C#是学习编程的一种极佳语言，本书将帮助您应对某些新概念的挑战！
- 有另一种语言的编程经验，但对.NET 编程较为陌生的程序员。.NET 在编程概念中代表着一种革新，如果您不熟悉这项技术，.NET 中面向对象编程方式的极端重要性就会被忽视。如果您以前所用的语言不支持(或仅部分支持)OOP，您就会欣喜地发现，本书中有一整篇的内容都在讲述 OOP。

本书内容

本书是根据 2002 年 1 月发布的.NET Framework 1.0 版本和 2002 年 2 月发布的 Visual Studio .NET 1.0 版本编写的，这些版本都是这些产品的发布版本，最近不会有太大的变化。

本书分为 7 个主要部分：

入门篇

前两章简要介绍在编写 C#应用程序之前需要理解的主要概念，然后用 Visual Studio .NET 创建一个非常简单的 C#程序，为我们完成大部分繁杂的工作。

C#语言基础篇

第 3～7 章讲述了 C#语言的基础知识，介绍了如何在 C#变量中存储数据，如何用条件分支和循环结构控制程序的流程，以及如何用函数构建程序。

对象编程篇

对象和面向对象编程(OOP)规则在 C#中起着非常重要的作用，所以第 8～12 章将介绍 OOP 及其体系结构，并讨论如何在 C#代码中使用对象。

Windows 窗体篇

前面各篇主要讨论如何创建简单的控制台应用程序，使您对 C#语言本身有一个全面的了解。而在本篇中，我们将详细研究如何用 C#创建真正的 Windows 应用程序。

.NET Framework 编程篇

如前所述，在 C#中做的几乎所有工作都依赖于.NET Framework。本篇将讨论需要使用.NET Framework 的类时的一些重要问题，包括访问数据库，在本地机器或网络上处理文件。还将详细探讨.NET 编程的两个特定功能：程序集(.NET 程序的部署单元)和属性(一种.NET 功能，允许提供程序各部分的其他信息)。

C#和 Web 篇

完成了前面的内容后，接着简要论述一个全新的主题，但该主题是整个.NET Framework 的一个组成部分：Internet 编程。在本书的最后一篇中，将介绍 ASP.NET 和 Web 服务。ASP.NET 可以用 C#编写动态的网页，而 Web 服务可以使应用程序通过 Internet 交换信息。

注意：

本书中的大多数章节在最后都附有一些练习，帮助您在学习 C#时理解该章的内容。这些练习的答案可在 P2P 网站 http://p2p.wrox.com 上找到。本前言的最后将说明如何注册 P2P。

第1章 C# 简 介

本章将概述 C#和.NET Framework，我们对这些技术的理解，使用它们的原因，以及它们之间的相互关系。

首先讨论一下.NET Framework。这是一种新技术，它包含的许多概念初看起来都不是很容易掌握的(主要因为该架构在应用程序开发环境中引入了执行操作的一种新方式)。也就是说，我们必须在很短的时间里介绍许多新概念，但是，快速浏览这些基础知识对于理解如何利用 C#进行编程是非常重要的，所以这是不可避免的。本书的后面将详细论述这里提到的许多论题。

之后，本章将讨论 C#本身，包括它的起源和与 C++的类似之处。

最后，介绍本书使用的主要工具：Visual Studio .NET (VS)。

1.1 什么是.NET Framework

.NET Framework 是 Microsoft 为开发应用程序而创建的一个富有革命性的新平台。

这句话最有趣的地方是它的含糊不清，但这是有原因的。首先，注意这句话没有说“在 Windows 操作系统上开发应用程序”。尽管.NET Framework 发布的第一个版本运行在 Windows 操作系统上，但以后将推出运行在其他操作系统上的版本，这些操作系统包括 FreeBSD、Linux、Macintosh，甚至个人数字助手(PDA)类设备。使用这项技术的一个主要原因是它可以作为集成各种操作系统的方式。

另外，上面给出的.NET Framework 定义并没有限制应用程序的类型。这是因为本来就没有限制。.NET Framework 可以创建 Windows 应用程序、Web 应用程序、Web 服务和其他各种类型的应用程序。

.NET Framework 的设计方式保证它可以用于各种语言，包括本书要介绍的 C#语言，以及 C++、Visual Basic、 JScript，甚至一些旧的语言，如 COBOL。为此推出了这些语言的.NET 版本：Managed C++、Visual Basic .NET、JScript .NET 和 J#等，目前还在不断推出更多的.NET 版本的语言。所有这些语言都可以访问.NET Framework，它们还可以彼此交互。C#开发人员可以使用 Visual Basic .NET 程序员编写的代码，反之亦然。

所有这些都提供了意想不到的多样性，这也是.NET Framework 具有诱人前景的部分原因。

1.1.1 .NET Framework 的内容

.NET Framework 主要包含一个非常大的代码库，在客户语言(如 C#)中通过面向对象编程技术(OOP)可以使用这些代码。这个库分为不同的模块，这样就可以根据希望得到的结果来选择使用其中的各个部分。例如，一个模块包含 Windows 应用程序的构件，另一个模块包含连网的代码块，还有一个模块包含 Web 开发的代码块。一些模块还分为更特殊的子模块，例如在 Web

开发模块中，有用于建立 Web 服务的子模块。

其目的是，不同的操作系统可以根据自己的特性，支持其中的部分或全部模块。例如，PDA 支持所有的核心.NET 功能，但不需要某些更深奥的模块。

部分.NET Framework 库定义了一些基本类型。类型是数据的一种表达方式，指定其中最基础的部分(例如 32 位有符号的整数)，以便于使用.NET Framework 在各种语言之间进行交互操作。这称为通用类型系统(Common Type System，CTS)。

除了支持这个库以外，.NET Framework 还包含.NET 公共语言运行时(Common Language Runtime，CLR)，它负责管理用.NET 库开发的所有应用程序的执行。

1.1.2 如何用.NET Framework 编写应用程序

使用.NET Framework 编写应用程序，就是使用.NET 代码库编写代码(使用支持.NET Framework 的任何一种语言)。本书中所有的范例都使用 VS 进行开发，VS 是一种强大的集成开发环境，支持 C#(以及托管和非托管 C++、Visual Basic .NET 和其他一些语言)。这个环境的优点是便于把.NET 功能集成到代码中。我们创建的代码完全是 C#代码，但使用.NET Framework，并在需要时利用 VS 中的其他工具。

为了执行 C#代码，必须把它们转换为目标操作系统能够理解的语言，即本机代码，这种转换称为编译代码，由编译器执行。但在.NET Framework 下，这个过程分为两个阶段。

1. MSIL 和 JIT

在编译使用.NET Framework 库的代码时，不是立即创建操作系统特定的本机代码，而是把代码编译为 Microsoft 中间语言(Microsoft Intermediate Language，MSIL)代码，这些代码不专用于任何一种操作系统，也不专用于 C#。其他.NET 语言，如 Visual Basic .NET 也可以在第一阶段编译为这种语言，当使用 VS 开发 C#应用程序时，编译过程就由 VS 完成。

显然，要执行应用程序，必须完成更多的工作，这是 Just-In-Time (JIT)编译器的任务，它把 MSIL 编译为专用于 OS 和目标机器体系结构的本机代码。这样 OS 才能执行应用程序。这里编译器的名称 Just-In-Time 反映了 MSIL 仅在需要时才编译的事实。

过去，常常需要把代码编译为几个应用程序，每个应用程序用于特定的操作系统和 CPU 结构。这通常是一种优化形式(例如，为了让代码在 AMD 芯片上运行得更快)，但有时是非常重要的(例如对于工作在 Win9x 和 WinNT/2000 环境下的应用程序)。现在就不必要了，因为顾名思义，JIT 编译器使用 MSIL 代码，而 MSIL 代码是独立于机器、操作系统和 CPU 的。目前有几种 JIT 编译器，每种编译器都用于不同的体系结构，我们总能找到一个合适的编译器创建所需的本机代码。

这样，用户需要做的工作就比较少了。实际上，我们可以不考虑与系统相关的细节，而把注意力集中在代码的功能上。

2. 程序集

在编译应用程序时，创建的 MSIL 代码存储在程序集中，程序集包括可执行的应用程序文件(其扩展名是.exe，这些文件可以直接在 Windows 上运行，不需要其他程序)和其他应用程序使用的库(其扩展名是.dll)。

除了包含 MSIL 外，程序集还包含元信息(即程序集中包含的数据的信息，也称为元数据)和可选的资源(MSIL 使用的其他数据，例如声音文件和图片)。元信息允许程序集是完全自我描述的。不需要其他信息就可以使用程序集，也就是说，我们不会遇到下述情形：不能把需要的数据添加到系统注册表中，而这种情形在使用其他平台进行开发时常常出现。

因此，部署应用程序非常简单，只需把文件复制到远程计算机上的目录下即可。因为不需要目标系统上的其他信息，所以只需从该目录中运行可执行文件即可(假定安装了.NET CLR)。

当然，不必把运行应用程序所需要的所有信息都安装到一个位置。可以编写一些代码，执行多个应用程序所要求的任务。此时，通常把这些可重用的代码放在所有应用程序都可以访问的位置。在.NET Framework 中，这就是全局程序集缓冲存储器(Global Assembly Cache ,GAC)，把代码放在这个缓冲存储器中很简单，只需把包含代码的程序集放在包含该缓冲存储器的目录下即可。

3. 托管代码

在把代码编译为 MSIL，再用 JIT 编译器把它编译为本机代码后，CLR 的任务还没有全部完成。用.NET Framework 编写的代码在执行(这个阶段通常称为运行时(runtime))时是托管的。即 CLR 管理着应用程序，其方式是管理内存、处理安全性，以及允许进行跨语言调试等。相反，不在 CLR 控制之下运行的应用程序是非托管的，某些语言如 C++可以用于编写这类应用程序，例如，访问操作系统的低级功能。但是，利用 C#只能编写在托管环境下运行的代码。我们将使用 CLR 的托管功能，让.NET 自己与操作系统进行交互。

4. 垃圾回收

托管代码最重要的一个功能是垃圾回收(garbage collection)的概念。这种.NET 方法可确保应用程序不再使用某些内存时，这些内存就会被完全释放。在.NET 推出以前，这项工作主要由程序员负责，代码中的几个简单错误会把大块内存分配到错误的地方，使这些内存神秘失踪。这通常意味着计算机的速度逐渐减慢，最终导致系统崩溃。

.NET 垃圾回收会频繁检查计算机内存，从中删除不再需要的内容。它没有设置时间帧，可能一秒钟内会进行上千次的检查，也可能几秒钟检查一次，或者随时进行检查，但可以肯定进行了检查。

这里要给程序员一些提示。因为这项工作在不可预知的时间进行，所以在设计应用程序时，必须记得要进行这样的检查。需要许多内存才能运行的代码应自己执行这样的检查，而不是坐等系统进行垃圾回收，但这不像听起来那样难。

5. 把它们组合在一起

在继续学习之前，我们先总结一下上述创建.NET 应用程序所需要的步骤。

(1) 使用某种.NET 兼容语言(如 C#)编写应用程序代码,如图 1-1 所示。

(2) 代码被编译为 MSIL，存储在程序集中，如图 1-2 所示。

图 1-1

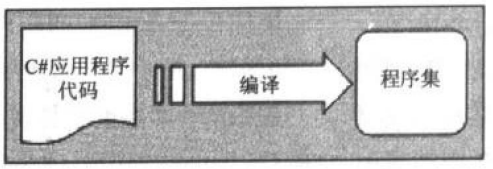

图 1-2

(3) 在执行代码时(如果这是一个可执行文件，就自动运行，或者在其他代码使用它时运行)，首先必须使用 JIT 编译器将代码编译为本机代码，如图 1-3 所示。

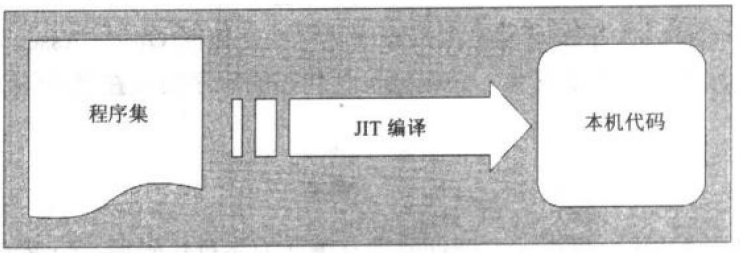

图 1-3

(4) 在托管的 CLR 环境下运行本机代码，以及其他应用程序或过程，如图 1-4 所示。

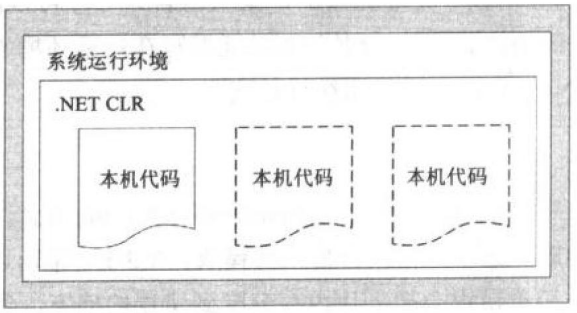

图 1-4

6. 链接

在上述过程中还有一点要注意。在第(2)步中编译为 MSIL 的 C#代码不需要包含在单独的文件中，可以把应用程序代码放在多个源代码文件中，再把它们编译到一个程序集中。这个过程称为链接，是非常有用的。原因是处理几个较小的文件要比处理一个大文件简单得多。可以把逻辑上相关的代码分解到一个文件中，以便单独处理它，这也更易于在需要代码时找到它们，让开发小组把编程工作分解为可管理的块，让每个人编写一小块代码，而不会损害已编写好的代码部分或其他人正在处理的代码部分。

1.2 什么是 C#

如上所述，C#是可用于创建要运行在.NET CLR 上的应用程序的语言之一，它从 C 和 C++ 语言演化而来，是 Microsoft 专门为使用.NET 平台而创建的一门语言。因为 C#是近期发展起来

的，所以它吸取了以前的教训，考虑了其他语言的许多优点，并解决了它们的问题。

使用 C#开发应用程序比使用 C++简单，因为其语法比较简单。但是，C#是一种强大的语言，在 C++中能完成的任务在 C#中也能完成。如前所述，C#中与 C++比较高级的功能等价的功能(例如直接访问和处理系统内存)，只能在标记为“不安全”的代码中使用。这个高级编程技术是非常危险的(正如它的名称)，因为它可能覆盖系统中重要的内存块，导致严重的后果。因此，本书不讨论这个问题。

C#代码常常比 C++略长一些。这是因为 C#是一种类型安全的语言(与 C++不同)。在外行人看来，这表示一旦为某些数据指定了类型，就不能转换为另一种不相关的类型。所以，在类型之间转换时，必须遵守严格的规则。执行相同的任务时，用 C#编写的代码通常比 C++长。但 C#代码更健壮，调试过程也比较简单，.NET 总是可以随时跟踪数据的类型。在 C#中，不能完成诸如“把 4 字节的内存放在数据中，并把它解释为 X”等的任务，但这并不是一件坏事。

C#只是用于.NET 开发的一种语言，但在我看来，这是最好的一种语言。C#的优点是，它是惟一为.NET Framework 而设计的语言，是在移植到其他操作系统上的.NET 版本中使用的主要语言。要使语言如 VB.NET 尽可能类似于其以前的语言，且仍遵循 CLR，就不能完全支持.NET 代码库的某些功能。但 C#能使用.NET Framework 代码库提供的每种功能。

1.2.1 用 C#能编写什么样的应用程序

如前所述，.NET Framework 没有限制应用程序的类型。C#使用.NET Framework，所以也没有限制应用程序的类型。这里仅讨论几种常见的应用程序类型。

- Windows 应用程序 这些应用程序如 Microsoft Office，有我们很熟悉的 Windows 外观和操作方式，使用.NET Framework 的 Windows Forms 模块就可以生成这种应用程序。Windows Form 模块是一个控件(例如按钮、工具栏、菜单等)库，其中的控件可以用于建立 Windows 用户界面(UI)。
- Web 应用程序 这些是 Web 页，可以通过任何 Web 浏览器查看。.NET Framework 包括一个动态生成 Web 内容的强大系统，允许个性化、实现安全性等。这个系统叫作 Active Server Pages.NET (ASP.NET)，我们可以使用 C#通过 Web Forms 创建 ASP.NET 应用程序。
- Web 服务 这是创建各种分布式应用程序的新方式，使用 Web 服务可以通过 Internet 虚拟交换数据。无论使用何种语言创建 Web 服务，也无论 Web 服务驻留在什么系统上，都使用一样简单的语法。

这些类型也需要某种形式的数据库访问，这可以通过.NET Framework 的 Active Data Objects.NET(ADO.NET)部分来实现。也可以使用许多其他资源，例如创建连网组件、输出图形、执行复杂数学任务的工具等。

1.2.2 本书中的 C#

本书介绍了 C# 语言的语法和用法，但不过分强调.NET Framework。这是必需的，因为我们不能完全没有 C# 编程基础就使用.NET Framework。首先介绍一些比较简单的内容，把面向对象编程(Object of Oriented Programming , OOP)的问题放在基础知识的后面论述。本书假定您

没有一些编程的知识，这些是首要的规则。

本书还将介绍如何开发上一节列出的各种类型的应用程序，本书的第 4 篇介绍如何进行 Windows Forms 编程，第 5 篇介绍其他有趣的.NET 问题(例如访问数据库)，第 6 篇介绍 Web 应用程序和 Web 服务编程。最后，详细介绍几个使用本书前面的内容编写的案例分析。

1.3 Visual Studio .NET

本书使用 Visual Studio .NET (VS)完成全部的 C#开发，包括简单的命令行应用程序，以及比较复杂的工程类型。VS 不是开发 C#应用程序所必需的，但使用它可以使任务更简单一些。可以在标准的文本编辑器中处理 C#源代码文件，例如常见的 Notepad，再使用命令行应用程序(是.NET Framework 的一部分)把代码编译到程序集。但是，为什么要使用功能全面的 VS 呢？

下面列出的一些使 VS 成为.NET 开发的首选工具的功能。

- VS 可以自动执行编译源代码的步骤，同时可以完全控制重写它们时应使用的任何选项。
- VS 文本编辑器可以配合 VS 支持的语言(包括 C#)，这样就可以智能检测错误，在输入代码时给出合适的推荐代码。
- VS 包括 Windows Forms 和 Web Forms 设计器，允许进行 UI 元素的简单拖放设计。
- 在 C#中，许多类型的工程都可以用已有的“模板”代码来创建，不需要从头开始。各种代码文件通常已经为我们准备好了，减少了从头开始一个工程所花的时间。
- VS 包括几个可自动执行常见任务的向导，它们可以在已有的文件中添加合适的代码，而不需要考虑(在某些情况下)语法的正确性。
- VS 包含许多能够可视化和导航工程中元素的强大工具，它们可以是 C#源文件代码，也可以是其他资源，例如位图图像或声音文件。
- 除了在 VS 中编写应用程序比较简单外，还可以在其中创建部署工程，以易于为客户提供代码，并方便地安装该工程。
- 在开发工程时，VS 可以使用高级调试技巧，例如能一次调试一行指令，并监视应用程序的状态。

C#还有许多功能，希望您能掌握它们！

Visual C# .NET Standard Edition

Visual C# .NET Standard Edition 是 Visual Studio .NET Professional 的一个简化版本，其价格也较低。它提供了许多与 Visual Studio .NET Professional 相同的功能，但缺乏其他一些功能，可是这并不妨碍使用 Standard Edition 来学习本书的内容。

在本书中，除非特别说明，否则术语"Visual Studio .NET" (或简称"VS")指代 Visual Studio .NET 和 Visual C# .NET Standard Edition 这两个版本中的任意一个。有时该术语是指其中的一种版本，而不是另一种版本，但此时我们会仔细地把它们标识出来，这样，即使您拥有的不是这里所说的版本，也不会弄混它们。

VS 解决方案

在使用 VS 开发应用程序时，可以通过创建解决方案来完成。在 VS 术语中，解决方案不仅仅是一个应用程序，它还包含工程，可以是 Windows Forms 工程、Web Form 工程等。但是，解决方案可以包含多个工程，这样，即使它们最终在硬盘上的多个位置编译为多个程序集，也可以把相关的代码组合到一个地方。

这是非常有用的，因为它可以处理“共享”代码(这些代码放在 Global Assembly Cache 中)，同时，应用程序也使用这段共享代码。在只使用一个开发环境时，调试代码是非常容易的，因为可以在多个代码块中单步调试指令。

1.4 小结

本章简要介绍了.NET Framework，并讨论了如何轻松地创建各种强大的应用程序。还探讨了把用 C#等语言编写的代码转换为可应用的应用程序所需要做的工作，以及使用在.NET Common Language Runtime 下运行的托管代码的优点。

本章还阐述了 C#的实质，以及它与.NET Framework 的关系，描述了进行 C#开发使用的工具——Visual Studio .NET。

第 2 章介绍如何使用 VS 运行 C#代码，给出足够的基础知识，并集中讨论 C#语言本身，而不是过多地讨论 VS 的工作原理。

第2章　编写C#程序

本章用一定的篇幅讨论 C#是什么，以及它是如何适应.NET Framework 的，然后编写一些代码。本书主要使用 Visual Studio .NET(VS)，所以首先介绍这个开发环境的一些基础知识。VS 是一个非常复杂、庞大的产品，可能会使初学者感到畏惧，但使用它创建简单的应用程序却非常容易。在本章中，开始使用 VS 时，不需要了解许多知识就可以编写 C#代码。本书的后面将介绍 VS 能执行的一些更复杂的操作，现在仅介绍基础知识。

介绍完 VS 之后，将把两个简单的应用程序组合在一起。现在不要过多地考虑代码，后面将通过应用程序的创建过程，证明这个过程是有效的。

下面要创建的第一个应用程序是一个简单的控制台应用程序。控制台应用程序没有使用图形化的 Windows 环境，所以不需要担心按钮、菜单、用鼠标指针进行的交互等，而是在一个命令行窗口中运行应用程序，用更简单的方式与它交互。

第二个应用程序是一个 Windows 窗体应用程序，其外观和操作方式对 Windows 用户来说会非常熟悉，而且该应用程序创建起来不需要花费太多的精力。但所需代码的语法比较复杂，尽管在许多情况下，并不需要考虑细节。

本书后面的两个部分也使用这两种应用程序类型，但开始时略微强调一下控制台应用程序。在学习 C#语言时，不需要了解 Windows 应用程序其他的灵活性能。控制台应用程序的简单性可以让我们集中精力学习语法，而无需考虑应用程序的外观和操作方式。

2.1　Visual Studio .NET 开发环境

首次加载 VS 时，会立即显示一系列窗口以及一组菜单和工具栏图标，其中大多数窗口是空的。本书将使用这些窗口、菜单和工具栏，您不久就会很熟悉它们。

如果是第一次运行 VS，则屏幕上会为以前使用过这个开发环境的用户显示一个参数列表，这些参数的默认设置很不错，现在就使用它们——其中的参数都是可以修改的。

VS 环境布局是完全可定制的，但默认设置很适合我们。其布局如图 2-1 所示。

在 VS 启动时，主窗口显示了一个介绍性的“开始页面”，该主窗口还会显示所有的代码。这个窗口有几个标签，单击文件名，就可以在文件之间切换。这个窗口也有其他功能：它可以显示图形用户界面，该界面可用于设计工程、纯文本文件、HTML 以及各种内置于 VS 的工具。本书将陆续介绍它们。

在主窗口的上面，有工具栏和 VS 菜单。这里有几个不同的工具栏，其功能包含保存和加载文件，建立和运行工程，以及调试控件等。在需要使用这些工具栏时将会讨论它们。